ETSI TS 126 253 V18.1.0 (2024-07)

LTE; 5G;
Codec for Immersive Voice and Audio Services - Detailed Algorithmic Description incl. RTP payload format and SDP parameter definitions
(3GPP TS 26.253 version 18.1.0 Release 18)

Reference
RTS/TSGS-0426253vi10

Keywords
5G,LTE

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found under https://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Intellectual Property Rights 2
Legal Notice 2
Modal verbs terminology 2
Foreword 23
1 Scope 25
2 References 25
3 Definitions of terms, symbols and abbreviations 26
3.1 Terms 26
3.2 Symbols 26
3.3 Abbreviations 27
4 General description of the coder 28
4.1 Introduction 28
4.2 IVAS codec overview 28
4.2.1 General 28
4.2.2 Mono (EVS-compatible) Operation 29
4.2.3 Stereo Operation 30
4.2.4 Scene-based Audio (SBA, Ambisonics) Operation 30
4.2.5 Metadata-assisted Spatial Audio (MASA) Operation 31
4.2.6 Objects (Independent Streams with Metadata, ISM) Operation 31
4.2.7 Multi-Channel (MC) Operation 31
4.2.8 Combined Objects and SBA (OSBA) Operation 31
4.2.9 Combined Objects and MASA (OMASA) Operation 32
4.2.10 Discontinuous Transmission (DTX) Operation 32
4.3 Input/output audio configurations 32
4.3.1 Input/output audio sampling rate 32
4.3.2 Input/output audio formats 32
4.4 Algorithmic delay 33
4.5 Rendering overview 34
4.5.1 Rendering introduction 34
4.5.2 Internal IVAS renderer 35
4.5.3 External IVAS renderer 35
4.5.4 Interface for external rendering 36
4.5.5 Split rendering 36
4.6 Organization of the rest of the Technical Standard 36
5 Functional description of the encoder 37
5.1 Encoder overview 37
5.2 Common processing and coding tools 38
5.2.1 High-pass filtering 38
5.2.2 Core-coder processing 38
5.2.2.1 Overview 38
5.2.2.2 Core-coder front pre-processing 39
5.2.2.2.1 General 39
5.2.2.2.2 Sample rate conversion 39
5.2.2.2.3 Pre-emphasis 40
5.2.2.2.4 Spectral analysis 40
5.2.2.2.5 Signal activity detection 40
5.2.2.2.6 Bandwidth detector 40
5.2.2.2.7 Time domain transient detection 41
5.2.2.2.8 Linear prediction analysis 42
5.2.2.2.9 Open-loop pitch analysis 43
5.2.2.2.10 Coding mode determination 43
5.2.2.2.11 Speech/Music classification 43

5.2.2.2.12 Core-coder technology pre-selection 54
5.2.2.3 Core-coder modules 61
5.2.2.3.1 Core-coder pre-processing 61
5.2.2.3.2 LP based coding 66
5.2.2.3.3 MDCT based coding 94
5.2.2.3.4 Switching coding modes 110
5.2.2.3.5 DTX/CNG operation 113
5.2.3 Common audio coding tools 115
5.2.3.1 General 115
5.2.3.2 Single Channel Element (SCE) 115
5.2.3.3 Channel Pair Element (CPE) 116
5.2.3.4 Multichannel Coding Tool (MCT) 118
5.2.3.4.1 Summary of the system 118
5.2.3.4.2 MCT configurations 119
5.2.3.4.3 Encoder single channel preprocessing up to whitened spectrum 119
5.2.3.4.4 Joint channel Encoding System Description 120
5.2.3.4.5 IGF for multi-channel processing 125
5.2.3.4.6 Bitrate distribution 125
5.2.3.4.7 Quantization and coding of each channel 126
5.2.4 Common spatial metadata coding tools 126
5.2.4.1 General 126
5.2.4.2 Spatial metadata composition 126
5.2.4.3 Direction metadata coding tools 127
5.2.4.3.1 Overview 127
5.2.4.3.2 Direction metadata quantization 127
5.2.4.3.3 Direction metadata raw encoding 130
5.2.4.3.4 Direction metadata entropy encoding tools 130
5.2.4.4 Diffuseness and energy ratio coding methods 131
5.2.4.4.1 Diffuseness and energy ratio definitions 131
5.2.4.4.2 Diffuseness parameter quantization 131
5.2.4.4.3 Diffuseness and energy ratio indices coding 132
5.2.4.4.4 Diffuseness and energy ratio coding with two concurrent directions 133
5.2.4.5 Direction metadata coding methods 135
5.2.4.5.1 Entropy coding 1 (EC1) 135
5.2.4.5.2 Entropy coding 2 (EC2) 140
5.2.4.5.3 Entropy coding 3 (EC3) 141
5.2.4.6 Coherence coding 145
5.2.4.6.1 Spread coherence coding 145
5.2.4.6.2 Surround coherence coding 148
5.2.4.7 DTX coding 149
5.2.5 Modified Discrete Fourier Transform (MDFT) Analysis Filter Bank 149
5.2.5.1 General 149
5.2.5.2 Modified Discrete Fourier Transform (MDFT) 150
5.2.5.3 Filter bank responses 150
5.2.5.3.1 General 150
5.2.5.3.2 Magnitude responses for 5 ms stride operation 151
5.2.5.3.3 Responses for sampling rates below 48 kHz 152
5.3 Stereo audio operation 153
5.3.1 Stereo format overview 153
5.3.2 Unified stereo 153
5.3.2.1 Overview 153
5.3.2.2 Common unified stereo tools 153
5.3.2.2.1 Inter-Channel Bandwidth Extension (IC-BWE) 153
5.3.2.2.2 Front VAD 164
5.3.2.3 TD-based stereo 165
5.3.2.3.1 Inter-channel Alignment (ICA) 165
5.3.2.3.2 TD stereo coder overview 177
5.3.2.3.3 Time-domain stereo downmix 178
5.3.2.3.4 Near out-of-phase operation (NOOP) 180
5.3.2.3.4.1 NOOP signal detection 180
5.3.2.3.4.2 NOOP sub-mode selection 182
5.3.2.3.4.3 NOOP signal coding 183

5.3.2.3.4.4 Adaptive mixing ratio for the NOOP signal 184
5.3.2.3.5 LP filter coherence 185
5.3.2.3.6 Open-loop pitch coherence 187
5.3.2.3.7 Excitation coding in the secondary channel using two or four subframes 189
5.3.2.3.8 Bit allocation 191
5.3.2.3.9 Quantization of LP coefficients for the secondary channel 192
5.3.2.4 DFT-based stereo 194
5.3.2.4.1 General 194
5.3.2.4.2 Time Domain ITD compensation 195
5.3.2.4.3 STFT analysis 196
5.3.2.4.4 GCC-PHAT ITD estimation 197
5.3.2.4.4.5 Refined ITD control mechanism 201
5.3.2.4.5 FD circular time-shift 203
5.3.2.4.6 Stereo parameters estimation 203
5.3.2.4.7 IPD calculation, stabilization and encoding scheme 203
5.3.2.4.8 Calculation of the side and residual prediction gains 207
5.3.2.4.9 Stereo parameter coding 208
5.3.2.4.10 Active Downmix 212
5.3.2.4.11 STFT synthesis 213
5.3.2.4.12 Residual coding 215
5.3.2.4.12.1 Overview 215
5.3.2.4.12.2 Adaptive residual signal encoding 218
5.3.2.4.12.2.1 Adaptive residual signal encoding parameter 218
5.3.2.4.12.2.2 Adaptive downmix for stereo coding 218
5.3.2.4.12.2.3 Calculation of downmixed signal and residual signal during transitional frame 219
5.3.2.4.12.2.4 Adaptive downmix 220
5.3.2.4.13 Reverberation gain parameter determination 220
5.3.2.5 Stereo classifier 221
5.3.2.5.1 General 221
5.3.2.5.2 Classification of uncorrelated content 221
5.3.2.5.3 Cross-talk detection using logistic regression 224
5.3.2.5.4 Cross-talk detection based on the GCC-PHAT 231
5.3.2.5.5 Stereo mode selection 234
5.3.3 MDCT-based stereo 237
5.3.3.1 General Overview 237
5.3.3.2 ITD compensation 237
5.3.3.3 MDCT core pre-processing 238
5.3.3.3.1 General 238
5.3.3.3.2 TCX-LTP parameter estimation 238
5.3.3.3.3 Time-to-frequency transformations 238
5.3.3.3.4 Mid-high bitrate pre-processing 241
5.3.3.3.5 Temporal noise shaping (TNS) 247
5.3.3.3.6 Spectral Noise Shaping (SNS) 248
5.3.3.4 Stereo processing 257
5.3.3.4.1 General 257
5.3.3.4.2 Stereo spectral bands 257
5.3.3.4.3 Global ILD normalization 259
5.3.3.4.4 Conditional ILD correction 259
5.3.3.4.5 Band-wise M/S decision 260
5.3.3.4.6 M/S decision for IGF 261
5.3.3.4.7 M/S Transformation 262
5.3.3.4.8 Encoding of the stereo parameters 263
5.3.3.5 Power spectrum calculation and spectrum noise measure 263
5.3.3.6 Intelligent gap filling (IGF) 263
5.3.3.6.1 General overview 263
5.3.3.6.2 Stereo IGF encoding 264
5.3.3.7 Bitrate distribution 265
5.3.3.8 Quantization and encoding 265
5.3.3.8.1 Spectrum quantization and coding 265
5.3.3.8.2 Global gain estimator 266
5.3.3.8.3 Rate-loop for constant bit rate and global gain 266
5.3.3.8.4 Stereo noise level estimation 266

5.3.4 Switching between stereo modes 266
5.3.4.1 Overview 266
5.3.4.2 General 267
5.3.4.3 Memory handling 269
5.3.4.4 Switching between DFT and TD stereo modes 270
5.3.4.4.1 Switching from TD stereo to DFT stereo 270
5.3.4.4.2 Switching from DFT stereo to TD stereo 271
5.3.4.5 Switching between DFT and MDCT stereo modes 274
5.3.4.5.1 Switching from DFT stereo to MDCT stereo 274
5.3.4.5.2 Switching from MDCT stereo to DFT stereo 274
5.3.4.6 Switching between TD and MDCT stereo modes 274
5.3.4.6.1 Switching from TD stereo to MDCT stereo 274
5.3.4.6.2 Switching from MDCT stereo to TD stereo 275
5.3.5 DTX operation 275
5.3.5.1 DTX in Unified stereo 275
5.3.5.1.1 Signal activity detection in Unified stereo 275
5.3.5.1.2 General 276
5.3.5.1.3 Stereo CNG side gain encoding 277
5.3.5.1.4 Stereo CNG ITD and IPD encoding 278
5.3.5.1.5 Stereo CNG coherence encoding 278
5.3.5.2 DTX in MDCT-based stereo 281
5.3.5.2.1 Overview 281
5.3.5.2.2 VAD in MDCT-based Stereo 281
5.3.5.2.3 Coherence estimation 281
5.3.5.2.4 Noise parameter estimation and encoding 282
5.4 Scene-based audio (SBA) operation 283
5.4.1 SBA format overview 283
5.4.2 Combined DirAC and SPAR based SBA coding 283
5.4.2.1 Combined DirAC and SPAR based SBA coding overview 283
5.4.2.2 FOA signal coding at all bitrates and HOA2, HOA3 signal coding at bitrates below 256 kbps 285
5.4.2.2.1 Overview 285
5.4.2.2.2 Metadata parameter analysis 285
5.4.2.2.3 Downmixing and core-coding 286
5.4.2.3 HOA2 and HOA3 signal coding overview at 256 kbps 286
5.4.2.3.1 Overview 286
5.4.2.3.2 Metadata parameter analysis 286
5.4.2.3.3 Downmixing and core-coding 287
5.4.2.4 HOA2 and HOA3 signal coding overview at 384 kbps 287
5.4.2.4.1 Overview 287
5.4.2.4.2 Metadata parameter generation 287
5.4.2.4.3 Downmixing and core-coding 287
5.4.2.5 HOA2 and HOA3 signal coding overview at 512 kbps 288
5.4.2.5.1 Overview 288
5.4.2.5.2 Metadata parameter generation 288
5.4.2.5.3 Downmixing and core-coding 288
5.4.3 SBA parameter estimation 289
5.4.3.1 SBA front VAD 289
5.4.3.2 Transient Detection 289
5.4.3.3 Analysis windowing and MDFT 289
5.4.3.4 DirAC parameter estimation 291
5.4.3.4.1 General 291
5.4.3.4.2 High-order operation mode 291
5.4.3.4.3 Low-order operation mode 295
5.4.3.5 Hybrid encoder 296
5.4.3.6 Mono signal detection 296
5.4.3.6.1 Overview 296
5.4.3.6.2 Signalling of Mono through the bitstream 300
5.4.3.7 SPAR Metadata computation 301
5.4.3.7.1 Overview 301
5.4.3.7.2 Banded covariance computation 302
5.4.3.7.3 Active W downmix detector 303
5.4.3.7.4 Banded covariance re-mixing 304

5.4.3.7.5 SPAR parameter estimation....305
5.4.3.7.6 Passive-W channel prediction coefficients computation....305
5.4.3.7.7 Active-W channel prediction coefficients computation....305
5.4.3.7.8 Cross-Prediction coefficients computation....308
5.4.3.7.9 Decorrelation coefficients computation....309
5.4.3.7.10 Overview of quantization, bitrate distribution and coding of SPAR parameters....309
5.4.3.7.11 SPAR bitrate distribution table....310
5.4.3.7.12 Quantization and coding process....313
5.4.3.7.13 Spectral filling in the side channels at 13.2 kbps and 16.4 kbps....316
5.4.3.7.14 Time differential coding....316
5.4.3.7.15 Band interleaving and 40ms MD update rate at 13.2 kbps and 16.4 kbps....317
5.4.3.7.16 Entropy coding with Arithmetic coders....318
5.4.3.7.17 Base2 coding with Huffman coder....320
5.4.3.8 DirAC and SPAR parameter merge....321
5.4.3.8.1 General....321
5.4.3.8.2 DirAC to SPAR parameter conversion....321
5.4.3.8.3 Covariance estimation from DirAC parameters....321
5.4.3.8.4 Directional diffuseness....323
5.4.3.8.5 SPAR parameter estimation from DirAC parameters....323
5.4.3.8.6 Active W prediction from DirAC parameters....323
5.4.3.8.7 Passive W prediction from DirAC parameters....324
5.4.4 Downmix matrix calculation....324
5.4.4.1 SPAR parameters to downmix matrix conversion....324
5.4.4.2 Energy compensation in DirAC bands....325
5.4.4.3 Band Interleaving of downmix matrix at 13.2 and 16.4 kbps....325
5.4.5 Downmix generation....325
5.4.5.1 Overview....325
5.4.5.2 Time domain to MDFT conversion....325
5.4.5.3 Filterbank mixing....326
5.4.5.4 MDFT to Time domain conversion....326
5.4.5.5 Windowing and crossfading....326
5.4.5.6 Time domain mixing....327
5.4.6 Principal Component Analysis (PCA)....327
5.4.7 Automatic Gain Control (AGC)....333
5.4.8 Core-coder encoding....335
5.4.8.1 General....335
5.4.8.2 MCT bitrate distribution in SBA high bitrate mode....335
5.4.9 DTX operation....336
5.4.9.1 Overview....336
5.4.9.2 DirAC parameter estimation....336
5.4.9.3 SPAR parameter estimation....336
5.4.9.3.1 General....336
5.4.9.3.2 Banded covariance smoothing....337
5.4.9.3.3 Quantization and coding in VAD inactive frames....337
5.4.9.4 SPAR and DirAC parameter merge....337
5.4.9.5 CNG parameters encoding....338
5.4.10 SBA bitrate switching....338
5.5 Metadata-assisted spatial audio (MASA) operation....339
5.5.1 MASA format overview....339
5.5.2 MASA format metadata input....340
5.5.2.1 Obtaining the MASA format metadata....340
5.5.2.2 Direction index deindexing....340
5.5.2.3 TF tile based energy calculation....342
5.5.3 Coding of MASA format input data....342
5.5.3.1 Overview....342
5.5.3.2 MASA metadata pre-encoding configuration, processing, and signalling....342
5.5.3.2.1 Overview....342
5.5.3.2.2 Bitrate dependent parameter settings....343
5.5.3.2.3 Metadata composition detection, alignment, and check for validity....344
5.5.3.2.4 Common MASA metadata codec configuration....350
5.5.3.2.5 Energy ratio compensation....352
5.5.3.2.6 Merging of MASA spatial parameter metadata across MASA frequency bands and subframes....353

5.5.3.2.7 Combining of MASA spatial audio metadata across multiple directions 355
5.5.3.2.8 Metadata reductions for low rate 357
5.5.3.2.9 Reordering of directional data with two concurrent directions 358
5.5.3.2.10 Signalling bits 359
5.5.3.2.11 Adjustment of highest transmitted MASA metadata band 359
5.5.3.3 MASA metadata quantization and encoding 360
5.5.3.3.1 Overview 360
5.5.3.3.2 Energy ratio encoding 361
5.5.3.3.3 Direction encoding 361
5.5.3.3.4 Spread coherence encoding 362
5.5.3.3.5 Surround coherence encoding 362
5.5.3.3.6 Coding of the second direction parameters 362
5.5.4 Encoding of MASA audio transport channels 363
5.5.5 DTX operation 363
5.5.6 Bitstream structure 363
5.5.7 MASA bitrate switching 364
5.6 Object-based audio (ISM) operation 364
5.6.1 ISM format overview 364
5.6.2 Discrete ISM coding mode 365
5.6.2.1 Overview 365
5.6.2.2 DiscISM encoding system 365
5.6.2.3 Bitrates distribution between audio streams 366
5.6.2.3.1 Bitrate distribution algorithm 366
5.6.2.3.2 Bitrate adaptation based on ISM importance 367
5.6.3 Parametric ISM coding mode 368
5.6.3.1 General 368
5.6.3.2 Parameters 368
5.6.3.3 Noisy Speech 369
5.6.3.4 Downmix 370
5.6.3.5 Encoded Data 372
5.6.4 ISM metadata coding 372
5.6.4.1 General 372
5.6.4.2 Direction metadata encoding and quantization 373
5.6.4.2.1 General 373
5.6.4.2.2 Intra-object metadata coding logic 374
5.6.4.2.3 Inter-object metadata coding logic 374
5.6.4.3 Extended metadata encoding and quantization 375
5.6.4.4 Panning gain in non-diegetic rendering 375
5.6.5 Bitstream structure 375
5.6.5.1 Overview 375
5.6.5.2 ISM signalling 376
5.6.5.3 Coded metadata payload 376
5.6.5.4 Audio streams payload 376
5.6.6 DTX operation 376
5.6.6.1 Overview 376
5.6.6.2 ISM DTX operation overview 377
5.6.6.3 Classification of Inactive Frames in ISM Encoder 377
5.6.6.3.1 General 377
5.6.6.3.2 Global SID Counter 378
5.6.6.4 SID Metadata Analysis, Quantization and Coding 379
5.6.6.5 CNG parameters encoding 380
5.6.6.6 SID bitstream structure 381
5.6.6.6.1 Overview 381
5.6.6.6.2 ISM Common Signalling in SID Frame 382
5.6.6.6.3 SID Data Payload 382
5.6.6.6.4 SID Audio Streams Payload 382
5.6.7 ISM bitrate switching 383
5.7 Multi-channel audio (MC) operation 383
5.7.1 MC format overview 383
5.7.2 LFE channel encoding 383
5.7.2.1 Low-pass filtering 383
5.7.2.2 MDCT based LFE encoding 384

5.7.2.2.1 Overview 384
5.7.2.2.2 Windowing and MDCT 384
5.7.2.2.3 Quantization 385
5.7.2.2.4 Coding 386
5.7.3 Multi-channel MASA (McMASA) coding mode 387
5.7.3.1 McMASA coding mode overview 387
5.7.3.2 LFE energy computation 389
5.7.3.2.1 LFE energy computation in normal sub-mode 389
5.7.3.2.2 LFE energy computation in separate-channel sub-mode 390
5.7.3.3 McMASA spatial audio parameter estimation 390
5.7.3.4 Even loudspeaker setup determination 393
5.7.3.5 McMASA transport audio signal generation 394
5.7.3.6 McMASA metadata encoding 395
5.7.3.6.1 McMASA metadata pre-encoding configuration and processing 395
5.7.3.6.2 Spatial metadata encoding 395
5.7.3.6.3 LFE-to-total energy ratio encoding 396
5.7.3.7 McMASA transport audio signal encoding 397
5.7.3.8 Bitstream structure 398
5.7.4 Parametric MC coding mode 398
5.7.4.1 ParamMC coding mode overview 398
5.7.4.2 ParamMC parameter frame index initialization and update 399
5.7.4.3 ParamMC transport channel and parameter band configuration 399
5.7.4.4 ParamMC transport audio signal generation 400
5.7.4.5 ParamMC time domain transient detection 401
5.7.4.6 ParamMC analysis windowing and MDFT 401
5.7.4.7 ParamMC covariance estimation 402
5.7.4.8 ParamMC default settings for the LFE channel 403
5.7.4.9 ParamMC parameter based transient detection 403
5.7.4.10 ParamMC band combining in case of transients 403
5.7.4.11 ParamMC complex valued to real valued covariance conversion 403
5.7.4.12 ParamMC LFE activity detection 404
5.7.4.13 ParamMC parameter quantization 404
5.7.4.13.1 Parameter quantizers 404
5.7.4.13.2 Interchannel Level differences (ICLDs) 404
5.7.4.13.3 Inter-channel coherences (ICCs) 406
5.7.4.14 ParamMC parameter encoding 406
5.7.4.14.1 Common ParamMC parameter encoding 406
5.7.4.14.2 ICC and ICLD Parameter quantization indices encoding 407
5.7.4.15 ParamMC transport audio signal encoding 410
5.7.4.16 ParamMC bit rate switching 410
5.7.5 Parametric upmix MC coding mode 411
5.7.5.1 General 411
5.7.5.2 Sectioning and downmixing 411
5.7.5.3 Metadata parameter calculation 411
5.7.5.4 Quantization 413
5.7.5.5 Entropy Coding 414
5.7.6 Discrete MC coding mode 415
5.7.7 MC bitrate switching 415
5.8 Combined Object-based audio and SBA (OSBA) operation 415
5.8.1 OSBA format overview 415
5.8.2 Low-bitrate pre-rendering OSBA coding mode 415
5.8.3 High-bitrate discrete OSBA coding mode 416
5.8.4 OSBA bitrate switching 416
5.9 Combined Object-based audio and MASA (OMASA) operation 417
5.9.1 OMASA format overview 417
5.9.2 OMASA format configurations 417
5.9.3 OMASA pre-coding processing tools 418
5.9.3.1 OMASA spatial audio parameter analysis 418
5.9.3.2 OMASA MASA metadata combining 419
5.9.3.3 OMASA audio signals downmix 420
5.9.3.4 Determination of an object to be separated 421
5.9.3.5 Separation of an object from other objects 422

5.9.4 Low-bitrate pre-rendering (Rend OMASA) coding mode 423
5.9.4.1 Overview 423
5.9.4.2 Low-bitrate pre-rendering coding method 423
5.9.5 One object with MASA representation (One MASA) coding mode 423
5.9.5.1 Overview 423
5.9.5.2 One object with MASA representation coding method 424
5.9.6 Parametric one object (Param OMASA) coding mode 424
5.9.6.1 Overview 424
5.9.6.2 Determination of OMASA parametric information 425
5.9.6.3 One object with parametric representation coding method 425
5.9.6.3.1 Coding method overview 425
5.9.6.3.2 Encoding of MASA-to-total energy ratios 426
5.9.6.3.3 Encoding of ISM energy ratios 426
5.9.6.3.4 Encoding of ISM metadata 430
5.9.7 Discrete (Disc OMASA) coding mode 431
5.9.8 Inter-format bitrate adaptation 431
5.9.9 OMASA bitrate switching 434
5.9.10 OMASA bitstream structure 434
5.9.10 OMASA bitstream structure 434
5.10 EVS-compatible mono audio operation 434
5.11 Stereo downmix operation for EVS mono coding 434
5.11.1 Overview 434
5.11.2 Phase-only correlation (POC) mode 435
5.11.2.1 Overview 435
5.11.2.2 ITD analysis 435
5.11.2.2.1 Cross talk adding procedure in the frequency domain 435
5.11.2.2.2 Approximated phase spectrum 435
5.11.2.2.3 Calculation of phase only correlation 436
5.11.2.2.4 Calculation of ITD and weighting coefficients 436
5.11.2.3 Downmix process 438
5.11.3 Phase compensation (PHA) mode 439
5.11.3.1 Overview 439
5.11.4 Mode selection rule 443
5.11.5 Handling of mode switching 443

6 Functional description of the decoder 444
6.1 Decoder overview 444
6.2 Common processing and decoding tools 445
6.2.1 Common decoder processing overview 445
6.2.1.1 High-pass filtering 445
6.2.1.2 Limiter 445
6.2.1.2.1 Overview 445
6.2.1.2.2 Detailed algorithmic description 445
6.2.1.2.3 Handling of strong saturations 446
6.2.1.3 Time-domain decorrelator 448
6.2.2 Core-decoder processing 449
6.2.2.1 Overview 449
6.2.2.2 LP based decoding 450
6.2.2.2.1 Variable bitrate ACELP decoding 450
6.2.2.2.2 Fast algebraic codebook decoding 450
6.2.2.2.3 AVQ Bit Savings Decoder 450
6.2.2.2.4 Bass Post-Filter 454
6.2.2.2.5 GSC core decoding 454
6.2.2.2.6 Bandwidth extension in time domain 457
6.2.2.2.7 Multimode FD bandwidth extension coding 457
6.2.2.3 MDCT based decoding 457
6.2.2.3.1 General 457
6.2.2.3.2 Variable bitrate MDCT decoding 457
6.2.2.3.3 TCX entropy decoding 457
6.2.2.3.4 PLC method selection 459
6.2.2.3.5 Phase ECU enhancements 460
6.2.2.3.6 Long term prediction processing 461

6.2.2.4 Background noise estimation 463
6.2.2.4.1 General 463
6.2.2.4.2 Power spectrum compression 463
6.2.2.4.3 Compensation for the loss of variance 465
6.2.2.4.4 Spectral smoothing 465
6.2.2.4.5 Initialization period 466
6.2.2.4.6 Power spectrum expansion 468
6.2.2.5 Switching coding modes 468
6.2.2.5.1 Improved handling of envelope stability parameter in HQ MDCT 468
6.2.2.6 DTX/CNG operation 469
6.2.2.6.1 General 469
6.2.2.6.2 FD-CNG VQ global gain dequantization 469
6.2.2.6.3 First stage VQ decoding for FD-CNG 469
6.2.3 Common audio decoding tools 470
6.2.3.1 General 470
6.2.3.2 Single Channel Element (SCE) decoder 470
6.2.3.3 Channel Pair Element (CPE) decoder 471
6.2.3.4 Multichannel coding tool (MCT) decoder 473
6.2.3.4.1 General overview 473
6.2.3.4.2 Core and SNS parameter decoding 473
6.2.3.4.3 MCT parameter decoding 473
6.2.3.4.4 Bitrate distribution 473
6.2.3.4.5 Spectral data decoding and noise filling 474
6.2.3.4.6 Application of IGF and stereo IGF 474
6.2.3.4.7 MCT decoding 474
6.2.3.4.8 Noise shaping 474
6.2.3.4.9 Inverse MDCT transform to time domain 475
6.2.3.4.10 MCT PLC 475
6.2.4 Common spatial metadata decoding tools 475
6.2.4.1 Direction metadata decoding tools 475
6.2.4.1.1 Overview 475
6.2.4.1.2 Direction dequantization tools 475
6.2.4.1.3 Direction metadata raw decoding 476
6.2.4.1.4 Direction metadata entropy decoding tools 476
6.2.4.2 Diffuseness and energy ratio decoding methods 477
6.2.4.2.1 Diffuseness decoding 477
6.2.4.2.2 Diffuseness and energy ratio decoding with two concurrent directions 477
6.2.4.3 Direction metadata decoding methods 478
6.2.4.3.1 Direction metadata decoding overview 478
6.2.4.3.2 Decoding EC1 478
6.2.4.3.3 Decoding EC2 479
6.2.4.3.4 Decoding EC3 479
6.2.4.4 Coherence decoding 481
6.2.4.4.1 Spread coherence decoding 481
6.2.4.4.2 Surround coherence decoding 482
6.2.4.5 DTX decoding 482
6.2.5 Common filter bank operation tools 483
6.2.5.1 Complex Low-delay Filter Bank (CLDFB) analysis 483
6.2.5.2 Complex Low-delay Filter Bank (CLDFB) synthesis 483
6.2.6 CLDFB domain decorrelator 484
6.2.6.1 Overview 484
6.2.6.2 Onset detection and removal 484
6.2.6.3 De-correlation 485
6.2.6.3.1 Overview 485
6.2.6.3.2 Derivation of filter coefficients from lattice coefficients 485
6.2.6.3.3 Decorrelator IIR Filtering 488
6.2.6.4 Energy adjustment 489
6.2.6.5 Final output 489
6.2.7 Common tools for decoding with time scale modification (JBM) 490
6.2.7.1 Overview 490
6.2.7.2 Transport channel buffer management 490
6.2.7.3 Time scale modification 491

6.2.7.3.1 Extension to multi-channel processing 491
6.2.7.3.2 Energy estimation 491
6.2.7.4 Metadata and processing parameter adaption 492
6.2.7.4.1 Overview 492
6.2.7.4.2 Metadata adaption 492
6.2.7.4.3 Processing parameter adaption 494
6.2.7.5 Bitrate switching 496
6.2.7.5.1 Renderer flushing 496
6.2.7.5.2 Transport channel reconfiguration 496
6.2.7.5.3 Renderer discard samples 496
6.3 Stereo audio decoding 497
6.3.1 Common stereo decoding tools 497
6.3.1.1 Common Stereo postprocessing 497
6.3.1.2 ITD compensation 497
6.3.1.2.1 Overview 497
6.3.1.2.2 Inter-channel Alignment (ICA) decoder 497
6.3.2 Unified stereo decoding 498
6.3.2.1 Inter-channel BWE (IC-BWE) decoder 498
6.3.2.1.1 Inter-Channel BWE Decoder overview 498
6.3.2.1.2 Inter-Channel BWE Decoder Block 499
6.3.2.2 TD-based stereo decoding 500
6.3.2.2.1 TD stereo decoder overview 500
6.3.2.2.2 Reusing close-loop pitch and adaptive cobebook of the primary channel 501
6.3.2.2.3 Reusing LP filter coefficients of the primary channel and dequantization of LSF parameters for the secondary channel 502
6.3.2.2.4 Estimation of spatial parameters of the background noise 503
6.3.2.3 DFT-based stereo decoding 503
6.3.2.3.1 General 503
6.3.2.3.2 STFT analysis 503
6.3.2.3.3 Stereo parameter decoding 504
6.3.2.3.4 Residual signal decoding 505
6.3.2.3.5 Bass post filter application 506
6.3.2.3.6 Residual predicted signal generation 508
6.3.2.3.7 Stereo upmix 515
6.3.2.3.8 Stereo comfort noise injection 515
6.3.2.3.9 STFT synthesis 517
6.3.2.3.10 DFT-based stereo PLC 518
6.3.3 MDCT-based stereo decoding 521
6.3.3.1 General Overview 521
6.3.3.2 Decoding of parameters 521
6.3.3.2.1 TCX block configuration 521
6.3.3.2.2 Core parameters decoding 521
6.3.3.2.3 SNS parameters decoding 521
6.3.3.2.4 Stereo parameters decoding 523
6.3.3.3 Decoding process 524
6.3.3.3.1 Spectral data decoding and noise filling 524
6.3.3.3.2 Application of stereo IGF 524
6.3.3.4 Stereo decoding process 525
6.3.3.4.1 Inverse mid-side transform 525
6.3.3.4.2 De-normalization to global ILD 527
6.3.3.5 Post-stereo decoding process 527
6.3.3.5.1 General 527
6.3.3.5.2 Spectral noise shaping 527
6.3.3.5.3 Temporal noise shaping 528
6.3.3.6 Frequency-to-time domain transformations 528
6.3.3.6.1 General 528
6.3.3.6.2 Kernel switching 528
6.3.3.6.3 Transition between long and short blocks with double overlap 528
6.3.3.7 PLC in MDCT-based stereo 530
6.3.3.7.1 Frequency-domain concealment 530
6.3.3.7.2 Time-domain concealment 534
6.3.3.7.3 Fading to zero output 535

6.3.4 Switching between stereo modes 535
6.3.4.1 Overview 535
6.3.4.2 General 535
6.3.4.3 Memory handling 538
6.3.4.4 Switching between DFT and TD stereo modes 539
6.3.4.4.1 Switching from TD stereo to DFT stereo 539
6.3.4.4.2 Switching from DFT stereo to TD stereo 542
6.3.4.5 Switching between DFT and MDCT stereo modes 545
6.3.4.5.1 Switching from DFT stereo to MDCT stereo 545
6.3.4.5.2 Switching from MDCT stereo to DFT stereo 545
6.3.4.6 Switching between TD and MDCT stereo modes 545
6.3.4.6.1 Switching from TD stereo to MDCT stereo 545
6.3.4.6.2 Switching from MDCT stereo to TD stereo 546
6.3.5 DTX operation 546
6.3.5.1 DTX in Unified stereo 546
6.3.5.1.1 General 546
6.3.5.1.2 Stereo CNG spectral shape extraction 546
6.3.5.1.3 Stereo CNG side gain, ITD and IPD decoding 549
6.3.5.1.4 Stereo CNG coherence decoding 550
6.3.5.1.5 Stereo CNG synthesis 550
6.3.5.2 DTX in MDCT-based stereo 551
6.3.5.2.1 General overview 551
6.3.5.2.2 Decoding of parametric noise data 551
6.3.5.2.3 Stereo CNG 551
6.3.5.2.4 Smoothing of transitions from DTX frame to active TCX 552
6.3.6 Stereo output format conversion 553
6.3.6.1 Overview 553
6.3.6.2 Stereo-to-Mono output 553
6.3.6.2.1 DFT stereo mono output 553
6.3.6.2.2 TD stereo mono output 554
6.3.6.2.3 MDCT stereo mono output 554
6.3.6.3 Stereo-to-MC output 558
6.3.7 Stereo audio decoding with TSM 558
6.4 Scene-based audio (SBA) decoding 558
6.4.1 Combined DirAC and SPAR based SBA decoding overview 558
6.4.1.1 Overview 558
6.4.1.2 FOA signal decoding at all bitrates and HOA2, HOA3 signal decoding at bitrates below 256 kbps 559
6.4.1.2.1 Overview 559
6.4.1.2.2 Metadata decoding 560
6.4.1.2.3 Core-coder decoding 560
6.4.1.2.4 SBA upmix and rendering 560
6.4.1.3 HOA2 and HOA3 signal decoding at bitrate 256 kbps 561
6.4.1.3.1 Overview 561
6.4.1.3.2 Metadata decoding 561
6.4.1.3.3 Core-coder decoding 561
6.4.1.3.4 SBA upmix and rendering 562
6.4.1.4 HOA2 and HOA3 signal decoding at bitrates 384 kbps 562
6.4.1.4.1 Overview 562
6.4.1.4.2 Metadata decoding 563
6.4.1.4.3 Core-coder decoding 563
6.4.1.4.4 SBA upmix and rendering 563
6.4.1.5 HOA2 and HOA3 signal coding at bitrates 512 kbps 564
6.4.1.5.1 Overview 564
6.4.1.5.2 Metadata decoding 564
6.4.1.5.3 Core-coder decoding 564
6.4.1.5.4 SBA upmix and rendering 564
6.4.2 DirAC parameter decoding 565
6.4.3 SPAR parameter decoding 565
6.4.3.1 General 565
6.4.3.2 Set active W prediction flag 565
6.4.3.3 Decode quantization and coding strategy bits 565

6.4.3.4 Time differential decoding 566
6.4.3.4.1 General 566
6.4.3.4.2 Time differential decoding across same quantization strategies 566
6.4.3.4.3 Time differential decoding across different quantization strategies 567
6.4.3.5 Band interleaved decoding at 13.2 kbps and 16.4 kbps 567
6.4.3.6 Arithmetic decoder 567
6.4.3.7 Huffman decoder 567
6.4.4 SPAR and DirAC parameter merge 567
6.4.4.1 SBA mono handling 567
6.4.4.2 SPAR to DirAC parameter conversion 568
6.4.4.3 DirAC model for the SPAR parameters 569
6.4.4.3.1 Overview 569
6.4.4.3.2 DirAC to SPAR conversion for parameters corresponding to downmix channels 569
6.4.4.3.3 DirAC to SPAR conversion for PR, CP and D coefficients corresponding to parametric channels 569
6.4.5 Upmix matrix calculation 570
6.4.5.1 Overview 570
6.4.5.2 SPAR matrix generation (P and C matrices) 570
6.4.5.3 Upmix matrix generation 571
6.4.6 Decoded audio processing 571
6.4.6.1 Downmix signal decoding (core-decoding) 571
6.4.6.2 AGC 571
6.4.6.3 PCA 572
6.4.6.4 SPAR upmix processing 574
6.4.6.4.1 General 574
6.4.6.4.2 Time domain decorrelator 574
6.4.6.4.3 CLDFB analysis 574
6.4.6.4.4 CLDFB upmixing 574
6.4.6.4.5 MDFT filterbank banded upmix matrix to CLDFB banded upmix matrix conversion 576
6.4.6.4.6 Crossfading in CLDFB 577
6.4.6.4.7 Additional VLBR smoothing 577
6.4.6.5 DirAC synthesis and rendering 577
6.4.6.5.1 Overview 577
6.4.6.5.2 Mode selection 578
6.4.6.5.3 High-order operation mode 578
6.4.6.5.4 Low-order operation mode 581
6.4.6.5.5 Bitrate switching 584
6.4.6.5.6 Binaural rendering 584
6.4.6.5.7 Loudspeaker rendering 585
6.4.6.5.8 Stereo and mono output 587
6.4.6.6 CLDFB synthesis 591
6.4.7 DTX operation 592
6.4.7.1 Overview 592
6.4.7.2 DTX in DirAC 592
6.4.7.3 DTX in SPAR 592
6.4.8 SBA PLC 593
6.4.8.1 PLC in DirAC 593
6.4.8.2 PLC in SPAR 593
6.4.8.2.1 SPAR processing of erroneous frames 593
6.4.8.2.2 Recovery after one or more lost frames 594
6.4.8.2.3 SPAR parameter recovery for 24.4 kbps and above 594
6.4.8.2.4 SPAR parameter recovery for 13.2 kbps and 16.4 kbps 594
6.4.8.2.5 Interpolation of missing SPAR parameters from current existing SPAR parameters 594
6.4.9 SBA bitrate switching 595
6.4.10 SBA output format conversion 595
6.4.11 SBA decoding with TSM 596
6.5 Metadata-assisted spatial audio (MASA) decoding 596
6.5.1 Overview 596
6.5.2 Decoding MASA configuration 597
6.5.3 MASA metadata decoding 598
6.5.3.1 Overview 598
6.5.3.2 Energy ratio decoding 599

6.5.3.2.1 Energy ratio decoding for bitrates up to 256 kbps 599
6.5.3.2.2 Energy ratio decoding for 384 kbps and 512 kbps 599
6.5.3.3 Direction decoding 599
6.5.3.3.1 Direction decoding for bitrates up to 256 kbps 599
6.5.3.3.2 Direction decoding for 384 kbps and 512 kbps 599
6.5.3.4 Spread coherence decoding 600
6.5.3.4.1 Spread coherence decoding for bitrates up to 256 kbps 600
6.5.3.4.2 Spread coherence decoding for 384 kbps and 512 kbps 600
6.5.3.5 Surround coherence decoding 600
6.5.3.5.1 Surround coherence decoding for bitrates up to 256 kbps 600
6.5.3.5.2 Surround coherence decoding for 384 kbps and 512 kbps 600
6.5.3.6 Decoding of the second direction parameters 600
6.5.3.6.1 Joint decoding of parameters associated with first and second sound source directions 600
6.5.3.6.2 Independent decoding of parameters associated with first and second sound source directions 600
6.5.3.7 Reconstruction of MASA metadata from low-rate reduction 600
6.5.3.8 Determining encoding metric based on the quality of the spatial metadata representation in encoding 601
6.5.3.9 Reconstruction of full MASA format metadata 602
6.5.4 MASA DTX operation 603
6.5.5 MASA PLC 603
6.5.6 MASA bitrate switching 603
6.5.7 MASA rendering 603
6.5.7.1 Binaural and stereo rendering 603
6.5.7.2 Multi-channel loudspeaker and Ambisonics rendering 603
6.5.7.2.1 Overview 603
6.5.7.2.2 Determining direct and diffuse power factors and surround coherence ratio 604
6.5.7.2.3 Determining directional responses 604
6.5.7.2.4 Determining diffuse responses 608
6.5.7.2.5 Determining prototype audio signals for multi-channel loudspeaker output 608
6.5.7.2.6 Determining prototype audio signals for Ambisonics output 609
6.5.7.2.7 Decorrelation 612
6.5.7.2.8 Spatial synthesis 612
6.5.7.3 Mono rendering 615
6.5.7.4 Output for external processing 615
6.5.8 MASA decoding with TSM 615
6.6 Object-based audio (ISM) decoding 615
6.6.1 Discrete ISM decoding mode 615
6.6.2 Parametric ISM decoding mode 616
6.6.2.1 General 616
6.6.2.2 Parameters 617
6.6.2.3 Downmix 617
6.6.2.4 ParamISM Decoding and Rendering 618
6.6.2.4.1 Output Formats 618
6.6.2.4.2 Mono, stereo, and binaural output 618
6.6.2.4.3 Loudspeaker and Ambisonics output 618
6.6.2.4.4 EXT output 621
6.6.3 ISM metadata decoding 622
6.6.4 DTX operation decoding 622
6.6.4.1 Overview 622
6.6.4.2 CNG control parameter decoding 622
6.6.4.3 Metadata Dequantization and Decoding 623
6.6.4.4 ISM DTX core decoding and CNG 623
6.6.5 ISM PLC 625
6.6.6 ISM bitrate switching 625
6.6.6.1 Metadata handling in bitrate switching 625
6.6.6.2 Codec reconfiguration in bitrate switching 626
6.6.7 ISM output format conversion 626
6.6.7.1 Overview 626
6.6.7.2 Mono downmix 627
6.6.8 ISM decoding with TSM 628
6.7 Multi-channel audio (MC) decoding 628
6.7.1 LFE channel decoding 628

6.7.1.1 Overview 628
6.7.1.2 Bitstream decoding 628
6.7.1.3 MDCT coefficient reconstruction 629
6.7.1.4 Inverse MDCT 629
6.7.1.5 Windowing and overlap-add 630
6.7.1.6 Output LPF and delay adjustment 630
6.7.1.7 LFE PLC 630
6.7.1.7.1 Overview 630
6.7.1.7.2 Burst loss determination 630
6.7.1.7.3 LFE substitution frame processing 631
6.7.2 Multi-channel MASA (McMASA) decoding mode 633
6.7.2.1 McMASA decoding mode overview 633
6.7.2.2 McMASA metadata decoding 633
6.7.2.2.1 Decoding McMASA configuration 633
6.7.2.2.2 Spatial metadata decoding 633
6.7.2.2.3 LFE-to-total energy ratio decoding 633
6.7.2.3 McMASA transport audio signal decoding 634
6.7.2.4 McMASA rendering 634
6.7.2.4.1 Binaural rendering 634
6.7.2.4.2 Multi-channel loudspeaker rendering 634
6.7.2.4.3 Ambisonics rendering 639
6.7.2.4.4 Stereo rendering 639
6.7.2.4.5 Mono rendering 639
6.7.2.5 McMASA PLC 639
6.7.2.6 McMASA decoding with TSM 639
6.7.3 Parametric MC decoding mode 640
6.7.3.1 ParamMC Overview 640
6.7.3.2 ParamMC configuration 640
6.7.3.3 ParamMC Parameter Decoding 640
6.7.3.3.1 Common ParamMC parameter decoding 640
6.7.3.3.2 ICC and ICLD Parameter decoding 641
6.7.3.4 ParamMC Transport Audio Signal Decoding 643
6.7.3.5 ParamMC Synthesis 643
6.7.3.5.1 ParamMC Synthesis Overview 643
6.7.3.5.2 ParamMC Target Covariance Calculation 643
6.7.3.5.3 ParamMC Mixing Matrix Calculation 645
6.7.3.5.4 Decorrelation 648
6.7.3.5.5 Upmix 648
6.7.3.6 ParamMC output processing 649
6.7.3.6.1 ParamMC output processing selection 649
6.7.3.6.2 ParamMC mono/stereo output processing 649
6.7.3.6.3 ParamMC loudspeaker output processing 650
6.7.3.6.4 ParamMC Ambisonics output processing 651
6.7.3.6.5 ParamMC binaural output processing 651
6.7.3.7 ParamMC PLC 652
6.7.3.8 ParamMC bit rate switching 652
6.7.3.9 ParamMC decoding with TSM 653
6.7.3.9.1 Decoding to mono/stereo 653
6.7.3.9.2 Decoding to all other output formats 653
6.7.4 Parametric upmix MC decoding mode 653
6.7.4.1 General 653
6.7.4.2 Special case of Binaural, Stereo and Mono rendering configurations 654
6.7.4.2.1 Binaural rendering configuration 654
6.7.4.2.2 Stereo and Mono rendering configuration 654
6.7.4.3 Metadata Huffman Decoding 654
6.7.4.4 Dequantization 654
6.7.4.5 Decorrelation 654
6.7.4.6 Upmixing operation 655
6.7.4.7 Output 655
6.7.4.8 Param Upmix PLC 655
6.7.4.9 Param Upmix MC decoding with TSM 656
6.7.5 Discrete MC decoding mode 656

6.7.5.1 Overview 656
6.7.5.2 Discrete MC PLC 656
6.7.6 MC bitrate switching 656
6.7.7 MC output format conversion 657
6.7.7.1 Overview 657
6.7.7.2 Common processing tools 657
6.7.7.2.1 Conversion of transport signals to TCX20 resolution 657
6.7.7.2.2 Computation of Equalization gain 657
6.7.7.3 Equalization of TCX Spectra 657
6.7.7.4 Rendering to a supported IVAS format 658
6.7.7.4.1 General 658
6.7.7.4.2 Mono and Stereo output 658
6.7.7.4.3 Remaining formats 658
6.7.7.5 Rendering to a custom loudspeaker layout 663
6.7.7.6 Rendering to binaural output with head tracking 663
6.7.8 MC decoding with TSM 664
6.8 Combined Object-based audio and SBA (OSBA) decoding 664
6.8.1 Low-bitrate pre-rendering OSBA decoding mode 664
6.8.2 High-bitrate discrete OSBA decoding mode 664
6.8.3 OSBA PLC 665
6.8.4 OSBA bitrate switching 665
6.8.5 OSBA output format conversion 665
6.8.6 OSBA decoding with TSM 665
6.9 Combined Object-based audio and MASA (OMASA) decoding 667
6.9.1 OMASA format decoder overview 667
6.9.2 Low-bitrate pre-rendering (Rend OMASA) decoding mode 668
6.9.3 One object with MASA (One OMASA) decoding mode 668
6.9.4 Parametric one object (Param OMASA) decoding mode 668
6.9.4.1 Overview 668
6.9.4.2 MASA-to-total ratios decoding 669
6.9.4.3 ISM energy ratios decoding 670
6.9.4.4 ISM metadata decoding 671
6.9.5 Discrete (Disc OMASA) decoding mode 672
6.9.6 OMASA rendering metadata generation 672
6.9.7 OMASA rendering 672
6.9.7.1 Binaural rendering 672
6.9.7.1.1 Discrete coding mode 672
6.9.7.1.2 Other coding modes 673
6.9.7.2 Stereo rendering 673
6.9.7.3 Multi-channel loudspeaker and Ambisonics rendering 673
6.9.7.3.1 Overview 673
6.9.7.3.2 Rendering of the MASA part 674
6.9.7.4 Mono rendering 674
6.9.8 OMASA PLC 674
6.9.9 OMASA bitrate switching 675
6.9.10 OMASA decoding with Time Scale Modification (TSM) 675
6.9.11 OMASA decoding to original combined input format 675
6.9.11.1 Overview 675
6.9.11.2 Decoding to original combined input format in pre-rendering mode 675
6.9.11.3 Decoding to original combined input format in one object with MASA representation mode 675
6.9.11.4 Decoding to original combined input format in parametric one object mode 677
6.9.11.5 Decoding to original combined input format in discrete mode 679

7 Functional description of the rendering, rendering control, and pre-rendering 679
7.1 Rendering overview 679
7.2 Rendering modes 680
7.2.1 Rendering for loudspeaker reproduction 680
7.2.1.1 Overview 680
7.2.1.2 Vector-base amplitude panning (VBAP) 680
7.2.1.2.1 VBAP initialization 680
7.2.1.2.2 VBAP gain determination 684
7.2.1.3 Edge Fading Amplitude Panning (EFAP) 686

7.2.1.3.1 EFAP overview 686
7.2.1.3.2 EFAP initialization 686
7.2.1.3.3 EFAP gain computation 688
7.2.1.4 All-round ambisonic panning and decoding 689
7.2.1.4.1 Overview 689
7.2.1.4.2 Mono and Stereo rendering 689
7.2.1.4.3 Rendering to other loudspeaker outputs 689
7.2.2 Rendering for binaural headphone reproduction 690
7.2.2.1 Binaural rendering overview 690
7.2.2.2 Time Domain binaural renderer 690
7.2.2.2.1 General 690
7.2.2.2.2 HRIR model 691
7.2.2.2.3 Obtain standard spline sample values 695
7.2.2.2.4 Obtain periodic spline sample values 696
7.2.2.2.5 ITD model 697
7.2.2.2.6 ITD synthesis 698
7.2.2.2.7 Distance and Direction Gain 700
7.2.2.2.8 HRIR convolution 702
7.2.2.3 Parametric binauralizer and parametric stereo renderer 703
7.2.2.3.1 Overview 703
7.2.2.3.2 Pre-processing the transport audio signals based on head orientation 704
7.2.2.3.3 Determination of input and target covariance matrices 707
7.2.2.3.4 Determining processing matrices based on input and target covariance matrices 711
7.2.2.3.5 Processing audio signals with the processing matrices 712
7.2.2.3.6 Determining direct part gains 713
7.2.2.3.7 Determining regularization factor 715
7.2.2.3.8 Decorrelation in the parametric binauralizer 716
7.2.2.3.9 SPAR metadata to MASA metadata mapping 716
7.2.2.3.10 Prototype signal generation with SBA format input 718
7.2.2.4 Fast convolution binaural renderer 719
7.2.2.4.1 CLDFB-domain convolution 719
7.2.2.4.2 Filter conversion 719
7.2.2.4.3 Late reverb model 720
7.2.2.4.4 Head tracking 724
7.2.2.5 Crend binaural renderer 724
7.2.2.5.1 Terms and Definitions 724
7.2.2.5.2 General 724
7.2.2.5.3 Convolver 724
7.3 Room acoustics rendering 727
7.3.1 Introduction to room acoustics rendering 727
7.3.2 Room impulse response convolution 727
7.3.3 Sparse frequency-domain reverberator 727
7.3.4 Feedback-delay network reverberator 728
7.3.4.1 Overview 728
7.3.4.2 Reverberator configuration 728
7.3.4.3 Processing 732
7.3.5 Early-reflection synthesis 732
7.3.5.1 General 732
7.3.5.2 Coordinate System 733
7.3.5.3 Source-receiver location correction 733
7.3.5.4 Reflections calculations 734
7.3.5.5 Process Loop 734
7.3.5.6 Low-complexity mode 736
7.4 Rendering control 736
7.4.1 Rendering control overview 736
7.4.2 Scene and listener orientation 737
7.4.2.1 Scene orientation 737
7.4.2.2 Listener orientation 737
7.4.3 Head tracking 737
7.4.3.1 Head tracking via scene displacement 737
7.4.3.2 Conversion from Euler angles to quaternions 738
7.4.3.3 Rotation matrix from quaternions 738

7.4.3.4 Application of rotations 738
7.4.3.4.1 Rotation in the spatial domain 738
7.4.3.4.2 Rotation in the spherical harmonic domain 739
7.4.4 Orientation tracking 739
7.4.4.1 Orientation tracking introduction 739
7.4.4.2 External reference orientation 740
7.4.4.3 External reference vector orientation 740
7.4.4.4 External reference levelled vector orientation 741
7.4.4.5 Adaptive long-term average reference orientation 741
7.4.5 External orientation input handling 743
7.4.5.1 Overview 743
7.4.5.2 Processing of the external orientation data 745
7.4.6 Combined rotations for rendering 746
7.4.6.1 Combining head and external rotations 746
7.4.6.2 External rotation interpolation 747
7.4.6.3 Initial values for combined rotation variables 749
7.4.7 HRTF and BRIR sets 749
7.4.7.1 HRTF and BRIR latency 749
7.4.7.2 Parametrization of Binaural renderers using binary file 750
7.4.7.3 HRTFs and BRIR conversion methods 751
7.4.7.3.1 Conversion from spatial domain to spherical harmonics domain 751
7.4.7.3.2 Conversion from Time domain to CLDFB domain for HRIRs (Fast convolution binaural renderer) 753
7.4.7.3.3 Conversion from Time domain to CLDFB domain for BRIRs (Fast convolution binaural renderer) 754
7.4.7.3.4 Conversion from Time domain to HRIR/ITD model (Time Domain binaural renderer) 754
7.4.7.3.5 Conversion from HRTF/BRIR to Crend binaural renderer convolver parameters 760
7.4.7.3.6 Conversion from Time domain to SH CLDFB domain for parametric binauralizer 763
7.4.8 Room acoustics parameters 766
7.4.8.1 Overview 766
7.4.8.2 Late reverb parameters 767
7.4.8.3 Early reflections parameters 768
7.4.9 Use of custom loudspeaker layouts 768
7.4.9.1 General 768
7.4.9.2 MASA, OMASA, and McMASA 769
7.4.9.3 ParamMC and Parametric Upmix MC 769
7.4.9.4 Remaining formats 769
7.5 Pre-rendering 769
7.5.2 Pre-rendering into SBA format 770
7.5.2.1 ISM to SBA rendering 770
7.5.2.2 Channel-based audio to SBA rendering 770
7.5.2.3 MASA to SBA rendering 770
7.5.3 Pre-rendering into MASA format 770
7.5.3.1 Overview 770
7.5.3.2 ISM to MASA rendering 770
7.5.3.3 Multi-channel to MASA rendering 770
7.5.3.4 SBA to MASA rendering 771
7.5.3.5 Merging audio signals rendered into MASA format 771
7.5.3.5.1 MASA input to MASA output rendering 771
7.5.3.5.2 MASA signal merge 771
7.5.3.5.3 Merging MASA metadata from ISM format input with inputs of other formats 772
7.5.3.5.4 Merging MASA metadata from other input formats 772
7.5.4 Pre-rendering into Binaural format 772
7.5.4.1 SBA and MC to Binaural rendering 772
7.5.4.2 ISM to Binaural rendering 773
7.5.4.3 MASA to Binaural rendering 773
7.5.5 Pre-rendering into MC format 773
7.5.5.1 ISM to MC rendering 773
7.5.5.2 SBA to MC rendering 773
7.5.5.3 MASA to MC rendering 773
7.5.5.4 MC to MC rendering 773
7.6 Split rendering 773

7.6.1 Overview 773
7.6.2 Split pre-rendering 774
7.6.2.1 Overview 774
7.6.2.2 Supported Split Rendering bitrates with LCLD or LC3plus codec 776
7.6.2.3 Supported Split Rendering bitrates with PCM output 776
7.6.2.4 Split pre-rendering of SBA 776
7.6.2.5 Split pre-rendering of MASA 777
7.6.2.6 Split pre-rendering of CBA 777
7.6.2.7 Split pre-rendering of ISM 777
7.6.2.8 Split pre-rendering of OSBA 777
7.6.2.9 Split pre-rendering of OMASA 777
7.6.3 Intermediate split renderer metadata format 777
7.6.3.1 Overview 777
7.6.3.2 Metadata computation, quantization and coding 778
7.6.3.2.1 Metadata computation for deviations about Yaw axis 778
7.6.3.2.2 Quantization and coding of Yaw metadata 779
7.6.3.2.3 Metadata computation for deviations about Pitch axis 780
7.6.3.2.4 Quantization and coding of Pitch metadata 780
7.6.3.2.5 Metadata computation for deviations about Roll axis 781
7.6.3.2.6 Quantization and coding of Roll metadata 781
7.6.3.3 Common split rendering metadata quantization and coding strategies 781
7.6.3.4 Intermediate split renderer metadata decoder 782
7.6.3.5 Intermediate split renderer metadata loss concealment 783
7.6.4 LCLD coded intermediate split renderer binaural audio format 783
7.6.4.1 LCLD codec overview 783
7.6.4.2 LCLD encoder 783
7.6.4.2.1 Overview 783
7.6.4.2.2 Perceptual Banding 784
7.6.4.2.3 Joint Channel Coding 785
7.6.4.2.3.1 Overview 785
7.6.4.2.3.2 Bitstream Syntax 786
7.6.4.2.3.3 Parameter Computation and Quantization 787
7.6.4.2.3.4 Joint Coding Type Decision 788
7.6.4.2.3.5 Entropy Coding of Parameters 789
7.6.4.2.4 Temporal Grouping 789
7.6.4.2.5 RMS Envelope 794
7.6.4.2.5.1 RMS Envelope Calculation 794
7.6.4.2.5.2 Normalizing the CLDFB Coefficients with the RMS Envelope 794
7.6.4.2.5.3 RMS Envelope Transmission 794
7.6.4.2.6 Perceptual Model 795
7.6.4.2.7 Linear Prediction 804
7.6.4.2.7.1 Overview 804
7.6.4.2.7.2 Bitstream Syntax 804
7.6.4.2.7.3 Prediction for 20ms frames 804
7.6.4.2.7.4 Prediction for frames shorter than 20ms 805
7.6.4.2.7.5 Prediction Signalling 805
7.6.4.2.7.6 Quantization of Prediction Parameters 806
7.6.4.2.7.7 Estimation of Prediction Parameters 807
7.6.4.2.8 Bit Allocation 808
7.6.4.2.9 Quantization of the Normalized CLDFB Coefficients 808
7.6.4.2.9.1 Overview 808
7.6.4.2.9.2 Differential Coding of the Normalized CLDFB Coefficients 809
7.6.4.2.9.3 Quantization of Normalized CLDFB coefficients and Prediction Residuals 809
7.6.4.2.9.4 Huffman Coding of Quantized Normalized CLDFB coefficients and Quantized Prediction Residuals 810
7.6.4.3 LCLD decoder 812
7.6.4.3.1 Overview 812
7.6.4.3.2 Decoding Group Information 812
7.6.4.3.3 Decoding RMS Envelope Information 812
7.6.4.3.4 Perceptual Model 813
7.6.4.3.5 Bit Allocation 813

7.6.4.3.6 Normalized CLDFB Coefficient and Prediction Residual Huffman Decoding and Inverse Quantization 813
7.6.4.3.7 Inverse Prediction 814
7.6.4.3.7.1 Overview 814
7.6.4.3.7.2 Status Tracking after Frame Loss 814
7.6.4.3.8 Inverse RMS Envelope Normalization 815
7.6.4.3.9 Inverse Joint Stereo Processing 816
7.6.4.4 LCLD packet loss concealment 816
7.6.4.4.1 General 816
7.6.4.4.2 Synthesis model 816
7.6.4.4.3 Analysis and parameter estimation 817
7.6.4.4.4 Tonality determination 818
7.6.4.4.5 Sinusoidal extension 818
7.6.4.4.6 Predictive extension 818
7.6.4.4.7 Cross-fade 818
7.6.4.4.8 Burst-loss handling 819
7.6.5 LC3plus coded intermediate split renderer binaural audio format 820
7.6.5.1 Introduction (Informative) 820
7.6.5.2 Overview 820
7.6.5.3 Encoder 820
7.6.5.4 Decoder 820
7.6.5.5 Frame Structure 820
7.6.5.6 Packet Loss Concealment 820
7.6.5.7 LC3 interoperable mode 820
7.6.6 Split post-rendering 821
7.6.6.1 Overview 821
7.6.6.2 Post rendering with pose correction 821
7.6.6.2.1 Metadata decoding 821
7.6.6.2.2 Metadata interpolation or extrapolation 822
7.6.6.2.3 Matrix mixing 822
7.6.6.3 Post rendering in 0 DOF mode 822
7.6.7 Bit allocation for Split rendering 822
7.6.8 Interface for Split rendering 826

8 Description of the transmitted parameter indices 826
8.1 Bit allocation overview 826
8.2 Bit allocation for stereo audio 827
8.2.1 Bit allocation for stereo in active frames 827
8.2.2 Bit allocation for stereo in SID frames 827
8.3 Bit allocation for scene-based audio (SBA) 827
8.3.1 Bit allocation for SBA in active frames 827
8.3.2 Bit allocation for SBA in SID frames 829
8.4 Bit allocation for metadata-assisted spatial audio (MASA) 830
8.4.1 Bit allocation for MASA in active frames 830
8.4.2 Bit allocation for MASA in SID frames 831
8.4.2 Bit allocation for MASA in SID frames 832
8.5 Bit allocation for object-based audio (ISM) 832
8.5.1 Bit allocation for ISM in active frames 832
8.5.2 Bit allocation for ISM in SID frames 832
8.6 Bit allocation for multi-channel audio (MC) 832
8.7 Bit allocation for combined Object-based audio and SBA (OSBA) 832
8.8 Bit allocation for combined Object-based audio and MASA (OMASA) 833
8.9 Bit allocation for EVS-compatible mono audio 836

Annex A (normative): RTP Payload Format and SDP Parameters 837

A.1 Introduction 837

A.2 Conventions, Definitions and Acronyms 837
A.2.1 Byte Order 837
A.2.2 List of Acronyms 837

A.3 Payload Format 837

A.3.1 Format Overview....837
A.3.2 RTP Header Usage....838
A.3.3 Packet Payload Structure....838
A.3.3.1 General....838
A.3.3.2 Format Description....838
A.3.3.3 Payload Header....839
A.3.3.3.1 General....839
A.3.3.3.2 ToC byte....839
A.3.3.3.3 E (Extra) byte....840
A.3.3.3.3.1 General....840
A.3.3.3.3.2 Initial E-byte (CMR)....841
A.3.3.3.3.3 Subsequent E-bytes....842
A.3.4 Frame Data....843
A.3.5 Processing Information (PI) data....843
A.3.5.1 General....843
A.3.5.2 PI data header....844
A.3.5.3 Media time when IVAS PI data is included in RTP packets....846
A.3.5.4 PI data handling during DTX....846
A.3.5.5 Supported PI data types....847
A.3.5.6 Forward direction PI data types....847
A.3.5.6.1 Orientation PI data (forward direction)....847
A.3.5.6.1.1 Orientation data structures....847
A.3.5.6.1.2 Scene orientation....848
A.3.5.6.1.3 Device orientation....848
A.3.5.6.2 Acoustic environment PI data....848
A.4 Payload Format Parameters....852
A.4.1 IVAS Media Type Registration....852
A.4.2 Mapping media type parameters into SDP....854
A.4.3 Detailed Description of Usage of SDP Parameters....854
A.4.3.1 Offer-Answer Model Considerations....854
Annex B (informative): Change history....857
History....858

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

shall indicates a mandatory requirement to do something

shall not indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

should indicates a recommendation to do something

should not indicates a recommendation not to do something

may indicates permission to do something

need not indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

can indicates that something is possible

cannot indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

will indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

will not indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

might indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

might not indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

is (or any other verb in the indicative mood) indicates a statement of fact

is not (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

1 Scope

The present document is a detailed description of the signal processing algorithms of the Immersive Voice and Audio Services (IVAS) coder including the IVAS renderer.

2 References

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 26.441: "Codec for Enhanced Voice Services (EVS); General Overview".

[3] 3GPP TS 26.445: "Codec for Enhanced Voice Services (EVS); Detailed Algorithmic Description".

[4] 3GPP TS 26.447: "Codec for Enhanced Voice Services (EVS); Error concealment of lost packets".

[5] 3GPP TS 26.448: "Codec for Enhanced Voice Services (EVS); Jitter Buffer Management"

[6] 3GPP TS 26.250: "Codec for Immersive Voice and Audio Services (IVAS); General overview".

[7] 3GPP TS 26.251: "Codec for Immersive Voice and Audio Services (IVAS); C code (fixed-point)".

[8] 3GPP TS 26.252: "Codec for Immersive Voice and Audio Services (IVAS); Test Sequences".

[9] 3GPP TS 26.254: "Codec for Immersive Voice and Audio Services (IVAS); Rendering".

[10] 3GPP TS 26.255: "Codec for Immersive Voice and Audio Services (IVAS); Error concealment of lost packets".

[11] 3GPP TS 26.256: "Codec for Immersive Voice and Audio Services (IVAS); Jitter Buffer Management".

[12] 3GPP TS 26.258: "Codec for Immersive Voice and Audio Services (IVAS); C code (floating point)".

[13] C. de Boor and K. Höllig (1987), B-splines without divided differences, in Geometric Modeling, G. Farin ed., SIAM, 21–27.

[14] Borß, C. A Polygon-Based Panning Method for 3D Loudspeaker Setups. In *Audio Engineering Society Convention 137*, Los Angeles, USA, Oct. 2014.

[15] Zotter, F. and Frank, M., All-Round Ambisonic Panning and Decoding, *J. Audio Eng. Soc.*, vol. 60, no. 10, pp. 807-820 (Oct. 2012).

[16] Allen, J. B., & Berkley, D. A., Image method for efficiently simulating small room acoustics. *Journal of the Acoustical Society of America*, 65(4), 943-950, (1979).

[17] Box, G. E. P. and Cox, D. R. (1964). An analysis of transformations, Journal of the Royal Statistical Society, Series B, 26, 211-252.

[18] 3GPP TS 26.118: “Virtual Reality (VR) profiles for streaming applications”.

[19] Ivanic, J., & Ruedenberg, K., Rotation matrices for real spherical harmonics. Direct determination by recursion. *The Journal of Physical Chemistry*, *100*(15), 6342-6347, 1996.

[20] C. R. Helmrich and B. Edler, “Signal-Adaptive Transform Kernel Switching for Stereo Audio Coding,” in Proc. IEEE WASPAA, New Paltz, NY, USA, Oct. 2015.

[21] AES, “AES69-2022: AES standard for file exchange - Spatial acoustic data”, Audio Engineering Society, 2022.

[22] F. Thomas, "Approaching Dual Quaternions From Matrix Algebra," in *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1037-1048, Oct. 2014.

[23] J, Daniel “R,eprésentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimedia”, Thèse de doctorat de l’Université Paris 6, 2001.

[24] M. Chapman, “A Standard for Interchange of Ambisonic Signal Sets. Including a file standard with metadata”, Ambisonics Symposium 2009, Graz, June 25-27.

[25] ISO/IEC 23091-3:2018 - Information technology Coding-independent code points Part 3: Audio.

[26] ISO/IEC 23008-3:2015 - Information technology High efficiency coding and media delivery in heterogeneous environments Part 3: 3D audio.

[27] 3GPP TS 26.249: " Immersive Audio for Split Rendering Scenarios; Detailed Algorithmic Description of Split Rendering Functions".

[28] ETSI TS 103 634: " Digital Enhanced Cordless Telecommunications (DECT); Low Complexity Communication Codec plus (LC3plus)".

[29] ETSI TR 103 633: " Digital Enhanced Cordless Telecommunications (DECT); Low Complexity Communication Codec plus (LC3plus); Performance characterization".

[30] ETSI TS 103 624: "Characterization Methodology and Requirement Specifications for the ETSI LC3plus speech codec".

[31] Bluetooth Special Interest Group, 'Basic Audio Profile', version 1.0.1.

[r1] IETF RFC 4566 (2006): "SDP: Session Description Protocol", M. Handley, V. Jacobson and C. Perkins.

[r2] 3GPP TS 26.114: "IP Multimedia Subsystem (IMS); Multimedia Telephony; Media handling and interaction".

[r3] IETF RFC 3550 (2003): "RTP: A Transport Protocol for Real-Time Applications", Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson.

[r4] IETF RFC 3551 (2003): "RTP Profile for Audio and Video Conferences with Minimal Control", Schulzrinne, H. and S. Casner

[r5] IETF RFC 4867 (2007): "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", Sjoberg, J., Westerlund, M., Lakaniemi, A., and Q. Xie.

[r6] IETF RFC 7160 (2014): "Support for Multiple Clock Rates in an RTP Session", Petit-Huguenin, M. and G. Zorn, Ed.

[X1] ETSI TS 103 634: " Digital Enhanced Cordless Telecommunications (DECT); Low Complexity Communication Codec plus (LC3plus)"

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

frame: an array of audio samples or metadata spanning a 20-ms time duration.

3.2 Symbols

For the purposes of the present document, the following symbols and conventions to mathematical expressions apply:

E	Energy
F	Sample rate
g	Gain
I	Metadata
L	Length of an audio buffer in samples (e.g., L_{frame} is the length of a frame in samples)
M	Mode (e.g., $M_{element}$ is the element mode or M_{core} is the core-coder mode)
N	Number of audio channels
R	Bitrate
s	Audio signal in time domain
S	Audio signal in frequency domain (spectrum)

$s(n)$	(n) indicates the nth sample of the audio signal s
$E(b)$	(b) indicates the bth band in the energy vector E
$S(k)$	(k) indicates the kth frequency bin
$S(k,n)$	(k,n) indicates the nth time slot of the kth frequency bin of the discrete time-frequency spectrum
$s(m;n)$	$(m;n)$ indicates the nth sample within mth subframe
$s(t)$	(t) indicates the time instant t in the continuous time domain
S_i, s_i	Lower index i indicates the ith channel (input or transport) of a multi-channel signal. The indexing starts from 1
$s_{HP20}(n)$	HP20 filtered time domain signal
$s_{inp}(n)$	Input signal to IVAS encoder
S^{MDFT}	Superscript $MDFT$ indicates the type of frequency-domain transform; also FFT and $CLDFB$
$\hat{S}$	Quantized (coded) version (of frequency-domain audio signal)
$\bar{E}$	Mean value (of energy)
$s^{[-1]}(n)$	Upper index indicates a particular frame, e.g., $[-1]$ refers to the previous frame. When omitted, current frame is assumed by default

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

ACELP	Algebraic Code-Excited Linear Prediction
AGC	Adaptive Gain Control
AllRAD	All-Round Ambisonics Decoding
BPF	Bass PostFilter
BRIR	Binaural Room Impulse Response
DirAC	Directional Audio Coding
CBR	Constant Bit Rate
CLDFB	Complex Low-Delay FilterBank
CNA	Comfort Noise Addition
CNG	Comfort Noise Generation
CPE	Channel Pair Element
DFT	Discrete Fourier Transform
DoA	Direction of Arrival
DoF	Degree of Freedom
DTX	Discontinuous Transmission
EC	Entropy Coding
EFAP	Edge Fading Amplitude Panning
FEC	Frame Erasure Concealment
FOA	First-Order Ambisonics
GCC-PHAT	Generalized Cross-Correlation PHAse Transform
HQ	High-Quality
HOA	Higher-Order Ambisonics
HRIR	Head-Related Room Impulse Response
HRTF	Head-Related Transfer Function
IC-BWE	Inter-Channel BandWidth Extension
ICA	Inter-Channel Alignment
ICC	Inter-Channel Coherence
IGF	Intelligent Gap Filling
ILD	Inter-Channel Level Difference
ITD	Inter-Channel Time Difference
ISM	Independent Streams with Metadata
JBM	Jitter Buffer Management
LCLD (codec)	Low-Complexity Low-Delay (codec)
LC3plus	Low Complexity Communication Codec plus
LFE	Low-Frequency Effects
LP	Linear Prediction
MASA	Metadata-Assisted Spatial Audio
McMASA	Multi-Channel MASA

MC	Multi-Channel
MCT	Multi-channel Coding Tool
MD	MetaData
MDCT	Modified Discrete Cosine Transform
MDFT	Modified Discrete Fourier Transform
MDST	Modified Discrete Sine Transform
OLA	OverLap-Add
OMASA	Objects with MASA
OSBA	Objects with SBA
PCA	Principal Component Analysis
PLC	Packet Loss Concealment
SAD	Sound Activity Detection
SBA	Scene-Based Audio
SCE	Single Channel Element
SH	Spherical Harmonics
SNS	Spectral Noise Shaping
SPAR	Spatial Reconstruction
STFT	Short-Term Fourier Transform
TCX	Transform-Coded eXcitation
TD	Time-Domain
TF	Time-Frequency
TNS	Temporal Noise Shaping
TSM	Time Scale Modification
VAD	Voice Activity Detection
VBAP	Vector Base Amplitude Panning
VBR	Variable Bit Rate

4 General description of the coder

4.1 Introduction

The present document is a detailed algorithmic description of the Immersive Voice and Audio Services (IVAS) coder. The IVAS coder is a framework for low-delay speech- and audio coding and rendering targeting stereo or immersive audio communication. It comprises:

- encoder,
- decoder, and
- renderer.

The procedure of this document is mandatory for implementation in all network entities and User Equipment (UE)s supporting the IVAS coder.

The present document does not describe the C code of the IVAS coder. In the case of discrepancy between the algorithmic description in the present document and its C code specifications contained in [7], [12] the C code specifications prevail.

4.2 IVAS codec overview

4.2.1 General

The IVAS codec is an extension of the 3GPP Enhanced Voice Services (EVS) codec [2]. It provides full and bit exact EVS codec functionality for mono speech/audio signal input. It further provides:

- Encoding and decoding of stereo and immersive audio formats such as multi-channel audio, scene-based audio (Ambisonics), metadata-assisted spatial audio (MASA), object-based audio (ISM), and their combination.
- VAD/DTX/CNG for rate efficient stereo and immersive conversational voice transmissions

- Error concealment mechanisms to combat the effects of transmission errors and lost packets. Jitter buffer management is also provided.
- The IVAS codec operates on 20-ms audio frames. In addition, rendering is possible with 5-ms granularity.
- Support for bit rate switching upon command.
- Stereo and immersive audio coding at the following discrete bit rates [kbps]: 13.2, 16.4, 24.4, 32, 48, 64, 80, 128, 160, 192, 256, 384, and 512, with supported bit rate ranges listed in Table 4.2-1.
- Support for WB, SWB and FB audio, with the supported bitrate range listed in Table 4.2-2.

Table 4.2-1: Ranges of supported bitrates for stereo and immersive coding of the IVAS codec

Input audio format	Range of supported bitrates [kbps]
Stereo	13.2 – 256
Scene-based audio (SBA)	13.2 – 512
Metadata assisted spatial audio (MASA)	13.2 – 512
Object-based audio (ISM)	13.2 – 512
Multi-channel audio (MC)	13.2 – 512
Combined ISM and MASA (OMASA)	13.2 – 512
Combined ISM and SBA (OSBA)	13.2 – 512

Note, that for the object-based audio format (ISM) the range of supported bitrates varies based on the number of objects as follows: 13.2 kbps – 128 kbps for 1 ISM, 16.4 kbps – 256 kbps for 2 ISMs, 24.4 kbps – 384 kbps for 3 ISMs, 24.4 kbps – 512 kbps for 4 ISMs.

Table 4.2-2 Supported audio bandwidth per input audio format and bitrate

Input audio format	Bitrates supporting WB [kbps]	Bitrates supporting SWB [kbps]	Bitrates supporting FB [kbps]
Stereo	13.2 – 256	13.2 – 256	32 – 256
Scene-based audio (SBA)	13.2 – 512	13.2 – 512	32 – 512
Metadata assisted spatial audio (MASA)	13.2 – 512	13.2 – 512	32 – 512
Object-based audio (ISM), 1 object	13.2 – 128	13.2 – 128	16.4 – 128
Object-based audio (ISM), 2 objects	13.2 – 256	24.4 – 256	32 – 256
Object-based audio (ISM), 3 objects	24.4 – 384	24.4 – 384	48 – 384
Object-based audio (ISM), 4 objects	24.4 – 512	24.4 – 512	64 – 512
Multi-channel audio (MC)	13.2 – 512	13.2 – 512	32 – 512
Combined ISM and MASA (OMASA)	13.2 – 512	13.2 – 512	32 – 512
Combined ISM and SBA (OSBA)	13.2 – 512	13.2 – 512	32 – 512

4.2.2 Mono (EVS-compatible) Operation

The IVAS codec supports mono operation with EVS compatibility by implementing all EVS functionality in a bit-exact manner. This mode is suitable for applications that require low bitrate and high-quality speech and audio. The codec provides the full range of EVS primary bitrates, from 7.2 kbps CBR / 5.9 kbps VBR to 128 kbps and can operate at narrowband, wideband, super-wideband and full-band sampling rates. EVS compatibility includes the AMR-WB IO modes, with bitrates from 6.6 kbps to 23.85 kbps. All functionalities already present in the EVS codec are retained, as also described in [3]. While staying bit-exact, some EVS code portions were reorganized to allow the extension towards immersive speech and audio services.

The EVS-compatible mono operation is interoperable to other implementations of the EVS codec, which enables seamless integration with existing systems and devices that use the EVS codec.

4.2.3 Stereo Operation

While the EVS codec only supports mono operation that could be combined for dual-mono operation, the IVAS codec supports native stereo coding at bitrates starting from 13.2 kbps to 256 kbps, offering high audio quality operation for stereo signals with all kinds of different stereo cues.

The stereo coding consists of coding modules operating in the time domain and frequency domain.

Low-rate stereo coding (13.2 to 32 kbps) mainly operates based on a combination of DFT-based processing for correlated signals and a time domain approach for uncorrelated signals. Which method is used to code the current frame is decided by a prior stereo classification stage.

For the DFT-based stereo approach, (band-wise) side parameters - including time, phase and loudness differences between the channels, as well as prediction parameters for the side residual - are extracted at the encoder followed by a downmix stage to obtain a mid-signal which is then given to the core-coder. At the decoder, the stereo signal is reconstructed from the downmix signal and the transmitted stereo parameters. For the mid-bitrate of 32 kbps part of the residual side signal is additionally coded and directly transmitted in the bitstream to further improve quality.

For the time-domain stereo approach, time-domain parameters are extracted and a weighted downmix is created resulting in a primary and a secondary channel which are both given to the core-coder but with the primary channel receiving a higher number of bits.

High-rate stereo coding (48-256 kbps) operates based on MDCT-based processing for band-wise stereo encoding. This can be seen as a sort of joint core coding as both stereo and discrete channel operations are done in the same domain without intermediate transforms. For stereo, additional loudness differences are transmitted and each stereo band can be transformed adaptively to a mid/side representation, depending on what is most efficient to code.

The stereo format algorithmic description is discussed in clause 5.3 for the encoder resp. in clause 6.3 for the decoder.

In addition, stereo downmix is supported to generate a mono signal for EVS interoperable stream with no extra delay. Its algorithmic description is discussed in clause 5.11.

4.2.4 Scene-based Audio (SBA, Ambisonics) Operation

Coding of ambisonics signals is supported for 1st- to 3rd-order inputs throughout the full bitrate range. The decoder's output can be SBA (of order 1, 2, or 3), mono, stereo, binaural or multi-channel. This flexibility of input order, bitrate and output format combinations is in part achieved by the combination of covariance-based and directional analysis at different frequencies.

For the lowest 8 bands, covariance analysis with a 20 ms stride is performed at the encoder and corresponding reconstruction is performed at the decoder. For the highest 4 bands, an estimation of the parameters of a psychoacoustic model is implemented with a time resolution of 5 ms.

At the encoder, the covariance-analysis metadata for the higher bands are estimated from these model parameters and combined with the directly calculated metadata for the lower bands. Based on these metadata, a downmix to 1 to 4 channels (dependent on bitrate) is obtained. The downmix channels are then coded with the appropriate core-coder.

At the decoder, the downmix channels plus the metadata are received. The latter comprise the transmitted covariance-analysis metadata and model parameters for the lower and higher bands, respectively. These metadata are used to reconstruct the HOA signal and render to the requested output format. In this, the psychoacoustic model parameters for the 8 lower frequency bands are estimated from the reconstructed audio channels. The model-based reconstruction allows for the output SBA order on the decoder side to be higher than the input order on the encoder side.

Low-latency operation (less than or equal to 38 ms end-to-end) is achieved by using a very-low-latency 1-ms MDFT-based filterbank at the encoder and a 5-ms CLDFB-based filterbank at the decoder, which additionally enables the type of signal modifications that are necessary for rendering to a broad variety of output configuration.

The SBA format algorithmic description is discussed in clause 5.4 for the encoder resp. in clause 6.4 for the decoder.

4.2.5 Metadata-assisted Spatial Audio (MASA) Operation

The IVAS codec supports coding of parametric spatial audio format called metadata-assisted spatial audio (MASA). This format is specifically optimized for the direct immersive audio capture from smartphones and other form factors that can be unsuitable for dedicated spherical microphone arrays.

The MASA format is based on 1-2 audio channels and associated metadata that is provided for each audio frame. The MASA spatial metadata describes the spatial audio characteristics of the captured immersive audio using several spatial parameters including spatial direction information, directional and non-directional energy ratios, and two types of coherence information. The spatial metadata is provided in each frame according to a time-frequency resolution of 4 subframes and 24 frequency bands. The MASA descriptive metadata provides additional information relating to the creation and understanding of the MASA audio signal.

The coding in this operation is based on compression of the metadata exploiting detected redundancies and prioritization of selected parameters at each bitrate. The 1 or 2 audio transport channels are coded using the SCE and CPE coding capabilities. A joint bitrate allocation between these two coding blocks is based on a metadata analysis and simplification processing.

The MASA format can be flexibly rendered for binaural or loudspeaker reproduction, including mono and stereo playback. Rendering to Ambisonics is also supported. Furthermore, decoded MASA format bitstream can be directly output from the decoder without rendering as a fully compliant MASA format output for further processing.

The MASA format algorithmic description is discussed in clause 5.5 for the encoder resp. in clause 6.5 for the decoder.

4.2.6 Objects (Independent Streams with Metadata, ISM) Operation

The IVAS codec supports coding of 1 to 4 independent audio objects with associated metadata. The coding is based on input Independent Streams (ISMs) analysis, metadata coding, and inter-object core-coder SCEs bitrate adaptation. The ISM coding employs two coding schemes, namely the “discrete mode in which each object is coded by one SCE, and parametric mode which downmixes 3 or 4 objects to two transport channels coded by two SCEs. The coded and transmitted metadata consists of azimuth and elevation (up to 48 kbps) or of azimuth, elevation, radius, pitch, and yaw (64 kbps and up). Alternatively, the metadata can consist of a non-diegetic panning gain. Objects with different types of metadata can be combined; the type of metadata of an object can change from one frame to another. On the decoder side, the objects can then be rendered to the requested output configuration. When the requested output is binaural, objects with non-diegetic metadata are panned on left/right channels while other objects are rendered accordingly to their associated metadata.

The ISM format algorithmic description is discussed in clause 5.6 for the encoder resp. in clause 6.6 for the decoder.

4.2.7 Multi-Channel (MC) Operation

Coding of multi-channel inputs is available for the channel layouts 5.1, 7.1, 5.1+2, 5.1+4, and 7.1+4. The coding technique is selected from a set of coding modes based on the available bitrate and specified channel layout. The general principle in technique selection is to aim for best possible quality given the allowed bitrate. For all techniques, LFE channel coding is also offered either separately or within the technique. The multi-channel operation supports output to mono, stereo, multi-channel (at the same or any other layout with up to 16 speakers), Ambisonics (at up to order 3), and binaural.

The MC format algorithmic description is discussed in clause 5.7 for the encoder resp. in clause 6.7 for the decoder.

4.2.8 Combined Objects and SBA (OSBA) Operation

The IVAS codec supports combined input format of ISMs and SBA called OSBA (Objects with Scene Based Audio). There are two different operation modes for this type of input: a low- (below 256 kbps) and a high- (256 kbps and up) bitrate one. In the low-bitrate mode the objects are pre-rendered into the SBA input, which is then processed in exactly the same way as in the native SBA format. The high-bitrate mode features separate coding of 1 – 4 independent objects with associated metadata. Specifically, the object metadata are encoded in the same way as in the native ISM format and written into the bitstream alongside the SBA metadata. The object audio channels are input to the MCT coding tool together with the downmix channels of the SBA coder. Therefore, efficient bit allocation techniques of MCT are applied to all coded audio channels together. On the decoder side, the audio can then be rendered to the requested output configuration.

The OSBA format algorithmic description is discussed in clause 5.8 for the encoder resp. in clause 6.8 for the decoder.

4.2.9 Combined Objects and MASA (OMASA) Operation

The IVAS codec supports combined input format of ISMs and MASA called OMASA (Objects with Metadata-Assisted Spatial Audio). The input consists of 1 – 4 independent objects with associated metadata and the 1 or 2 transport channels of MASA and the associated metadata. The encoding configuration for OMASA is decided based on the codec bitrate and the number of objects. There are 4 coding modes designed such that the encoded output quality is optimal. Based on the selected coding mode a combination of one audio channel pair with MASA metadata is encoded as MASA format data and none, one or all objects are encoded within ISM format encoding framework. When one or more objects are encoded, the adaptive inter-format bitrate allocation between the object(s) and the MASA format data is used at each frame based on the audio content. On the decoder side, the audio can then be rendered to the requested output configuration. For example, the combined format input format with ISMs and MASA can carry several speech signals with one having a spatial background or, e.g., 1 – 4 talkers and an ambience audio representation that comprises an ambient (diffuse) background parameterized according to the MASA format, defined in Annex A of [12].

The OMASA format algorithmic description is discussed in clause 5.9 for the encoder resp. in clause 6.9 for the decoder.

4.2.10 Discontinuous Transmission (DTX) Operation

DTX is a functionality of operation where the encoder encodes frames containing only background noise with a lower bit rate and lower packet frequency than normally used for encoding speech. A terminal and the network may adapt their transmission scheme to take advantage of the smaller frames and longer transmission interval to reduce power consumption, average bit rate and network activity. The discontinuous transmission (DTX) functionality of the IVAS codec includes voice activity detection (VAD) and comfort noise generation (CNG). DTX functionality is supported for IVAS operation points, i.e., audio formats and bitrates, that are especially optimized for efficient stereo and immersive conversational voice transmissions.

The size of the SID frames for EVS interoperable modes is unchanged relative to EVS, 48 bits for EVS primary modes. For IVAS modes, the SID frame size is 104 bits. The default SID frame interval is once per 8 frames, but other update intervals are also supported.

DTX is supported in IVAS formats and bitrates as summarized in Table 4.2-3. The algorithmic description of DTX/CNG is then discussed in related IVAS format specific clauses.

Table 4.2-3: DTX support overview in the IVAS codec

Input audio format	DTX support
Stereo	Yes, up to 256 kbps
Scene-based audio (SBA)	Yes, up to 80 kbps
Metadata assisted spatial audio (MASA)	Yes, up to 512 kbps
Object-based audio (ISM)(1)	Yes, up to 512 kbps
Multi-channel audio (MC)	No
Combined ISM and SBA (OSBA)	No
Combined ISM and MASA (OMASA)	No

4.3 Input/output audio configurations

4.3.1 Input/output audio sampling rate

The IVAS coder is capable of processing input audio signals sampled at 16, 32 and 48 kHz. The sampling frequency is denoted as F_s. The same set of sample rates is also supported by the IVAS decoder. In case of EVS-compatible mono operation the IVAS coder supports also 8 kHz sample rate both for input and output signals. The input audio is processed in frames of 20 ms.

4.3.2 Input/output audio formats

The IVAS coder accepts the following input audio formats:

– single-channel mono audio format denoted as $s(n)$

– two-channel stereo and binaural input audio format, where the left channel is denoted as $s_1(n)$ and the right channel is denoted as $s_2(n)$

– scene-based (ambisonic) input audio format, where the ambisonic order is denoted as l and the number of individual input channels is $(l+1)^2$. The individual input channels are stored in the ACN component ordering, denoted as $s_{ml}(n)$, where $m \in \langle -l, \ldots, +l \rangle$ is the ambisonic degree. In case of the first-order ambisonic format (FOA), the individual input channels may also be denoted as:

$$s_{00}(n) = W(n)$$
$$s_{-11}(n) = X(n)$$
$$s_{01}(n) = Y(n)$$
$$s_{11}(n) = Z(n)$$

For both, first-order ambisonic signals and higher-order ambisonic signals (HOA), the notation $s_{ml}(n)$ may be simplified to:

$$s_k(n) = s_{ml}(n), \qquad k = l^2 + l + m$$

- object-based input audio format (ISM), where the number of objects is denoted as N_{ISM} and the individual "streams" related to the objects are denoted as $s_k(n)$ where $k \in \langle 1, \ldots, N_{ISM} \rangle$. Each individual stream is associated with its input metadata signal, denoted as $I_k(m)$ where m is the frame index.

- multi-channel input audio format (MC), where the individual channels are denoted as $s_k(n)$ where $k \in \langle 1, \ldots, N_{chan} \rangle$. The input channels correspond to one of the following loudspeaker layouts, where the loudspeaker positions are assumed to have azimuth and elevation as per ISO/IEC 23091-3:2018 Table 3 [24], unless otherwise noted below. The channel order is as per ISO/IEC 23008-3:2015 Table 95 [25]:

Table 4.2-4: Ordering of input channels in MC format

loudspeaker layout	CICP layout
5.1	CICP6
5.1+2	CICP14, 35°elevation
5.1+4	CICP16, 35°elevation
7.1	CICP12
7.1+4	CICP19, 35°elevation

- metadata-assisted spatial audio (MASA) format, where for mono-MASA (MASA1), the mono channel is denoted as $s(n)$ and, for stereo-MASA (MASA2), the left channel is denoted as $s_1(n)$ and the right channel is denoted as $s_2(n)$. The MASA audio channel(s) is/are associated with the MASA metadata that is denoted as $I(m)$ where m is the frame index

- objects with metadata-assisted spatial audio (OMASA), which is a combination of MASA and object-based input audio format (ISM)

- objects with scene-based (ambisonic) input audio format (OSBA), which is a combination of SBA and object-based input audio format (ISM)

4.4 Algorithmic delay

The input signals (audio, or audio and metadata) are processed using 20-ms frames. The codec algorithmic delay depends on the input/output audio formats as described in Table 4.4-1.

Table 4.4-1: IVAS algorithmic delay for different input/output format combinations (rounded to integer milliseconds; in case multiple values are provided they depend on the bitrate)

		Decoder output format						
		Mono	Stereo	Multi-Channel	Binaural audio	Scene-based audio	Object-based audio	Metadata-assisted spatial audio
Encoder input format	Mono	32						
	Stereo	32	32	32				
	Binaural	32			32			
	Multi-channel	32	32	32 / 37	32 / 37	32 / 37		
	Scene-based audio	33	33	38	38	38		
	Object-based audio	32	32	32 / 37	32 / 37	32 / 37	32 / 37	
	Metadata-assisted spatial audio	32 / 37	37	37	37	37		32 (NOTE
	OSBA	33	33	38	38	38		
	OMASA	32 / 37	37	37	37	37		

NOTE: Metadata-assisted spatial audio (MASA) decoder output allows also for mono or stereo decoder output at 32 ms algorithmic delay by stripping the metadata file.

The algorithmic delay related to the core-coder coding in IVAS is 32 ms similarly as in EVS though its splitting between the encoder and the decoder is slightly different. It consists of 8.75 ms for the encoder look-ahead and 3.25 ms for the decoder delay related to the time-domain BWE and resampling in the DFT domain.

Further, the IVAS delay consists of 5 ms delay related to the rendering to the related output configuration, thus making the overall delay of 32 ms in some set-ups and 37 ms in other set-ups.

Finally, in SBA format, an additional encoder delay of 1 ms is present and it is related to the filter-bank analyses prior to the encoding.

It is also noted that the delay figures exclude any HRIR/BRIR induced delay.

The codec delay for mono (EVS) operation and stereo downmix operation in EVS compatible operation is 32 ms described in clause 4.3 in [3].

4.5 Rendering overview

4.5.1 Rendering introduction

The codec for Immersive Voice and Audio Services is part of a framework comprising of an encoder, decoder, and renderer. An overview of the audio processing functions of the receive side of the codec is shown in Figure 4.5-1. This diagram is based on [6], with rendering features highlighted.

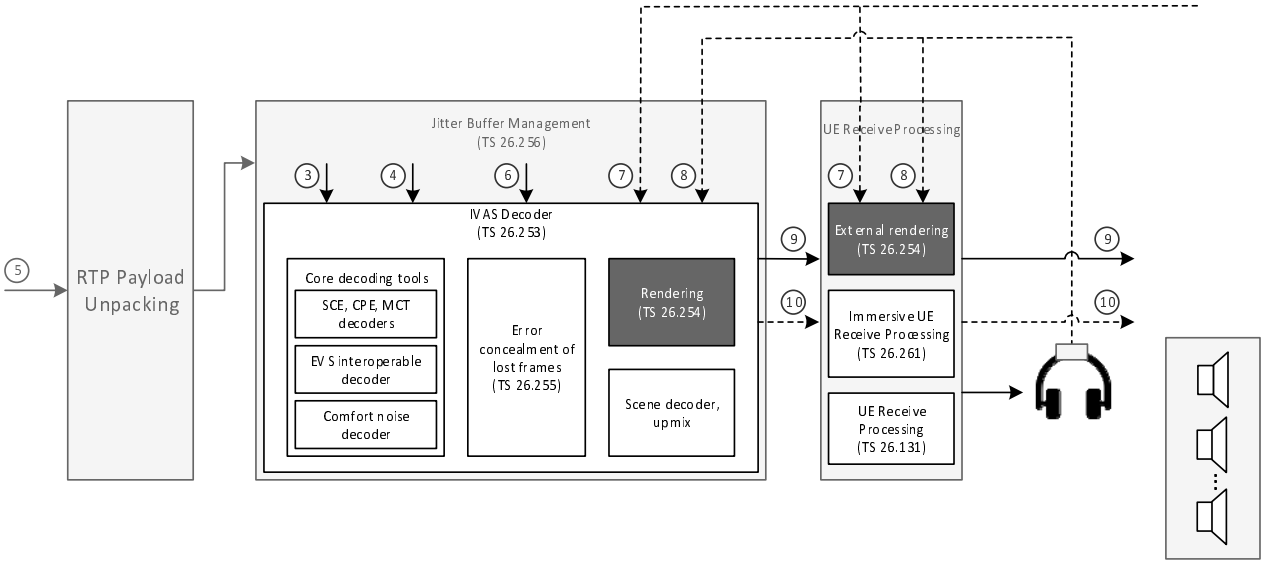


Figure 4.5-1: Overview of IVAS audio processing functions – receiver side

The interfaces are marked consistently with [6] using the following numbers:

3: Encoded audio frames (50 frames/s), number of bits depending on IVAS codec mode,
4: Encoded Silence Insertion Descriptor (SID) frames,
5: RTP Payload packets,
6: Lost Frame Indicator (BFI),
7: Renderer config data,
8: Head-tracker pose information and scene orientation control data,
9: Audio output channels (16-bit linear PCM, sampled at 8 (only EVS), 16, 32, or 48 kHz),
10: Metadata associated with output audio.

Rendering is the process of generating digital audio output from the decoded digital audio signal. Rendering is used when output format is different than input format. In case output format is the same as input format, the decoded audio channels are simply passed through to the output channels. Binaural rendering is a special case, where binaural output channels are prepared for headphone reproduction. This process includes head-tracking and scene orientation control, head-related transfer function processing, and room acoustic synthesis. IVAS rendering is integrated with IVAS decoder but can also be operated standalone as external rendering while bypassing the internal renderer. The external renderer can be applied e.g., in the case of rendering outputs originating from multiple sources, such as decoders or audio streams.

A special feature of the renderer is that it supports split operation with pre-rendering and transcoding to a head-trackable intermediate representation that can be transmitted to a post-rendering end-device. This enables moving a large part of the processing load and memory requirements for IVAS decoding and rendering to a (more) capable node/UE while offloading the final rendering end-device. The IVAS specific split rendering functionality is mostly described in the present document whereas more generic split rendering functionality is specified in TS 26.249 [27].

4.5.2 Internal IVAS renderer

The internal IVAS renderer is integrated into the IVAS decoder. In case of specific operating points, this integration allows for combining decoding and rendering processes, resulting in efficient processing. Therefore, rendering algorithm descriptions associated with specific audio formats are discussed in clause 6. The rendering modes and rendering control are discussed separately in clause 7.

4.5.3 External IVAS renderer

The external IVAS renderer supports all the functionality of the internal renderer. However, since the external renderer operates stand-alone, combined decoding and rendering processing is not available.

4.5.4 Interface for external rendering

IVAS renderer and its interface provide support to IVAS codec design constraints. The details of the rendering library API are provided in [7] for the fixed-point code and [12] for the floating-point code. The details of the rendering library API are provided in [9].

4.5.5 Split rendering

The IVAS renderer supports splitting the rendering process between a capable device or network node (where the IVAS decoding and rendering happen) and a less capable device with limited computing and memory resources and motion-sensing for head-tracked binaural output. Split Rendering support consists of three core components, a metadata-based pose correction scheme (in CLDFB domain), a coding scheme in CLDFB domain for low-complexity low-delay stereo coding (LCLD), see clause 7.6.4, and the pre-existing coding scheme LC3plus, see clause 7.6.5. IVAS split rendering functionality is illustrated in Fig. 4.5-2.

The metadata-based pose correction scheme allows adjusting in a lightweight process a binaural audio signal originally rendered for a first pose according to a second pose. In many split rendering scenarios, the first pose is the potentially outdated lightweight-device pose available at the pre-renderer while the second pose is the current and accurate pose available at the lightweight-device. The metadata is calculated at the capable device or network node based on additional binaural renditions at probing poses different from the first pose. For increasing degrees-of-freedom (DOF) an increasing number of additional binaural renditions at different probing poses is required. The metadata is transmitted to the lightweight device along with the coded binaural audio signal rendered for the first pose.

The binaural audio signal rendered to the first pose is encoded using one of the two codecs, LCLD or LC3plus. The two codecs have complementary properties giving implementors the freedom to make individual trade-offs between complexity, memory, latency, and rate-distortion performance and to implement a design that is optimized for a given IVAS service and hardware configuration. The coding scheme for the binaural audio signal defaults based on the domain the binaural renderer operates in, with LCLD being the default for CLDFB-based rendering and LC3plus being the default where rendering is performed in time-domain. The default can also be overwritten, meaning that irrespective of the rendering domain, either the LCLD or the LC3plus codec or even transmission over a PCM interface may be selected. The latter allows using further coding solutions for the binaural audio signal.

The split rendering can operate at various DOFs, ranging from 0-DOF (no pose correction) to 3-DOF (pose correction on the three rotational axes yaw, pitch, roll) at bit rates from 256 kbps (0-DOF) to 384 - 768 kbps (1 to 3-DOF). The split rendering operates on and delivers audio sampled at a rate of 48 kHz.

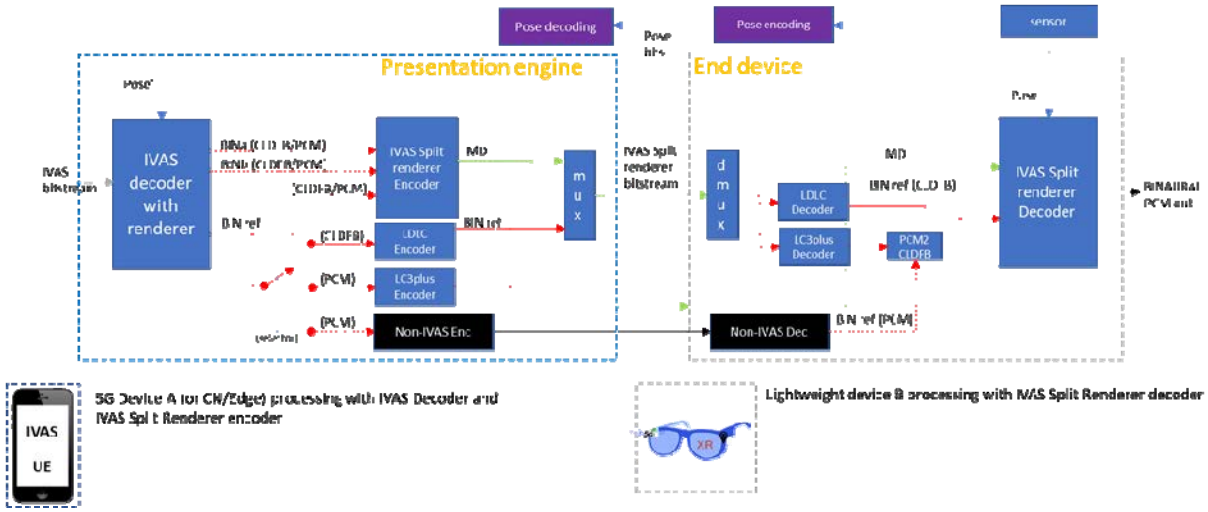


Figure 4.5-2: IVAS Split Rendering functions

4.6 Organization of the rest of the Technical Standard

The detailed description is organized as follows:

- In clause 5, the detailed functional description of the encoder is given. This begins with a description of the common processing and coding tools, followed by each of the coding operations based on the input audio formats.
- In clause 6, the detailed functional description of the decoding is given. This description begins with common decoder processing, followed by each of the decoding operations for processing bitstream. The decoding processing also describes the rendering specific for the audio input format coding modes and the audio output format.
- In clause 7, the detailed functional description of the rendering operations is given. This includes the rendering modes, rendering control functionalities, and the pre-rendering capability for rendering multiple audio inputs into one for encoding.
- Bit allocation is summarized in clause 8.

Annex A provides the RTP payload format and SDP parameter definitions for the IVAS codec.

5 Functional description of the encoder

5.1 Encoder overview

The IVAS codec encoder expects mono, stereo, objects, multichannel, ambisonics, MASA, combination of objects and MASA, or combination of objects and SBA as input audio channels. In case of objects or MASA, also input metadata are expected. The encoder analyzes the scene, derives the spatial audio parameters and downmixes the input channels to the so-called transport channels which are subsequently processed by the encoding tools. These tools comprise Single Channel Elements (SCE comprising one core-coder, see clause 5.2.3.2), Channel Pair Elements (CPE comprising two core-coders, see clause 5.2.3.3), and Multichannel Coding Tool (MCT comprising a joint coding of multiple core-coders, see clause 5.2.3.4) while the core-coder is inherited from the EVS codec with additional flexibility and variable bitrate (clause 5.2.2).

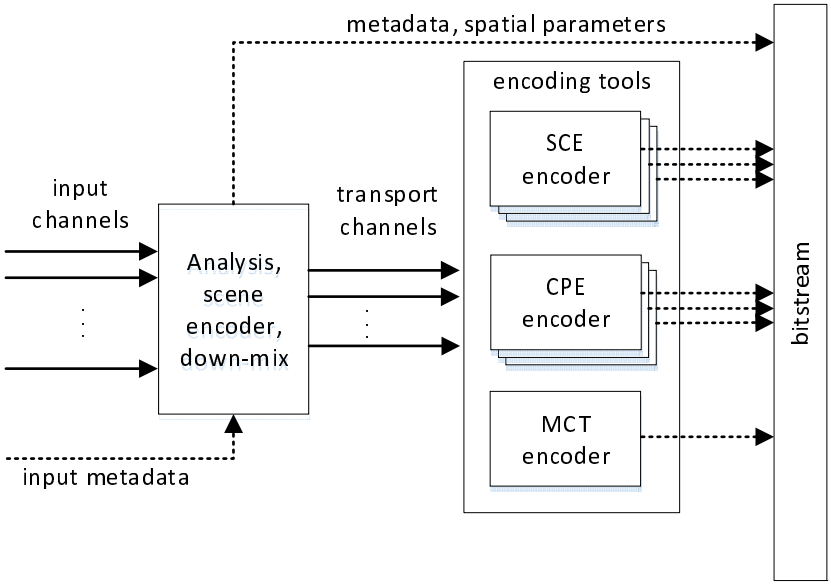


Figure 5.1-1: Encoder Data Flow from input data to IVAS bitstream

The IVAS format is signalled in every active frame at the beginning of the IVAS bitstream. The number of signalling bits in active frames and their values corresponding to individual formats is summarized in Table 5.1-1.

Table 5.1-1: IVAS format signalling, active frames

IVAS format	configuration	number of bits	Value
STEREO	-	2	0
ISM	< 24.4 kbps	2	2
	≥ 24.4 kbps	3	4
SBA	-	3	6
MASA	-	3	7
MC	-	2	1
OSBA	< 24.4 kbps	3	6
	≥ 24.4 kbps	4	11
OMASA	Rend mode	3	7
	other modes	4	10

In SID frames of the DTX operation, the IVAS format is signalled at the beginning of the IVAS bitstream. The number of signalling bits in SID frames and their values corresponding to individual formats is summarized in Table 5.1-2.

Table 5.1-2: IVAS format signalling, SID frames

IVAS format	configuration	number of bits	Value
STEREO	Unified stereo	2	0x0
	MDCT stereo	2	0x1
ISM	-	2	0x2
SBA	1 TC	2	0x5
	2 TCs	2	0x6
	3+ TCs	N/A	N/A
MASA	-	2	0x3
	-	2	0x7
MC	-	N/A	N/A
OSBA	1 TC	2	0x5
	2 TCs	2	0x6
	3+ TCs	N/A	N/A
OMASA	-	N/A	N/A

5.2 Common processing and coding tools

5.2.1 High-pass filtering

The input signal $s_{in}(n)$ sampled at the input sampling frequency F_s is high-pass filtered to supress undesired low-frequency components. The transfer function of the HP filter has a cut-off frequency of 20 Hz (–3 dB) and is given by

$$H_{20Hz} = \frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}}. \quad (5.2\text{-}1)$$

where a_i and b_i are the coefficients of the HP filter. The coefficients of the HP filter are constant and defined in Table 1 of [3].

In case of multi-channel input signal, the HP filter is not applied on the LFE channel (index 3). In case of ambisonic input format the HP filter is only applied on analyzed channels (see clause 5.4.2).

The input signal, filtered with the HP filter, is denoted as $s_{HP}(n), n = 0, \dots, N-1$. In case of skipping the HP filtering operation $s_{HP}(n) = s_{in}(n)$ in channel n.

5.2.2 Core-coder processing

5.2.2.1 Overview

The core-coder in the IVAS codec is based on the EVS core-codec [3] with added flexibility and adaptation for multichannel processing. In the remaining part of clause 5.2.2, only the differences of the IVAS core-coder wrt. the EVS core-coder [3] are provided.

The core-coder module bitrate (one per one core-coder) is not directly signalled in the bitstream but it is derived from the bit-budgets of all codec modules in the following steps:

First, an IVAS format signalling is determined and written to the bitstream.

Second, stereo or spatial coding tools processing (if present) is done and their respective bit-budgets computed.

Similarly, a metadata coding module (if present) is processed and its bit-budget is computed.

The bit-budgets for IVAS format signalling, the stereo / spatial coding tools and metadata coding module are then subtracted from the IVAS total bit-budget.

Finally, the remaining bit-budget is distributed among core-coders based on the type of the core. The details of the LP-based core-coding are provided in clause 5.2.2.3.2 while the MDCT-based core-coder details are provided in clause 5.2.2.3.2.6.

5.2.2.2 Core-coder front pre-processing

5.2.2.2.1 General

The look-ahead is shorter in the IVAS coder compared to the EVS coder by 0.9375 ms (corresponding to a Finite FIR resampling delay (see clause 5.2.2.2.2). This has an impact on the procedure of pre-processing the mono signal or stereo signal (down-mixed signal for TD and DFT stereo modes) in every frame as described in next clauses.

5.2.2.2.2 Sample rate conversion

The shorter look-ahead in IVAS wrt. EVS has an impact on the procedure of resampling the down-processed signal (down-mixed signal for TD and DFT stereo modes) in every frame. Two methods are used:

- SCE mono, TD stereo, MDCT stereo: time-domain FIR resampling (decimation, see clause 5.1.3 in [3]) is performed using the delay of 0.9375 ms. As this resampling delay is not available in IVAS, the resampling delay is compensated by adding zeroes at the end of the down-mixed signal. Consequently, the 0.9375 ms long compensated part of the down-mixed signal needs to be recomputed (resampled again) at the next frame;

- DFT stereo: Resampling is performed in the DFT domain and, therefore, introduces no additional delay.

The resampling in the SCE coder, the DFT stereo coder, the TD stereo coder and the MDCT stereo coder, is done from the input sampling rate (16, 32, or 48 kHz) to the internal sampling rate (12.8, 16, 25.6, or 32 kHz). The resampled signal is then used in the pre-processing and the core-coding.

Also, the look-ahead contains a part of the mono or a down-processed signal (down-mixed signal for TD and DFT stereo modes) signal that is not accurate but rather extrapolated or estimated which also has an impact on the resampling process. The inaccuracy of the look-ahead mono or down-processed signal (down-mixed signal for TD and DFT stereo modes) depends on the current stereo coding mode:

- DFT stereo: The length of 8.75 ms of the look-ahead corresponds to a windowed overlap part of the down-mixed signal related to an OLA part of the DFT analysis window, respectively an OLA part of the DFT synthesis window. In order to perform pre-processing on an as meaningful signal as possible, this look-ahead part of the down-mixed signal is redressed (or unwindowed, i.e. the inverse window is applied to the look-ahead part). As a consequence, the 8.75 ms long redressed down-mixed signal in the look-ahead is not accurately reconstructed in the current frame;

- TD stereo: Before time-domain down-mixing, an Inter-Channel Alignment (ICA) is performed using an Inter-channel Time Delay (ITD) synchronization between the two input channels $l(n)$ and $r(n)$ in the time-domain. This is achieved by delaying one of the input channels (l or r) and by extrapolating a missing part of the down-mixed signal corresponding to the length of the ITD delay; a maximum value of the ITD delay is 7.5 ms. Consequently, up to 7.5 ms long extrapolated down-mixed signal in the look-ahead is not accurately reconstructed in the current frame.

- SCE mono, MDCT stereo: By default, no down-mixing or time shifting is performed, thus the lookahead part of the input signal is accurate.

The redressed/extrapolated signal part in the look-ahead is not subject to actual coding but used for analysis and classification. Consequently, the redressed/extrapolated, signal part in the look-ahead is re-computed in the next frame

and the resulting mono or down-processed signal (down-mixed signal for TD and DFT stereo modes) is then used for the actual core-coding. The length of the re-computed signal depends on the stereo mode and coding processing:

- DFT stereo: The 8.75 ms long signal is subject to re-computation both at the input stereo signal sampling rate and internal sampling rate;
- TD stereo: The 7.5 ms long signal is subject to re-computation at the input stereo signal sampling rate while the 7.5 + 0.9375 = 8.4375 ms long signal is subject to re-computation at the internal sampling rate.
- MDCT stereo: Re-computation is usually not needed at the input stereo signal sampling rate while the 0.9375 ms long signal is subject to re-computation at the internal sampling rate. However, in practice, some signals with length of 7.5 + 0.9375 = 8.4375 ms are recomputed similarly as in the TD stereo.
- SCE mono: Re-computation is usually not needed at the input stereo signal sampling rate while the 0.9375 ms long signal is subject to re-computation at the internal sampling rate.

5.2.2.2.3 Pre-emphasis

In IVAS, the pre-emphasis is applied to down-sampled signal and corresponds to the pre-emphasis operation from EVS (see clause 5.1.4 in [3]). In case of the TD / MDCT stereo or DFT stereo, also redressed samples from previous frame are recomputed in the current frame while the length of the redressed segment is 7.5 ms in the TD and MDCT stereo and 3.125 ms in the DFT stereo.

5.2.2.2.4 Spectral analysis

The spectral analysis in IVAS reuses the spectral analysis from EVS (see clause 5.1.5 in [3]). The positions of spectral analysis windows are the same as in EVS which means that the redressed samples from the end of the lookahead are employed in the analysis.

There is only one exception in the DFT stereo in which the DFT coefficients are already available from the DFT stereo processing (see 5.3.2.4.3). Consequently, the spectral analysis step is thus skipped and energies per critical band are directly computed from available DFT coefficients.

5.2.2.2.5 Signal activity detection

When operating in EVS mode, the signal activity detection operates as in clause 5.1.12 of [3]. When the IVAS core-coding is active, the CLDFB domain analysis is not present. Hence, the signal activity detection related to the CLDFB domain parameters as described in clause 5.1.12.6 of [3] is not employed.

5.2.2.2.6 Bandwidth detector

The Audio Bandwidth Detection (BWD) algorithm in IVAS is similar to the BWD algorithm in EVS (see clause 5.1.6 in [3]) and it is applied in its EVS-based form in the ISM format, DFT stereo and TD stereo modes. However, in the MDCT stereo mode and MCT coding tool (including higher-bitrate SBA, higher-bitrate MASA, multi-channel format, or combined formats), updates to the BWD were introduced in order the BWD is present at all operating points in IVAS. The updates consist of a BWD that comprises an analysis of the sound signal module and a final bandwidth (BW) decision module which is located upstream of the sound analysis module and which depends on the result of the analysis module while the analysis module is integrated to the core-coder stage and the final BW decision module is integrated to the front pre-processing stage.

The BWD algorithm comprises several operations: a) computation of mean and maximum energy values in a number of spectral regions of the input signal; 2) updating long-term parameters and counters; and 3) final decision about the detected and thus coded audio bandwidth while operating in a transform domain. The transform domain is the CLDFB domain in case of EVS or DCT domain in case of AMR-WB IO mode within IVAS (see clause 5.1.6 in [3]).

In IVAS, the BWD is run on all transport channels (except of LFE channel in the MC format) and the transform domain is the CLDFB domain in case of the SCE coding tool and DFT stereo and TD stereo within the CPE coding tool similarly as in EVS. However, in case of MDCT stereo within the CPE tool or the MCT tool, it is the MDCT spectrum which forms the BWD energy vector.

In the case of the MDCT energy vector, there are nine frequency bands of interest whereby the width of each band is 1500 Hz. One to four frequency bands are assigned to each of the spectral regions as defined in Table 5.2-11 where i is

the index of the frequency band and k_{start} is the energy band start index and k_{stop} is the energy band end index (see clause 5.1.6.1 in [3]).

Table 5.2-1: MDCT bands for energy calculation in BWD

i	k_{start}	k_{stop}	band-width in kHz	spectral region
0	1	1	1.5 – 3.0	*nb*
1	3	3	4.5 – 7.5	*wb*
2	4	4		
3	6	6	9.0 – 15.0	*swb*
4	7	7		
5	8	8		
6	9	9		
7	11	11	16.5 – 19.5	*fb*
8	12	12		

The BWD analysis is then adjusted from the EVS native BWD algorithm such that the MDCT spectrum is scaled proportionally to the input sampling rate. The energy $E_{bin}(i)$ (see clause 5.1.6.1 in [3]) of the MDCT spectrum of each transport channel signal in the MDCT stereo mode is thus computed in the nine frequency bands as follows:

$$E_{bin}(i) = \sum_{k=k_{start}(i)\cdot b}^{k=k_{stop}(i)\cdot b+b} S^2(k), i = 0, \dots, 8, \quad (5.2\text{-}2)$$

where $S(k)$ is the MDCT spectrum and the width of the energy band is $b = 60$ samples (which corresponds to 1500 Hz regardless of the sampling rate). The rest of the BWD algorithm is the same as in EVS and consists of computing long-term values of the mean energy of spectral regions, updating spectral region counters, and comparing them to given thresholds (see clause 5.1.6.1 in [3] for more details).

There are however three other particularities in IVAS: 1) the BWD analysis in the MDCT domain is not done at the front pre-processing stage but only at the beginning of the TCX core coding where the MDCT spectrum is available from the TCX core processing. 2) Consequently, the final BWD decision operation is postponed to the front pre-processing of the next frame. Thus, the former EVS BWD algorithm is split into two parts; the BWD analysis operation (i.e. computing energy values per frequency band and updating long-term counters) is done at the beginning of current TCX core coding and the final BWD decision operation is done only in the next frame before the TCX core encoding starts. 3) The BWD analysis is skipped in switching frames (tcx_last_overlap_mode == TRANSITION_OVERLAP) having a longer duration.

Next, in MDCT stereo coding, the final BWD decision from the decision module about the input and thus coded audio bandwidth is done not separately for each of the two channels (or multiple channels in case of MCT) but as a joint decision for both channels. In other words, in the MDCT stereo coding, both channels are always coded using the same audio bandwidth and the information about the coded audio bandwidth is transmitted only once per one CPE. If the final BWD decision is different between the two CPE channels, both CPE channels are coded using the broader audio bandwidth BW of the two channels. E.g. in case that the detected audio bandwidth BW is the WB bandwidth for the first channel and the SWB bandwidth for the second channel, the coded audio bandwidth BW of the first channel is rewritten to SWB bandwidth and the SWB bandwidth information is transmitted in the bitstream. The only exception is a case when one of the MDCT stereo channels corresponds to the LFE channel, then the coded audio bandwidth of the other channel is set to the audio bandwidth of this channel. This is applied mostly in the MC format mode when multiple MC channels are coded using several MDCT stereo CPEs. The information about the coded audio bandwidth is then sent in the bitstream as one joint parameter for the multichannel coding.

Finally, the BWD uses a hysteresis for switching between coded bandwidths. In general, a shorter hysteresis of 10 frames is used when switching from a lower bandwidth to a higher one while a longer hysteresis of 100 frames is used when switching from a higher bandwidth to a lower one. In IVAS, the lower hysteresis is further adjusted in the following set-ups: a) MCT is present, b) the MDCT stereo with element bitrate equal or higher than 48 kbps, c) ISM format with the element bitrate equal or higher than 32 kbps. In these set-ups the shorter hysteresis is set to zero frames.

5.2.2.2.7 Time domain transient detection

5.2.2.2.7.1 General

The time-domain transient detection in IVAS is based on the EVS transient detector as described in clause 5.1.8. of [3]. Some adaptations have been made for IVAS which will be described in the following.

5.2.2.2.7.2 Low-rate adaptation of transient detection

Low-rate adaption of transient detection is based on forward and reverse time direction analysis to detect a transient attack and transient release in the input signal. First the subframe energies, where each subframe is 2.5 ms long, are computed on a high pass filtered signal of the input as described in clause 5.1.8 (see equation 36) of Reference [3]. For each subframe a low pass filtered max energy envelope or accumulated energy is computed according to equation 37 of [3].

For the forward analysis a transient attack is detected if the energy of the subframe, $E_{TD}(i)$, where $i = -2, \ldots, 6$ is above the low pass filtered max energy envelope by factor of ϑ_{fwd}. ϑ_{fwd} is set to 8.5 if condition $c_{3,adapt-fwd}$ below is satisfied otherwise it is set to 8.0.

$$m_{sum} > th_{tonal} * 0.8 \tag{5.2-3}$$

where m_{sum} and th_{tonal} are defined in clause 5.1.11.2.5 of [3].

For transient release analysis, transient detection is performed in the reverse time direction. A low pass filtered energy envelope in the time reversed direction is computed as follows:

$$E_{accRev} := \max(E_{TD}(i+1), 0.8125 E_{accRev}, i = 3, .., -4 \tag{5.2-4}$$

An additional high-resolution analysis within the 5th subframe of the preceding frame, denoted as $E(-4)$, is performed to set a $ramp\_up\_flag$ indicating higher energy in the second half of the subframe compared to the first half. The $ramp\_up\_flag$ is set when the condition below is satisfied.

$$\sum_{j=\frac{k}{2}}^{j=k-1} sb_{-4}(j)^2 > \sum_{j=0}^{\frac{k}{2}-1} sb_{-4}(j)^2 \tag{5.2-5}$$

where the sb_{-4} is the time domain samples in subframe -4 and k is the total number samples in each subframe.

To detect a release, which can be viewed as a transient in the reversed time direction. Transient release thresholds, ϑ_{rev_low} and ϑ_{rev_high}, are set based on condition $c_{3,adpt-rev}$ below being satisfied.

$$m_{sum} > th_{tonal} * 0.6 \tag{5.2-6}$$

ϑ_{rev_low} and ϑ_{rev_high} are set to 4.5 and 5.5 respectively if condition $c_{3,adpt-rev}$ above is satisfied, otherwise they are set to 4.25 and 5.25.

A transient release in subframe -3 is detected if energy in $E_{TD}(-3)$ is above E_{accRev}(-3) by a factor of ϑ_{rev_low}.

A transient release in subframe -4 is detected if energy in $E_{TD}(-4)$ is above E_{accRev}(-4) by a factor of ϑ_{rev_low} and the $ramp\_up\_flag$ being set or if energy in $E_{TD}(-4)$ is above $E_{accRev}(-4)$ by a factor of ϑ_{rev_high}.

5.2.2.2.7.3 Synchronization of variables for downmix-based channel pairs

For every IVAS mode the transient detector is always run for each channel in a channel pair or single channel element separately computing independent attack and temporal flatness values. However, for modes where the channel pairs are processed with a downmix-based stereo mode (see DFT-based stereo of clause 5.3.2.4 and TD-based stereo of clause 5.3.2.3) the values for both channels are synchronized.

If an attack was detected in one the channels the attack index is set to 1 for both channels. For DFT-based stereo the flags *attackIsPresent* and *wasTransient* – indicating an attack in the current or previous frame, respectively – are updated. *AttackIsPresent* is updated with the synchronized attack index from the transient detection of the current frame, *wasTransient* is set to the value of the previous frame.

The temporal flatness values – computed by equation 39 in clause 5.1.8. of [3] – are also synchronized by setting the values for both channels to the higher value of the two. For DFT-based stereo the temporal flatness value used in the ITD estimation (see clause 5.3.2.4.4) is also updated with this maximum value.

5.2.2.2.8 Linear prediction analysis

The linear prediction (LP) analysis in IVAS reuses the LP analysis from EVS (see clause 5.1.9 of [3]). The positions of LP analysis windows are the same as in EVS which means that the redressed samples from the end of the lookahead are employed in the analysis.

5.2.2.2.9 Open-loop pitch analysis

The open-loop (OL) pitch analysis in IVAS reuses the OL pitch analysis from EVS (see clause 5.1.10 of [3]). The positions of analysis segments are the same as in EVS which means that the redressed samples from the end of the lookahead are employed in the analysis.

The fractional OL pitch estimation as described in clause 5.1.10.9 of [3] is performed for element bitrates lower or equal to 32 kbps (in EVS it is lower or equal to total bitrates of 24.4 kbps).

5.2.2.2.10 Coding mode determination

The LP-based core-coder technology in the IVAS codec operates in one of the following coding modes tailored for a specific class of the input signal:

- Unvoiced Coding (UC) mode
- Voiced Coding (VC) mode
- Transition Coding (TC) mode
- Audio Coding (AC) mode
- Inactive Coding (IC) mode
- Generic Coding (GC) mode

The selection of the coding mode is based on the mechanism from the EVS codec described in clause 5.1.13 in [3].

5.2.2.2.11 Speech/Music classification

5.2.2.2.11.1 Overview

The speech/music classifier (S/M classifier) in the IVAS codec is based on the same technology from the EVS codec, described in clause 5.1.13.6 and 5.1.14.1 of [3]. The IVAS S/M classifier is based either on the Gaussian Mixture Model (GMM) (described in clause 5.2.2.2.11.2) or the SNR-based model (described in clause 5.2.2.2.11.3) depending on the element bitrate of the SCE/CPE as seen in Table 5.2-2 and its output is the probability that the input signal in the current frame is either “speech” or “music”. The output of the IVAS S/M classifier together with some additional parameters is then used for the selection of the IVAS core-coder technology.

Table 5.2-2: S/M classification method based on element type and bitrate

	SCE	**CPE**
GMM-based	< 16 kbps	< 24.4 kbps
SNR-based	≥ 16 kbps	≥ 24.4 kbps

5.2.2.2.11.2 S/M classification based on the GMM model

5.2.2.2.11.2.1 Overview

The GMM model in the IVAS S/M classifier has been extended, improved and optimized for the processing of stereo signals. The GMM model uses the parameters extracted during the pre-processing stage, described in 5.2.2.2, as input features and provides probabilistic estimates for the following three classes: “speech”, “music” and “background noise”. The parameters of the GMM model are trained on a large collection of manually labelled feature vectors. The GMM model provides probabilistic estimates for each of the three classes in every 20-ms frame. Outlier detection logic ensures proper processing of frames where one or more features does not fullfil the condition of normal distribution. The output probabilities are turned into a single unbound score metric by means of logistic regression. The IVAS S/M classifier contains a state machine for the partitioning of the input signal into one of four states. Adaptive smoothing is applied on the output score depending on the current state of the classifier. Fast reaction of the IVAS S/M classifier to events in rapidly varying environments is achieved by onset/attack detection logic based on relative energy. The smoothed score is the basis for the selection among the following three classes: “SPEECH/NOISE”, “MUSIC”, “UNCLEAR”.

The first stage of the IVAS S/M classifier consists of the following functional blocks (listed in logical order):

- State machine for signal partitioning
- Onset/attack detection based on relative frame energy
- Feature extraction
- Outlier detection based on histograms
- Short-term feature vector filtering
- Non-linear feature vector transformation (Box-Cox)
- Principal Component Analysis (PCA)
- Gaussian Mixture Model (GMM)
- Adaptive smoothing
- State-dependent categorical classification

The GMM model is trained using the Estimation-Maximization (EM) algorithm on a large, manually labeled, database.

5.2.2.2.11.2.2 State machine for signal partitioning

First, the input signal is partitioned into four states INACTIVE, ENTRY, ACTIVE and UNSTABLE using the state machine described in clause 5.1.13.6.4 of [3]. The INACTIVE state is selected as the initial state. It is switched to the ENTRY state when the VAD flag is changed from zero to unity. The VAD flag used by the speech/music classifier is extracted from the HE-SAD [3]. The ENTRY state marks the first onset after a longer period of silence frames. After eight frames in the ENTRY state the classifier automatically enters the ACTIVE state which marks the beginning of a stable signal with enough energy. If the energy suddenly drops closer to the level of the background noise the classifier's state is changed to UNSTABLE where it may stay for up to 12 frames. After this period, it returns automatically to the INACTIVE state. If the energy suddenly increases while the classifier is in the UNSTABLE state, the classifier enters the ACTIVE state again. This ensures continuity of classification during short pauses. In the rest of this document the current state of the speech/music classifier is denoted f_{SM}. The constants assigned to the individual states are defined as follows

- INACTIVE: $f_{SM} = -8$
- UNSTABLE: $f_{SM} \in \langle -7, -1 \rangle$
- ENTRY: $f_{SM} \in \langle 0,7 \rangle$
- ACTIVE: $f_{SM} = +8$

Note, that in the INACTIVE and in the ACTIVE state f_{SM} is a single constant whereas in the UNSTABLE and in the ENTRY state f_{SM} takes on multiple values depending on the progression of the state machine. Thus, in the UNSTABLE and in the ENTRY state f_{SM} can be used as a short-term counter.

More details about the signal partitioning scheme can be in clause 5.1.13.6.4 of [3].

5.2.2.2.11.2.3 Onset/attack detection based on relative frame energy

The objective of the onset/attack detection logic inside the S/M classifier is to localize the beginnings of speech utterances and the onsets of musical segments. These events are usually associated with abrupt changes in the characteristics of the input signal. Successful detection of signal onsets and attacks after a longer period of signal inactivity improves the responsivness of the S/M classifier. The onset/attack detection logic plays a similar role as the ENTRY state in the state machine.

The key parameter in the onset/attack detector is the cumulative sum of the differential energy updated in every frame, $v_{run}(n)$. This parameter is initialized to 0 and updated only when the relative energy in the current frame, $E_r(n)$, is greater than the relative frame energy in the previous frame, $E_r(n-1)$. The relative energy in the current frame, $E_r(n)$, is defined in eq. (29) of [3]. The update of $v_{run}(n)$ is done as follows

$$v_{run}(n) = v_{run}(n-1) + (E_r(n) - E_r(n-1)) \quad (5.2\text{-}7)$$

where *n* is the index of the current frame. The cumulative sum of the differential energy, $v_{run}(n)$, is used in updating of the counter of onset/attack frames, v_{cnt}. The counter of onset/attack frames, v_{cnt} is initialized to 0 and incremented by 1 in every ENTRY frame where $v_{run}(n) > 5$. Otherwise, it is reset to 0.

The final output of the attack/onset detector is the binary flag, f_{att}, which is set to 1 when $0 < v_{run}(n) < 3$ and reset to 0 otherwise.

5.2.2.2.11.2.4 Feature extraction

The features used by the GMM model for speech/music/noise classification are listed in Table 5.2-3.

Table 5.2-3: Features used in the GMM model

parameter	format	window length	description	definition
T_{OL}	integer	28.75 ms	open-loop pitch	eq. 317 in clause 5.1.13.6.1 in [3]
r_{voi}	float	28.75 ms	normalized cross-correlation	eq. 293 in clause 5.1.13.1.1 in [3]
LSF	float	25 ms	line-spectral frequencies from the LP analysis	eq. 318 in clause 5.1.13.6.1 in [3]
ϵ_P	float	25 ms	residual energy from the LP analysis (Levinson-Durbin)	eq. 322 in clause 5.1.13.6.1 in [3]
C_{map}	float	20 ms	short-term correlation map	eq. 319 in clause 5.1.13.6.1 in [3]
n_{sta}	float	20 ms	non-stationarity	eq. 320 in clause 5.1.13.6.1 in [3]]
S_{MFCC}	float	20 ms	mel-frequency cepstral coefficients	
P_{diff}	float	20 ms	spectral difference	eq. 324 in clause 5.1.13.6.1 in [3]
P_{sta}	float	20 ms	spectral stationarity	eq. 326 in clause 5.1.13.6.1 in [3]]

Note, that the open-loop pitch T_{OL} and the normalized cross-correlation r_{voi} are averaged over three adjacent half-frames in the total length of 28.75 ms. The LSF vector has *M* values where *M* is the order of the LP analysis. The residual LP energy ϵ_P is a vector of *M*+1 values and it is calculated as part of the Levinson-Durbin algorithm. The short-term correlation map C_{map} is a by-product of the harmonic spectral analysis described in clause 5.1.11.2.5 of [3]. It reflects both, the harmonicity and spectral stability of the input signal. The S_{MFCC} is a vector of mel-frequency cepstral coefficients calculated in $N_{mel} = 40$ spectral bands. The parameters P_{diff} and P_{sta} are the measures of spectral difference and spectral stationarity, respectively.

Let the mel filter bank used in the IVAS S/M classifier de defined as a matrix $\mathbf{F}_{mel}$ of size $N_{mel} \times N_{fft}/2$ where $N_{fft} = 256$ is the length of the FFT transform used in the spectral analysis module, described in clause 5.2.2.2.4. The values of the mel filter bank $\mathbf{F}_{mel}$ are pre-calculated and stored as constants in the memory of the IVAS codec. Let $\mathbf{T}_{DCT}$ be a square matrix of size $N_{mel} \times N_{mel}$ containing the coefficients of the DCT-II transform. The coefficients of the DCT-II matrix $\mathbf{T}_{DCT}$ are calculated as follows

$$T_{DCT}(i,j) = \sqrt{\frac{2}{N_{mel}}}\cos\left(\frac{\pi(2i+1)\cdot j}{2N_{mel}}\right), \quad \text{, for } i,j = 0,\dots,N_{mel}-1 \quad (5.2\text{-}8)$$

where the values in the first row $T_{DCT}(0,j)$ are divided by the factor of $\sqrt{2}$ for all $j = 0,\dots,N_{mel}-1$. The MFCC coefficients are calculated from the power spectrum $PS(k)$, $k = 0,\dots,N_{fft}/2-1$, defined in eq. 28 in clause 5.1.5 of [3]. By re-arranging the values of $PS(k)$ for $k = 0,\dots,N_{fft}/2-1$ in a single column the power spectrum becomes a column vector $\mathbf{S}_p(n)$ where *n* denotes the current frame. The MFCC coefficients are calculated as follows

$$\mathbf{S}_{MFCC}(n) = \mathbf{T}_{DCT}\log\left(\mathbf{F}_{mel}\mathbf{S}_p(n) + 1.0e^{-5}\right) \quad (5.2\text{-}9)$$

Note, that only three MFCC coefficients of the vector $\mathbf{S}_{MFCC}$ are used by the S/M classifier. These are $S_{MFCC}(2)$, $S_{MFCC}(6)$ and $S_{MFCC}(12)$.

The extracted and calculated features are arranged in a column vector $\mathbf{X}(n)$.The total number of features used by the IVAS S/M classifier is $F = 15$.

5.2.2.2.11.2.5 Outlier detection based on feature histograms

The accuracy of the GMM is, to a large extent, affected by the statistical distribution of the individual features. The maximum accuracy is achieved when all individual features are distributed normally, i.e. when $X_j \sim \mathcal{N}(\mu, \sigma)$ for $j = 1, \ldots, 13$. The GMM can, to some extent, model features with non-normal distribution. If the value of one or more features is significantly different than its mean value, then the entire feature vector in the current frame is an outlier. Outliers usually lead to incorrect probability estimates. Instead of discarding the feature vector of an outlier it is better to replace the outlying features with their values from the previous frame or with an average value across multiple previous frames or by some global mean value.

For each feature in the IVAS S/M classifier the lower bound and the upper bound are pre-calculated based on their histogram and stored as constants in the ROM memory. When running the IVAS encoder, the outlier detector compares the values of all features in the current frame against the pre-calculated bounds and marks features lying outside of the range. This can be expressed as

$$f_{odv}(j) = \begin{cases} 0 & \text{if } H_{low} \le X_j(n) \le H_{high} \\ 1 & \text{otherwise} \end{cases} \quad \text{for } j = 1, .., F \tag{5.2-10}$$

Let c_{odv} be the total number of detected outlying features calculated as

$$c_{odv} = \sum_{j=1}^{F} f_{odv}(j) \tag{5.2-11}$$

If the number of outliers is equal to or higher than two, then a binary flag, f_{out}, is set to one. That is

$$f_{out} = \begin{cases} 1 & \text{if } c_{odv} \ge 2 \\ 0 & \text{otherwise} \end{cases} \tag{5.2-12}$$

The flag f_{out} is used for signaling that the entire feature vector is an outlier. If the flag f_{out} is equal to one, then the outlying features are replaced with the values from the previous frame. That is

$$X_j(n) = X_j(n-1) \quad \text{for } j = 1, .., F \quad \text{if } f_{odv}(j) = 1 \tag{5.2-13}$$

where $n - 1$ refers to the previous frame.

5.2.2.2.11.2.6 Short-term feature vector filtering

The speech/music classification accuracy is improved with feature vector smoothing. This is done by applying short-term IIR filter

$$\tilde{X}_j(n) = \alpha_m \tilde{X}_j(n-1) + (1 - \alpha_m X_j(n)) \quad \text{for } j = 1, .., F \tag{5.2-14}$$

where $\alpha_m = 0.5$ is the so-called forgetting factor. The filtered feature vector is initalized with $\tilde{X}_j(0) = 0$. Feature smoothing is not performed in frames in the ENTRY state where $f_{att} = 1$ or $f_{out} = 1$. This is to avoid the smearing effect on strong attacks or outliers at the beginning of active segments where the informative potential of the feature vector in the previous frame is limited. Smoothing is also not performed in the STABLE state of the speech/music classifier where $t_{cnt} = 1$ or $t_{cnt} = 2$. This parameter is set in the transition analysis of the IVAS codec. When short-term feature vector filtering is not performed, the values of the feature vector are simply copied over. That is

$$\tilde{X}_j(n) = X_j(n) \quad \text{for } j = 1, .., F \tag{5.2-15}$$

In the following text, the original symbol for feature values $X_j(n)$ is used instead of $\tilde{X}_j(n)$, i.e. it is assumed that

$$X_j(n) \leftarrow \tilde{X}_j(n) \quad \text{for } j = 1, .., F \tag{5.2-16}$$

5.2.2.2.11.2.7 Non-linear feature vector transformation

The features with non-normal distribution are transformed with a non-linear power transformation [17] to normalize their probability density function. The non-linear transformation is defined as

$$X'_j = \begin{cases} \frac{(X_j + \Delta_j)^{\lambda_j} - 1}{\lambda_j} & \text{if } \lambda_j \ne 0 \\ \log(X_j + \Delta_j) & \text{if } \lambda_j = 0 \end{cases} \quad \text{for } j = 1, .., F \tag{5.2-17}$$

where λ_j is the pre-calculated exponent of the power transform varying from -5 to +5 among the individual features and Δ_j is a small pre-calculated bias ensuring that the input to the $\log(.)$ function is positive.

The feature vector $X_j(n)$ is then replaced by the transformed feature vector using the following relation

$$X_j(n) \leftarrow X'_j(n) \quad \text{for } j = 1,..,F \tag{5.2-18}$$

5.2.2.2.11.2.8 Principal component analysis (PCA)

After short-term smoothing and non-linear transformation the feature vector is standardized by removing its mean and scaling it to unit variance. This is done as follows

$$\acute{X}_j(n) = \frac{X_j(n-1)-\mu_j}{s_j} \quad \text{for } j = 1,..,F \tag{5.2-19}$$

where μ_j is the pre-calculated mean and s_j is the pre-calculated standard deviation of the *j*th feature. The feature vector $X_j(n)$ is then replaced by the standardized feature vector using the following relation

$$X_j(n) \leftarrow \acute{X}_j(n) \quad \text{for } j = 1,..,F \tag{5.2-20}$$

The feature vector is then transformed with the Principal Component Analysis (PCA) where the dimensionality is reduced from 15 to 12. The PCA is an orthogonal transformation converting a set of possibly correlated variables into a set of linearly uncorrelated variables called principal components. In the IVAS speech/music classifier the feature vectors are transformed with the PCA as follows

$$\mathbf{Y}(n) = \mathbf{W}^T\mathbf{X}(n) \tag{5.2-21}$$

where $\mathbf{X}(n)$ is the *F*-dimensional feature vector and $\mathbf{W}$ is the 15×12 matrix of pre-calculated PCA loadings that are stored in the memory of the IVAS codec. The feature vector $\mathbf{X}(n)$ is then replaced by the transformed feature vector $\mathbf{Y}(n)$ using the following relation

$$\mathbf{X}(n) \leftarrow \mathbf{Y}(n) \tag{5.2-22}$$

5.2.2.2.11.2.9 Gaussian mixture model (GMM)

Each component (mixture) in the multivariate GMM model of the IVAS S/M classifier is parameterized by its weight, mean and covariance matrix. Note, that the IVAS S/M classifier uses a separate GMM model for each of the SPEECH, MUSIC and NOISE class. Let the number of components in the GMM model be defined as $K = 6$. Each component in the GMM model has its weight ϕ_k, its mean vector $\boldsymbol{\mu}_k$ and a covariance matrix $\boldsymbol{\Sigma}_k$. The sum of component weights must fullfill the following condition

$$\sum_{k=0}^{K-1} \phi_k = 1 \tag{5.2-23}$$

such that the probability distribution function of the GMM is normalized. Based on the established notation the probability that a given feature vector $\mathbf{X}(n)$ has been generated by the GMM model can then be expressed as

$$p(\mathbf{X}(n)) = \sum_{k=0}^{K-1} \phi_k \frac{1}{\sqrt{(2\pi)^K|\boldsymbol{\Sigma}_k|}} \exp\left(-\frac{1}{2}(\mathbf{X}(n)-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{X}(n)-\boldsymbol{\mu}_k)\right) \tag{5.2-24}$$

The parameters of the GMM model (mean vector, weights and covariance matrix) are pre-calculated using the Expectation-Maximization (EM) algorithm. In the above formula, it's necessary to calculate the exponential function which is a complex operation. However, by taking a logarithm of the whole term inside the sum of eq. (5.2-24) it is possible to calculate the "score" and use it instead of the probablity function. That is

$$p_{scr}(\mathbf{X}(n)) = \sum_{k=1}^{K} \log(\phi_k) - \frac{1}{2}\log((2\pi)^K|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{X}(n)-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{X}(n)-\boldsymbol{\mu}_k) \tag{5.2-25}$$

The score p_{scr} is an unbounded variable proportional to the probability. The higher the score, the higher the probability that a given feature vector is generated by the GMM model. The score is calculated in the IVAS S/M classifier for all three GMM models. The score of the "speech" GMM model and the score of the "music" GMM model are then combined into a single value by calculating their difference. That is

$$dlp(\mathbf{X}(n), b_s) = p_{scr,M}(\mathbf{X}(n)) - p_{scr,S}(\mathbf{X}(n)) + b_S \tag{5.2-26}$$

where $p_{scr,M}$ is the score of the "music" GMM model, $p_{scr,S}$ is the score of the "speech" GMM model and b_s is the pre-calculated decision bias. Negative values of the differential score indicate that the input signal is closer to speech whereas positive values indicate that the input signal is closer to music.

The differential score $dlp(\mathbf{X}(n), b_s)$, calculated with the optimal value of b_s, is then limited to be within the range of (-30.0, +30.0). The differential score is reset to 0 when the VAD flag, f_{VAD}, or when the total frame energy, E_{tot}, is lower than 10 dB or when the S/M classifier is in the ENTRY state and either f_{att} or f_{out} are 1. See clause 5.2.2.2.11.2.2 for details. Note, that the calculation of the VAD flag is described in detail in clause 5.1.12 of [3] where it is denoted as f_{SAD} and referred to as the SAD decision. The total frame energy E_{tot} is defined in eq. (27) of [3].

For simplicity of notation the notation of the score will be simplified to $dlp(n)$ in the following clauses.

5.2.2.2.11.2.10 Adaptive smoothing

The differential score $dlp(n)$ is smoothed with an adaptive IIR filter. The filtering operation can be described as

$$wdlp(n) = wght(n) \cdot wdlp(n-1) + (1 - wght(n)) \cdot dlp(n) \tag{5.2-27}$$

where $wght$ is the so-called forgetting factor of the IIR filter. The forgetting factor is a product of three individual weights

$$wght(n) = wrelE(n) \cdot wdrop(n) \cdot wrise(n) \tag{5.2-28}$$

The first weight, *wrelE*, is linearly proportional to the relative frame energy, $E_r(n)$, defined in eq. (29) in [3], and is calculated as

$$wrelE(n) = 0.99 + (E_r(n) + 15)\frac{0.9 - 0.99}{30} \tag{5.2-29}$$

The first weight *wrelE* is limited to the interval (0.9, 0.99). It reaches the upper threshold of 0.99 when the relative energy $E_r(n)$ is higher than 15 dB. Similarly, it reaches the lower threshold of 0.9 when the relative energy $E_r(n)$ is lower than -15 dB. The value of *wrelE* strongly influences the forgetting factor of the IIR filter. Smoothing is stronger in energetically weak segments where it is expected that the features carry less relevant information about the input signal.

The second weight, *wdrop*, is proportional to the derivative of *dlp*. First, a short-term mean of *dlp* is calculated as

$$dlp_{ST}(n) = 0.8 \cdot dlp_{ST}(n-1) + 0.2 \cdot dlp(n) \tag{5.2-30}$$

The second weight *wdrop* is set to 0 and modified only in frames where the following two conditions are met:

$$\begin{aligned} dlp(n) &< 0 \\ dlp(n) &< dlp_{ST}(n) \end{aligned} \tag{5.2-31}$$

Thus, *wdrop* is updated only when the differential score $dlp(n)$ indicates that the input signal looks like a speech signal and the tendency of the differential score is descreasing. In the first frame, when the two conditions in eq. (5.2-31) are met, and if $dlp_{ST}(n) > 0$, *wdrop* is set to

$$wdrop(n) = -dlp(n) \tag{5.2-32}$$

Otherwise, *wdrop* is steadily increased as follows

$$wdrop(n) = wdrop(n-1) + (dlp_{ST}(n-1) - dlp(n)) \tag{5.2-33}$$

If the two conditions in eq. (5.2-31) are not fullfilled, *wdrop* is reset to 0. Thus, the parameter *wdrop* reacts to the sudden drops of the differential score *dlp* below the zero-level indicating potential speech onset. The final value of the weight *wdrop* is linearly mapped to the interval (0.7, 1.0) using the following relation

$$wdrop(n) \leftarrow 1.0 + wdrop(n)\frac{0.7 - 1.0}{15} \tag{5.2-34}$$

Note that the value of $wdrop$ is "overwritten" in the formula above.

The third weight, *wrise*, is calculated similarly as *wdrop* with the only difference that it reacts to the sudden rises of *dlp* indicating potential music onsets. The parameter *wrise* is initialized to 0 and modified only in frames where the following conditions are all met:

$$\begin{gathered} f_{SM}(n) = 8 \quad \text{(ACTIVE)} \\ dlp_{ST}(n) > 0 \\ dlp_{ST}(n) > dlp_{ST}(n-1) \end{gathered} \tag{5.2-35}$$

Thus, *wrise* is updated only when the S/M classifier is in the ACTIVE state and the differential score has increasing tendency and when it indicates that the current frame belongs to the MUSIC class. In the first frame, when all the above-specified conditions are met, and if $dlp_{ST}(n-1) < 0$, *wrise* is set to

$$wrise(n) = -dlp_{ST}(n) \tag{5.2-36}$$

Otherwise, it is steadily increased as follows

$$wrise(n) = wrise(n-1) + (dlp_{ST}(n) - dlp_{ST}(n-1)) \tag{5.2-37}$$

If the conditions in eq. (5.2-35) are not true, *wrise* is reset to 0. Thus, the parameter *wrise* reacts to the sudden rises of *dlp* above the zero-level indicating potential music onset. The final value of this parameter is linearly mapped to the interval (0.95, 1.0) using the following relation

$$wrise(n) \leftarrow 1.0 + wrise(n)\frac{0.95 - 1.0}{5}$$

Note that the value of $wrise$ is "overwritten" in the formula above.

Furthremore, the IIR forgetting factor, $wght(n)$, is decreased when the S/M classifier indicates strong SPEECH content or strong MUSIC content. This is done by analyzing the mean and the variance of the differential score $dlp(n)$ that are calculated by means of long-term IIR filters. That is

$$\begin{gathered} \bar{\mu}_{dlp}(n) = 0.9 \cdot \bar{\mu}_{dlp}(n-1) + 0.1 \cdot dlp(n) \\ \bar{\sigma}_{dlp}(n) = 0.9 \cdot \bar{\sigma}_{dlp}(n-1) + 0.1 \cdot (dlp(n) - \bar{\mu}_{dlp}(n))^2 \end{gathered} \tag{5.2-38}$$

In the ENTRY state, the variables are reset with $\bar{\mu}_{dlp}(n) = dlp(n)$ and $\bar{\sigma}_{dlp}(n) = 0$. When the absolute value of the differential score is high, with low variations around its mean value, there is a good chance that the S/M classifier has enough certainty about its content. This is can be expressed by the following mean to variance ratio

$$r_{m2v}(n) = \frac{|\bar{\mu}_{dlp}(n)|}{\sqrt{|\bar{\sigma}_{dlp}(n)|}} \tag{5.2-39}$$

Note that the expression in the denominator is the long-term standard deviation of the differential score. The weight $wght(n)$ is decreased in frames where $r_{m2v}(n) > 15$ as follows

$$wght(n) \leftarrow 0.9 \cdot wght(n) \tag{5.2-40}$$

The final value of the weight $wght(n)$ is limited to the range (0.01, 1.0). In all frames where the total frame energy, $E_{tot}(n)$, is below 10 dB the weight $wght(n)$ is set to 0.92. This ensures proper smoothing of the differential score during silence periods.

The filtered differential score, $wdlp(n)$, is the fundamental parameter for the S/M classifier to generate a categorical output. This is described below.

5.2.2.2.11.2.11 State-dependent categorical classification

The final step in the IVAS S/M classifier is the decision to which of the following three categories (classes) the input signal in the current frame belngs to:

- SPEECH/NOISE (0)
- UNCLEAR (1)
- MUSIC (2)

The numbers in parentheses are the numeric constants associated with the three output classes. Note, that the above set of classes is different than the SPEECH/MUSIC/NOISE classes that have been discussed so far in relation to the GMM model and the differential score. The first difference is that the SPEECH class and the NOISE class are combined together. This is a preparatory step before the IVAScore-coder technology selection in which ACELP core is selected

for the encoding of both, speech signals as well as background noise. The UNCLEAR class has been added to represent mixed speech signals with high level of background music. A typical differential score $wdlp(n)$ of an input signal in the UNCLEAR is close to 0. Fig. 5.2-1 shows the histograms of the differential scores $wdlp(n)$, calculated on a large set of testing signals, and the transitions among output categories of the IVAS S/M classifier.

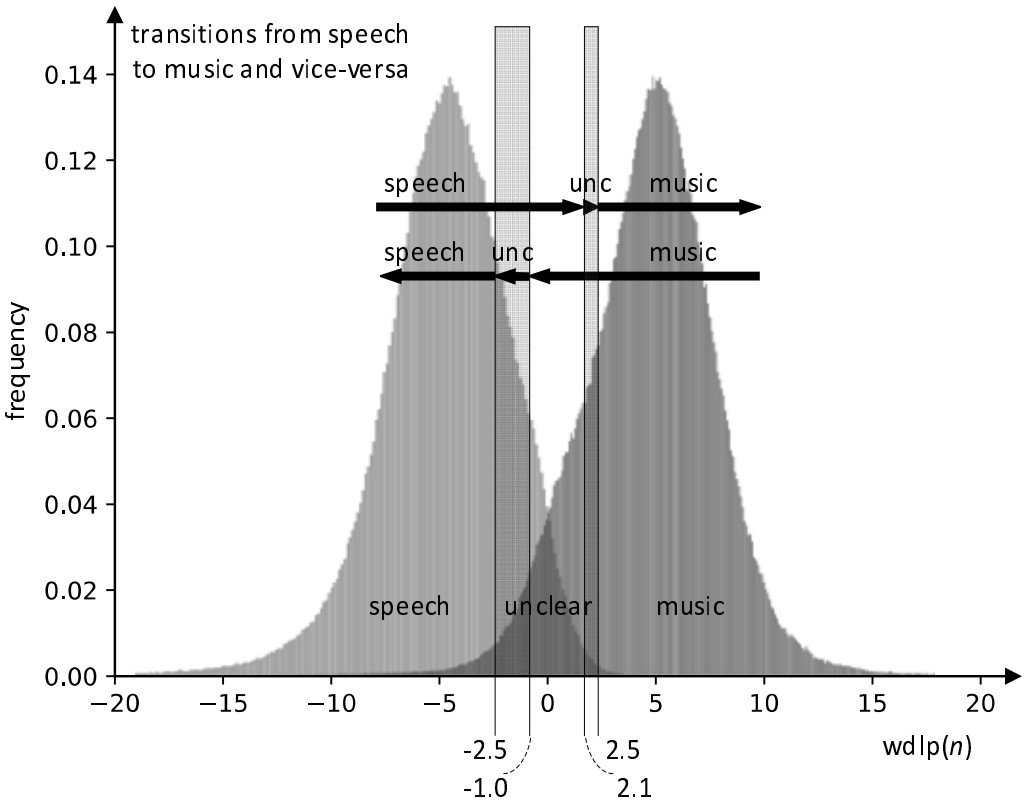


Figure 5.2-1: Transitions between output classes of the IVAS speech/music classifier

Let d_{SMC} denote the final output class selected in the first stage of the IVAS speech/music classifier. When the S/M classifier in the current frame is in the ENTRY state the final output class is selected based on the weighted average of previous differential scores of the ENTRY period. The weighted average of previous differential scores is calculated as follows

$$wdlp(n) \leftarrow \sum_{k=0}^{n-n_{ENTRY}} \alpha_k(n - n_{ENTRY}) \cdot dlp(n-k) \quad \text{for } n = n_{ENTRY}, .., n_{ENTRY} + 7 \qquad (5.2\text{-}41)$$

where n is the current frame, n_{ENTRY} is the frame index corresponding to the beginning of the ENTRY period and $\alpha_k(n - n_{ENTRY})$ are the weights associated with the frames inside the ENTRY period. Thus, the number of samples used in the sum in eq. (5.2-41) ranges from 0 to 7 depending on teh value of n_{ENTRY}. Note, that n_{ENTRY} is essentially the duration of the ENTRY period. The calculation of the weighted average of differential scores is illustrated in Fig. 5.2-2.

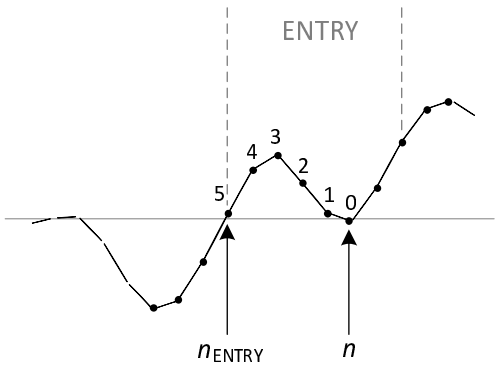


Figure 5.2-2: Calculating weighted average of differential scores in the ENTRY period

The weights are sorted in decreasing order such that the most recent frame is associated with the highest weight. See Table 5.2-4 below.

Table 5.2-4: Weights used in the calculation of the weighted average of differential scores in the ENTRY period

$n - n_{ENTRY}$	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7
0	1	0	0	0	0	0	0	0
1	0.6	0.4	0	0	0	0	0	0
2	0.47	0.33	0.2	0	0	0	0	0
3	0.4	0.3	0.2	0.1	0	0	0	0
4	0.3	0.25	0.2	0.15	0.1	0	0	0
5	0.233	0.207	0.18	0.153	0.127	0.1	0	0
6	0.235	0.205	0.174	0.143	0.112	0.081	0.05	0
7	0.2	0.179	0.157	0.136	0.114	0.093	0.071	0.05

If the absolute frame energy E_{tot}, defined in eq. (27) in [3], is lower than 10 dB the final output class is automatically set to SPEECH/NOISE regardless of the weighted differential score $wdlp(n)$. That is

$$d_{SMC} = 0 \quad \text{if } E_{tot} < 10 \tag{5.2-42}$$

This is to avoid misclassifications during silence periods.

If the IVAS S/M classifier is in the ENTRY state with $f_{SM} \in \langle 0,7 \rangle$ the final output class is selected based on the weighted differential score $wdlp(n)$. This is done as follows

$$d_{SMC} = \begin{cases} 0 & \text{if } wdlp(n) \leq 2 \\ 2 & \text{if } wdlp(n) > 2 \text{ AND } dlp(n) > 2 \\ 1 & \text{otherwise} \end{cases} \tag{5.2-43}$$

In all other states of the IVAS S/M classifier the selection of the final output class is based on the weighted differential score $wdlp(n)$, taking into account the final output class selected in the previous frames. This is done as follows. The final output class is first initialized to the value from the previous frame. That is

$$d_{SMC}(n) = d_{SMC}(n-1) \tag{5.2-44}$$

where the index *n* denotes the current frame and the index $n-1$ denotes the previous frame. The final output class is modified if the value of weighted differential score $wdlp(n)$ gets beyond the limits defined by the lower and the upper threshold for the output class selected in the previous frame. The transitions between the output classes are shown in Fig. 5.2-3.

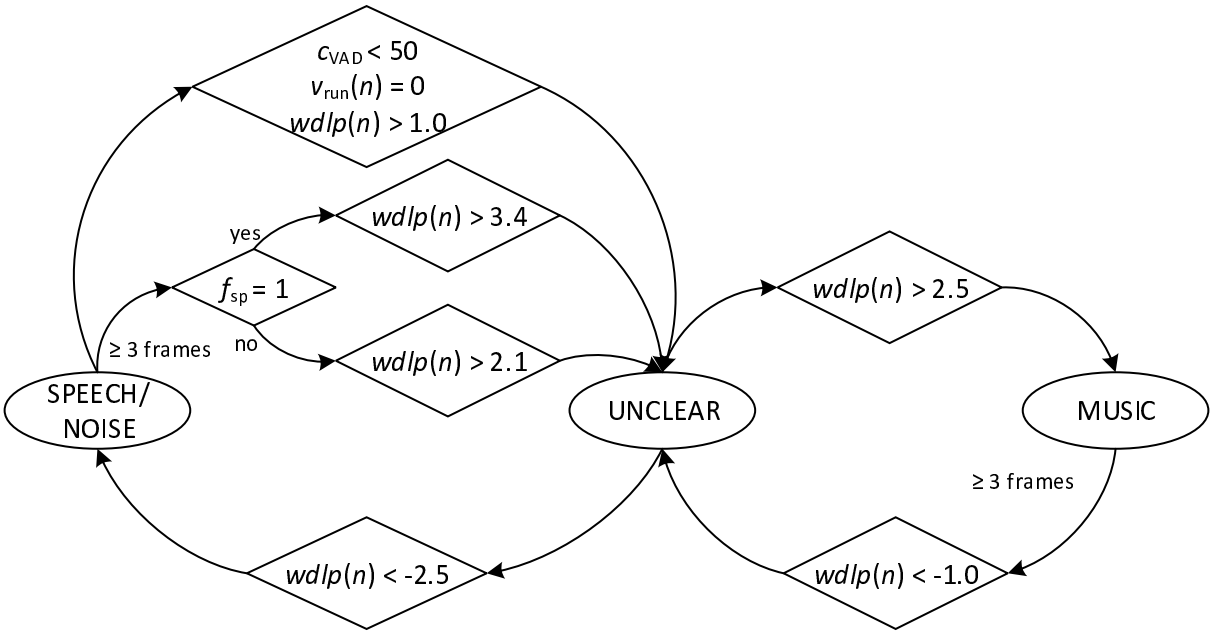


Figure 5.2-3: Final class selection in the IVAS S/M classifier

For example, if the final output class selected in the previous frame is SPEECH/NOISE and the weighted differential score $wdlp(n)$ in the current frame is higher than 1.0 the final output class is changed to UNCLEAR. The graph in 5.2-1 shows the histograms of $wdlp(n)$ for the SPEECH/NOISE class and the MUSIC class obtained by running the IVAS S/M classifier on a large set of testing signals. There are two pairs of thresholds. The first pair of thresholds (2.1; 2.5) is applied for SPEECH/NOISE->UNCLEAR->MUSIC transition. The second pair of thresholds (-1.0; -2.5) is applied for MUSIC->UNCLEAR-> SPEECH/NOISE transition. Note, that it is not possible to switch the final output class directly from SPEECH/NOISE to MUSIC and vice-versa. The values of the thresholds are slightly more favorable for the SPEECH class. This introduces a small bias to the IVAS S/M classifier. The transitions between classes and the associated thresholds are summarized in Table 5.2-5.

Table 5.2-5: Thresholds for output class transitions

		TO		
		SPEECH/NOISE	**UNCLEAR**	**MUSIC**
FROM	**SPEECH/NOISE**	--	>2.1	
	UNCLEAR	<-2.5	--	>2.5
	MUSIC		<-1.0	--

The complete set of rules for transitions between the output classes is shown in the schematic diagram in Fig. 5.2-3. The arrows indicate the direction in which an output class may be changed if the condition inside the rhomb is satisfied. In case of multiple conditions, logical AND is assumed between them, i.e. all conditions must be fulfilled. If an arrow is conditioned by the notation "≥ N frames", it means that the output class may be changed after, at least, N frames. This adds a short hysteresis to some of the class transitions. The parameter f_{sp} is the short pitch flag, defined in eq. (98) in clause 5.1.10.8 of [3]. The the short pitch flag is a by-product of the open-loop pitch analysis, described in clause 5.1.10 of [3]. The parameter c_{VAD} is the counter of active frames. The parameter c_{VAD} is initialized to zero and updated in every frame as follows

$$c_{VAD} \leftarrow \begin{cases} 0 & \text{if } f_{VAD} = 0 \\ c_{VAD} + 1 & \text{if } f_{VAD} = 1 \text{ AND } c_{VAD} < 50 \end{cases} \quad (5.2\text{-}45)$$

5.2.2.2.11.3 SNR-based speech/music classifier

For CPEs starting from an element bitrate of 24.4kbps, as well as for SCEs starting from and element bitrate of 16kbps the first-stage speech/music classification output d_{SMC} is based on the SNR-based speech/music classifier.

The SNR-based speech/music classifier in IVAS is mainly the same as for EVS and described in clause 5.1.14.1 of [3]. The major difference compared to EVS is that the for almost all cases the TCX and ACELP SNR estimates are computed using the 12.8 kHz sampling rate input signals even if the core sampling rate for coding is 16 kHz. The only

exception is for SBA input format using the SCE and for bitrates 24.4 kbps and 32 kbps for which the *flag_16k_smc* is set to 1. For this case, the required input buffers for estimating the TCX and ACELP SNRs are resampled to 16 kHz sampling rate.

5.2.2.2.11.4 Correction for non-harmonic transient signals

For low bitrates a determination is made to identify if the input signal contains a problematic transient (attack or release) based on forward and reverse time direction signal analysis to adjust the encoding scheme selection through frame classification. The problematic transient cause annoying artifacts when encoded in the FD domain and thus a correction of the frame classification is performed. Frames classified as speech are encoded with a TD domain encoding scheme and music classified frames are encoded with a FD domain encoding scheme. A transient attack or a transient release is detected in the input signal, the location in the current and previous frame is used to adaptively adjust a frame classification. Additionally, an analysis of the harmonicity of the input signal is performed to adjust the frame classification. Finally, three conditions, (c_1, c_2, c_3), are evaluated to get a decision on whether to override the frame classification as speech.

The first condition, c_1, is whether a transient is detected in the current frame, F_N, excluding the last subframe. The second condition, c_2, to be checked especially when the last frame was a TD frame is whether a transient is detected in the last half of the previous frame. At low rates, both conditions are checked by performing a refined transient analysis in both the forward and reversed time direction as described in clause 5.1.2.6.1. The third condition, c_3, is whether the signal is harmonic which is determined by a harmonicity flag set according to equation (136) in [3]. The decision to change the frame classification to speech is determined by the flag below.

$$forceTD = (c_1 \mid c_2) \,\&\, !\, c_3 \tag{5.2-46}$$

The third condition c_3 not being fulfilled (false), indicates the signal is not harmonic. When the flag $forceTD$ is set, the frame is classified as speech, otherwise the frame classification is left unchanged to that determined by the IVAS S/M classifier. The first condition, c_1, addresses both forward and backward spreading (and smearing) caused by the scarcity of bits in the low-rate FD TCX20 compression scheme, where TCX20 is a regular MDCT frame type producing 20 ms of synthesized output signal. The second condition, c_2, addresses smearing caused by the suboptimal transition window (TCX25) used when switching from TD to FD coding. If there is a strong transient in the end of the TD coded frame, part of its energy might be included in the beginning of the FD coded frame, which then causes smearing. The third condition, c_3, is restricting the switch to TD for signals with high harmonicity when the low-rate TD coding mode is likely not performing as well as the FD coding.

First, condition c_3 is evaluated by checking harmonicity of the signal based on the long-term correlation map, m_{sum}, and an adaptive threshold, th_{tonal}, where c_3 is denoted p_{tonal} in [3]. If the harmonicity flag is set, meaning there is a high degree of harmonicity in the signal, the classification is not changed with respect to potential transients and further analysis is not carried out. However, if the harmonicity flag is not set, a transient analysis is performed to determine whether to classify the frame as speech.

The subframes analysed for transients are the ones that fall within the transform window that would be encoded if an FD encoding scheme would be used. This corresponds to subframes {-4, 6}. For subframes {-2, 6} a refined transient detector described in clause 5.1.2.1.6.1 is run in the forward direction. If a transient is detected, which corresponds to c_1 in equation (5.2-46) being fulfilled, the $forceTD$ flag is set.

For subframes {-3} and {-4} an enhanced transient detector is run in the reverse time direction as described in clause 5.1.2.1.6.1 to detect a potentially harmful transient release whose energy might spread too much into the current frame to be encoded.

For the reverse analysis of subframe {-3} it is checked whether the transient release energy is above ϑ_{rev_low}, and if that is the case, $forceTD$ is set, adjusting the coding scheme to be TD as using an FD encoding scheme might lead to smearing. For the reverse analysis of subframe {-4} it is checked whether a transient release is detected with both thresholds ϑ_{rev_low} and ϑ_{rev_high}. If a transient release is detected with threshold ϑ_{rev_high}, the $forceTD$ flag is set. If a transient release is detected with only threshold ϑ_{rev_low}, it is additionally checked whether the energy of the second half is greater than the first half of subframe -4, and if that is the case, the $forceTD$ flag is set. The reason for the additional high resolution time domain analysis within subframe {-4} is that: only a part of the 1st half of the subframe actually falls within the TCX25 transform window and that the signal is weighted by the TCX25 window, so if most energy is located in the 2nd part of subframe {-4} then we might get smearing even though we are lower than limit ϑ_{rev_high}. In all other cases $forceTD$ flag is not set.

The status of the flag $forceTD$ is used to override the output class of the IVAS S/M classifier, d_{SMC}, described in clause 5.2.2.2.11.2.11. The output class d_{SMC} may be modified only when it was previously set to "MUSIC", i.e. when $d_{SMC} = 2$. The output class d_{SMC} is set to zero ("SPEECH") as follows:

In case of SCE processing

$$d_{SMC} = 0 \quad \text{if } brate_{element} < 16000 \tag{5.2-47}$$

In the DFT stereo mode

$$d_{SMC} = 0 \quad \text{if } brate_{element} \leq 16400 \tag{5.2-48}$$

for both the first and the second element of the CPE.

5.2.2.2.12 Core-coder technology pre-selection

5.2.2.2.12.1 Overview

The core-coder technologies employed in the IVAS codec are based on identically-named coder technologies from the EVS codec. The details of coder technologies in the IVAS codec are provided in clauses 5.2.2.3.2 and 5.2.2.3.3. As part of the front pre-processing module the IVAS codec pre-selects the core-coder technology from the following list:

- ACELP core-coder technology
- GSC core-coder technology
- MDCT core-coder technology

The pre-selection of the core-coder technology is based on the mechanism described in clause 5.1.13.6.6 of [3] as the "Second stage of the S/M classifier". The core-coder technology pre-selection in the IVAS codec is based on the output class of the IVAS S/M classifier $d_{SMC}(n)$, described in clause 5.2.2.2.11.2.11, and several auxiliary parameters such as long-term signal stability, attack detection, signal tonality and other auxiliary parameters. These are described in detail in the following clauses.

Note, that the ultimate selection of the core-coder technology (described later in clause 5.2.2.3.1.2) follows the pre-selected core-coder technology, taking into account bitrate limitations, bandwidth limitations, VAD flag and other auxiliary parameters. The ultimate selection of the core-coder technology is described in detail in clause 5.2.2.3.1.2.

5.2.2.2.12.2 Long-term signal stability estimation

Long-term signal stability is an important parameter for successful discrimination between vocal music and opera. In the context of the core-coder technology pre-selection, signal stability is understood as long-term stationarity of an input signal with high autocorrelation. The estimation of long-term signal stability is based on the *voicing* parameter, calculated as a by-product in the open-loop pitch analysis of the IVAS codec, described in clause 5.1.10 of [3]. Note, that the *voicing* parameter is defined as the maximum value of the normalized autocorrelation function in the current frame. As signal stability is closely related to the fluctuations of energy, it's beneficial to express the variance of the *voicing* parameter. First, the mean of the variance is caculated as follows

$$\overline{voicing} = \frac{1}{10}\sum_{k=1}^{10} voicing(n-k) \tag{5.2-49}$$

and the variance is calculated as

$$cor_{var}(n) = \frac{1}{10}\sum_{k=1}^{10}(voicing(n-k) - \overline{voicing})^2 \tag{5.2-50}$$

The parameter $cor_{var}(n)$ is also referred to as the "short-term" variance of the *voicing* parameter. The variance of the *voicing* parameter is then smoothed with an IIR filter according to the following formula

$$cor_{LT}(n) = 0.9 \cdot cor_{LT}(n-1) + 0.1 \cdot voicing(n) \tag{5.2-51}$$

and $cor_{LT}(n)$ is referred to as the "long-term" variance of the *voicing* parameter. If the long-term variance of the *voicing* parameter is high and the short-term variance of the *voicing* parameter is low the input signal is considered as stable for the purposes of core-coder technology pre-selection. This is measured by comparing the above values to pre-calculated thresholds, i.e.

$$f_{STAB}(n) = \begin{cases} 1 & \text{if } cor_{var}(n) < 0.0005 \text{ and } cor_{LT}(n) > 0.8 \\ 0 & \text{otherwise} \end{cases} \quad (5.2\text{-}52)$$

Thus, the binary flag, $f_{STAB}(n)$, is an indicator of signal stability and it is used in the core-coder technology pre-selection discussed later in the section.

5.2.2.2.12.3 Segmental attack detection and coding

5.2.2.2.12.3.1 Overview

The speech/music classifier contains an onset detector as described in clause 5.1.13.6.6 of [3]. If the onset is detected in the last subframe of the current frame, the ACELP core coding mode is set to the transition coding (TC) mode and its glottal-shape codebook is assigned to the last subframe of this frame. In IVAS, this concept is extended to improve the coding efficiency of frames including onsets, attacks, and strong transitions, further referred for simplicity as attacks.

In IVAS, the attack detector from EVS, considered as a first stage, is complemented with a second-stage logic to not only detect a larger number of frames including an attack but also, upon coding of such frames, to force the use of the TC coding mode and corresponding glottal-shape codebook in all subframes in which an attack is detected. The block diagram of the two-stage attack detector used in IVAS is shown in figure 5.2-4.

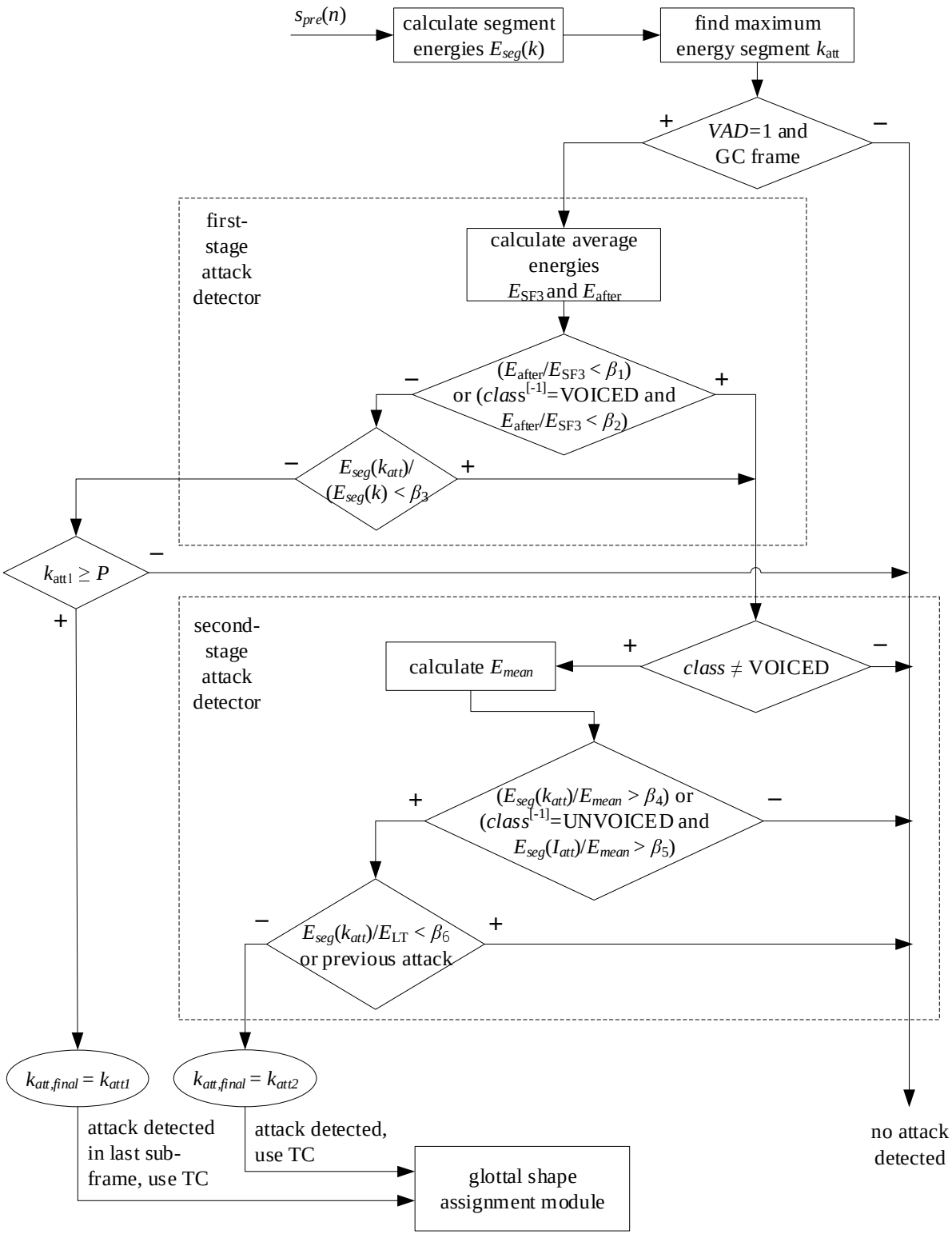


Figure 5.2-4: Block diagram of the two-stage attack detector

5.2.2.2.12.3.2 Energy analysis in segments

As shown in figure 5.2-4, the detection of attacks starts with a preprocessing where energies in several segments of the input signal in the current frame are calculated, followed by a detection performed sequentially in two stages and by a final decision. The first-stage detection is based on comparing calculated energies in the current frame while the second-stage detection takes into account also past frame energy values.

First, an energy in successive energy segments of the perceptually weighted input signal, $s_{pre}(n)$, $n = 0, \dots, L-1$, where L is the length of the frame at internal sampling rate in samples, is calculated using equation (365) from [3] and denoted as $E_{seg}(k)$ where $k, k = 0, \dots, L/K - 1$, is the segment number and $K = 8$ is the analysis segment length in samples. Thus, the total number of energy analysis segments is L/K =32 in IVAS front pre-processing at internal sampling rate of 12.8 kHz (i.e. $L = 256$).

Next, the segment with the maximum energy is found using equation (366) from [3] and denoted as k_{att}.

The detection then continues when the VAD fag is 1 and the originally selected coding mode is generic coding (GC).

5.2.2.2.12.3.3 First-stage attack detection

The first-stage attack detector comprises a calculation of average energies E_{SF3} and E_{after} computed following equations (367) resp. (368) from [3]. The ratio of these energies is then compared to thresholds $\beta_1 = 8$ and $\beta_2 = 20$ as shown in and after equation (369) from [3].

Next, the ratio between the segment with maximum energy $E_{seg}(k_{att})$ and with energy $E_{seg}(k)$ of the other analysis segments of the current frame is compared against the threshold $\beta_3 = 1.3$ as defined in equation (370) from [3]. If the result from this comparison is false and $k_{att} \geq P$, the attack is detected in the last subframe in segment k_{att1} where $P = 24$ is the number of segments before the last subframe. Otherwise, the attack is not detected in the first-stage attack detector and k_{att1} is set to 0.

5.2.2.2.12.3.4 Second-stage attack detection

If the attack is not detected in the last subframe, the detection continues with the second-stage attack detector. The second-stage attack detection starts with comparing whether the signal class (defined in clause 5.1.13.3.1 of [3]) in the current frame is VOICED. If so, the decision module outputs the decision that no attack is detected. If an attack was not detected in the first-stage attack detector, i.e. $k_{att1} = 0$, and the signal class of the current frame is other than VOICED, then the second-stage attack detector is further applied.

The second-stage attack detector then calculates a mean energy across $L/K = 32$ analysis segments before the candidate attack k_{att} – including segments from the previous frame – as follows:

$$E_{mean} = \frac{1}{L/K}\left(\sum_{k=k_{att}}^{L/K-1} E_{seg}^{[-1]}(k) + \sum_{k=0}^{k_{att}-1} E_{seg}(k)\right) \tag{5.2-53}$$

where $E_{seg}^{[-1]}(k)$ are energies per segments from the previous frame.

The mean energy from equation (5.2-53) is then subject to a logic decision as follows:

$$\begin{aligned} &\text{if } \left\{\left(\frac{E_{seg}(k_{att})}{E_{mean}} > \beta_4\right) \text{ OR } \left(\left(\frac{E_{seg}(k_{att})}{E_{mean}} > \beta_5\right) \text{ AND } \left(class^{[-1]} == \text{UNVOICED}\right)\right)\right\} \\ &\text{then } k_{att2} = k_{att} \\ &\text{otherwise } k_{att2} = 0 \end{aligned} \tag{5.2-54}$$

where k_{att} was found in Equation (366) of [3], $\beta_4 = 16$ and $\beta_5 = 12$ are attack detection thresholds, and $class^{[-1]}$ is the signal class from previous frame. When the comparison from (5.2-54) determines that $k_{att2} = 0$, no attack is detected.

In order to further reduce the number of falsely detected attacks when $k_{att2} > 0$, an energy ratio is compared to a certain threshold as follows:

$$\text{if } \left\{\frac{E_{seg}(k_{att})}{E_{LT}} < \beta_6\right\} \text{ then } k_{att2} = 0 \tag{5.2-55}$$

where the threshold $\beta_6 = 20$, and E_{LT} is a long-term energy computed using:

$$E_{LT} = \alpha \cdot E_{LT}^{[-1]} + (1-\alpha) \cdot \frac{1}{L/K}\sum_{k=0}^{L/K-1} E_{seg}(k) \tag{5.2-56}$$

where the parameter $\alpha = 0.95$ and $E_{LT}^{[-1]}$ being the long-term energy from the previous frame. Again, when $k_{att2} = 0$, no attack is detected.

Finally, the energy comparator sets the attack position to $k_{att2} = 0$ if an attack was detected in the previous frame. In this case no attack is detected.

5.2.2.2.12.3.5 Final attack detection decision

A final decision whether the current frame is determined as an attack frame to be coded using the TC coding mode is conducted based on the positions of the attacks k_{att1} and k_{att2} obtained in the first-stage attack detector from clause 5.2.2.2.12.3.3 and the second-stage attack detector from clause 5.2.2.2.12.3.4, respectively.

If the current frame is active ($VAD = 1$) and previously classified for coding in the GC coding mode, the following logic is applied:

$$\begin{aligned} &\text{if } (k_{att1} \geq P) \\ &\quad \text{then } k_{att,final} = k_{att1} \\ &\text{else if } k_{att2} > 0 \\ &\quad \text{then } k_{att,final} = k_{att2} \end{aligned} \tag{5.2-57}$$

Specifically, the attack detector determines in the first-stage if $k_{att1} \geq P$. If so, then k_{att1} is the position of the detected attack, $k_{att,final}$, in the last subframe of the current frame and is used to determine in the glottal-shape assignment module that the glottal-shape codebook of the TC coding mode is used in this last subframe. Otherwise, no attack is detected.

Concerning the second-stage of the attack detection, if the comparison of equation (5.2-55) is true or if an attack was detected in the previous frame, then $k_{att2} = 0$ and no attack is detected. Otherwise, an attack is detected in the current frame at position $k_{att,final} = k_{att2}$. The position of the detected attack, $k_{att,final}$, is used to determine in the glottal-shape assignment module in which subframe the glottal-shape codebook of the TC coding mode is used.

The information about the final position $k_{att,final}$ of the detected attack is used to determine in which subframe of the current frame the glottal-shape codebook within the TC coding mode is employed and which TC mode configuration is used (see clause 5.2.3.2.2 from [3] for TC mode configuration details). Specifically, in case of a coded frame of $L = 256$ samples which is divided into four (4) subframes and L/K =32 energy analysis segments, the glottal-shape codebook is used in the first subframe if the final attack position $k_{att,final}$ is located in segments 1 – 7, in the second subframe if the final attack position $k_{att,final}$ is located in segments 8 – 15, in the third subframe if the final attack position $k_{att,final}$ is located in segments 16 – 23, and finally in the last (fourth) subframe of the current frame if the final attack position $k_{att,final}$ is located in segments 24 – 31. The value $k_{att,final} = 0$ signals that an attack was not found and that the current frame is coded according to the original coder type classification (usually GC coding mode).

In case of a frame of $L = 320$ samples which is divided into five (5) subframes, the L/K =32 energy analysis segments are translated such that the glottal-shape codebook is used in the first subframe if the final attack position $k_{att,final}$ is located in segments 1 – 6, in the second subframe if the final attack position $k_{att,final}$ is located in segments 7 – 12, in the third subframe if the final attack position $k_{att,final}$ is located in segments 13 – 19, in the fourth subframe of the current frame if the final attack position $k_{att,final}$ is located in segments 20 – 25, and finally in the last (fifth) subframe of the current frame if the final attack position $k_{att,final}$ is located in segments 26 – 31.

5.2.2.2.12.4 Signal tonality estimation

In the context of core-coder technology pre-selection signal tonality is a binary flag reflecting both, spectral stability and harmonicity, in the lower frequency range of the input signal up to 4 kHz. Signal tonality is calculated based on the correlation map, $S_{map}(k), k = 0, \ldots, 79$. The correlation map is a by-product of the tonal stability estimation, described in detail in clause 5.1.11.2.5 of [3]. The correlation map is denoted as $M_{cor}(k)$ in eq. (134) of [3]. The correlation map is smoothed with an IIR filter and summed across all bins in the frequency range to yield a single number. That is

$$\overline{S_{map}}(n,k) = \beta(n) \cdot \overline{S_{map}}(n-1,k) + (1-\beta(n)) \cdot S_{map}(n,k), \quad k = 0, \ldots, 79 \tag{5.2-58}$$

and

$$S_{mass} = \frac{1}{80} \sum_{k=0}^{79} \overline{S_{map}}(n,k) \tag{5.2-59}$$

where *n* denotes the current frame and *k* denotes the frequency bin. The weight $\beta(n)$ used in the equation above is called the soft VAD parameter. It is initialized to 0 and updated in each frame as

$$\beta(n) = 0.95 \cdot \beta(n-1) + 0.05 \cdot f_{VAD}(n) \quad (5.2\text{-}60)$$

where $f_{VAD}(n)$ is the binary VAD flag from the IVAS encoder. The weight $\beta(n)$ is limited to the range (0.05, 0.95). The tonality flag is set by comparing S_{mass} with an adaptive threshold, thr_{mass}. The threshold is initialized to 0.65 and incremented or decremented in steps of 0.01 in each frame. If S_{mass} is higher than 0.65, then it is increased by 0.01, otherwise it is decreased by 0.01. The threshold is upper limited to 0.75 and lower limited to 0.55. This adds a small hysteresis to the tonality flag.

The tonality flag, f_{ton}, is set to 1 if S_{mass} is higher than thr_{mass}. Otherwise, it is set to 0.

5.2.2.2.12.5 Spectral peak to average ratio

The peak-to-average ratio of the power spectrum is a measure of spectral dynamics. Input signals with strong spectral peaks are usually better encoded with a transform-domain coding rather than an LP-based coding. The spectral peak to average ratio is calculated from the power spectrum of the input signal in the logarithmic domain. Let the power spectrum in the logarithmic domain be denoted as $S(n,k)$ where n denotes the current frame and k denotes the spectral bin. The power spectrum $S(n,k)$ is calculated with

$$S(n,k) = 10 \cdot log\big(PS(k)\big), \quad k = 0, \ldots, 79 \quad (5.2\text{-}61)$$

where $PS(k)$ is the total per-bin energy, defined in eq (28) in clause 5.1.5 of [3]. The power spectrum $S(n,k)$ is smoothed with an IIR filter as follows

$$\overline{S_{LT}}(n,k) = 0.9 \cdot \overline{S_{LT}}(n-1,k) + 0.1 \cdot S(n,k), \quad k = 0, \ldots, 79 \quad (5.2\text{-}62)$$

Note, that only the lower part of the spectrum from 0 to 4 kHz (first 80 frequency bins) is taken into account.

The spectral peak-to-average ratio is then calculated as follows

$$r_{p2a}(n) = \max_k\big(\overline{S_{LT}}(n,k)\big) - \frac{1}{80}\sum_{k=0}^{79}\overline{S_{LT}}(n,k) \quad (5.2\text{-}63)$$

5.2.2.2.12.6 Pre-selection of the core-coder technology

The pre-selection of the core-coder technology is performed in the TD stereo mode and the DFT stereo mode. Note that both, S/M classification and core-coder pre-selecetion is skipped in the secondary channel of the TD stereo mode. Coder pre-selection is also not performed in the MDCT stereo mode.

The pre-selection of the core-coder technology is based on the output class of the IVAS S/M classifier, d_{SMC}, described in clause 5.2.2.2.11.2.11, and all auxiliary parameters, described in the clauses above. The information about the selected core-coder technology is stored and used in the IVAS codec in the form of two binary flags, f_{SM1} and f_{SM2}. The binary flag f_{SM1} corresponds to the first-stage speech/music decision from the EVS codec, described in clause 5.1.13.6.4 in [3]. The binary flag f_{SM2} corresponds to the second-stage speech/music decision from the EVS codec, described in clause 5.1.13.6.6 in [3].

The pre-selection of the core-coder technology is done using the following mapping:

Table 5.2-6: Mapping of the core-coder technology to S/M binary flags

core-coder technology	f_{SM1}	f_{SM2}
ACELP	0	0
GSC	1	0
MDCT	1	1

The initial selection of the core-coder technology is done using the logic described in Fig. 5.2-5.

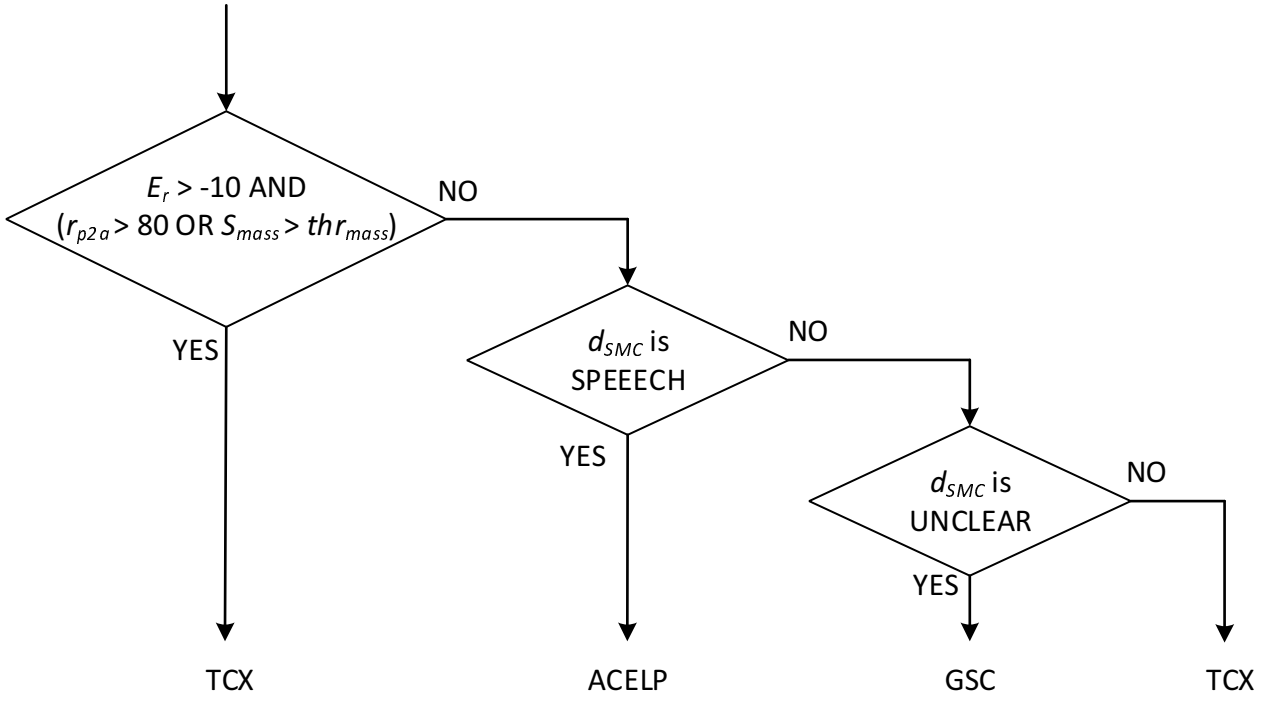


Figure 5.2-5: Pre-selection of the core-coder technology

The parameter E_r is the relative frame energy defined in eq. (29) in clause 5.1.5.2 of [3] and S_{mass} is the spectral mass defined in (5.2-59). As can be seen from the figure above, the initial selection of the core-coder technology follows the output class d_{SMC} of the IVAS S/M classifier. Note the exceptional case of strongly tonal signals for which the TCX technology is selected based on the values of r_{p2a} and S_{mass}.

The core-coder technology selected using the logic in Fig. 5.2-5 may be changed in certain special situations. For example, when GSC has been pre-selected but the input signal contains low-energy musical components below 400 Hz the core-coder technology is changed to ACELP. The presence of low-energy musical components below 400 Hz is identified by analyzing the following energy ratio

$$rat_{LF} = \frac{10 \cdot \log\left(\frac{1}{8}\sum_{k=0}^{7} E_{bin}(k)\right)}{E_{tot}} \tag{5.2-64}$$

where $E_{bin}(k)$, $k = 0, \ldots, 127$ is the power spectrum of the input signal in linear domain defined in eq. (25) in clause 5.1.5.2 of [3]. The summation term in the nominator extends over the first 8 bins of the energy spectrum corresponding to the frequency range of 0 to 400 Hz. The core-coder technology is changed from GSC to ACELP when the energy ratio rat_{LF} is sufficiently low and when the IVAS S/M classifier points to the MUSIC class. That is

$$f_{SM1} = 0, f_{SM2} = 0 \quad \text{if } rat_{LF} < 0.18 \text{ AND } dlp_{ST}(n) > 15.0 \tag{5.2-65}$$

where $dlp_{ST}(n)$ is the short-term mean of the differential score defined in eq. (5.3-67).

For input signals with very short and stable pitch period GSC is not the optimal as the core-coder technology. Input signals with very short and stable pitch period are identified by the parameter f_{sp}, defined in eq. (98) in clause 5.1.10.8 of [3]. The core-coder technology is changed from GSC to ACELP when the following condition is fulfilled

$$f_{SM1} = 0, f_{SM2} = 0 \quad \text{if } f_{sp} = 1 \text{ AND } wdlp(n) < 2.58 \tag{5.2-66}$$

Otherwise, the core-coder technology is changed from GSC to TCX when the following condition is fulfilled

$$f_{SM1} = 1, f_{SM2} = 1 \quad f_{sp} = 1 \text{ AND } wdlp(n) \geq 2.58 \tag{5.2-67}$$

Highly correlated input signal with low energy variation is another type of signal for which the GSC is not suitable as the core-coder technology. In such a situation the TCX core-coder technology is more appropriate. The condition below must be fulfilled to change the core-coder technology from GSC to TCX

$$f_{SM1} = 1, f_{SM2} = 1 \quad \text{if } f_{STAB}(n) = 1 \text{ AND } T_{OL}^{[0]} \geq 130 \tag{5.2-68}$$

where $T_{OL}^{[0]}$ is the absolute pitch value from the first half-frame of the open-loop pitch analysis, described in 5.1.10.of [3] and $f_{STAB}(n)$ is the long-term signal stability defined in eq. (5.2-52).

The pre-selected core-coder technology may also be changed from GSC to ACELP in frames where attack has been detected. The following condition must be fulfilled

$$cond_E = \begin{cases} 1 & E_{tot}(n) - E_{tot}(n-1) > 4.5 \text{ AND } E_{tot}(n) - 2 \cdot E_{tot}(n-1) + E_{tot}(n-2) > 10.0 \\ 0 & \text{otherwise} \end{cases} \quad (5.2\text{-}69)$$

The condition above ensures that the alteration of the core-coder technology happens only in segments with rising energy. If the condition above is fulfilled and, at the same time, the transition frame counter TC_{cnt}, has been previously set to 1, then the core-coder technology is changed from GSC to ACELP and the frame type is set to TRANSITION. That is

$$f_{SM1} = 0, f_{SM2} = 0 \quad \text{if } cond_E = 1 \text{ AND } TC_{cnt} = 1 \quad (5.2\text{-}70)$$

This means that the attack will be encoded with the TRANSITION mode in the ACELP core. Note, that the transition frame counter is denoted as i_{TC} in clause 5.1.13.4 of [3].

If an attack has been detected using the segmental attack detection procedure, described in clause 5.2.2.2.12.3, then the index (position) of this attack, k_{ata} , defined in eq. (366) from [3], is further examined. If the position of the detected attack is in the last subframe, then the core-coder technology is changed to ACELP and the frame type is set to TRANSITION. That is

$$f_{SM1} = 0, f_{SM2} = 0 \quad \text{if } TC_{cnt} \neq 1 \text{ AND } k_{ata} > 24 \quad (5.2\text{-}71)$$

If the position of the detected attack is not located in the last subframe but at least beyond the first quarter of the first subframe, then the attack will be encoded with the GSC core-coder technology. That is

$$f_{SM1} = 1, f_{SM2} = 0 \quad \text{if } TC_{cnt} \neq 1 \text{ AND } k_{ata} > 4 \quad (5.2\text{-}72)$$

In both situations, described above, i.e. when the attack is encoded either with the ACELP core-coder technology or with the GSC core-coder technology, an attack flag is set as follows

$$f_{ata} = k_{ata} + 1 \quad (5.2\text{-}73)$$

As the parameter f_{ata} reflects the position of the detected attack the name “attack flag” is somewhat. However, it’ll be used in this document for consistency with the source code of the IVAS codec.

When ACELP core-coder technology has been pre-selected the frame type may be changed from GENERIC to TRANSITION in case an attack has been detected with the segmental procedure as described in clause 5.2.2.2.12.3, i.e. where $k_{ata} > 0$. The frame type may be changed from GENERIC to TRANSITION only in active frames, i.e. when the local VAD flag f_{LVAD}, defined in clause 5.1.12.1 of [3] as f_{LSAD}, has been set to 1. After changing the frame type the attack flag is set similarly as in the previous situation. That is

$$f_{ata} = k_{ata} + 1 \quad (5.2\text{-}74)$$

5.2.2.3 Core-coder modules

5.2.2.3.1 Core-coder pre-processing

5.2.2.3.1.1 Selection of internal sampling rate

The LP-based core-coder within the IVAS codec operates at two internal sampling rates, 12.8 kHz and 16 kHz and the the MDCT-based core-coder operates at four internal sampling rates, 12.8, 16, 25.6 and 32 kHz.

The selection of the internal sampling rate of the LP-based core-coder is based on bitrate, the operating mode, the bandwidth and the ACELP16k binary flag, $f_{ACELP16k}$. The ACELP16k binary flag is used as an indicator that 16kHz sampling rate shall be selected for the LP-based core coder. The ACELP16k binary flag is set as follows.

In the DFT strereo mode, the ACELP16k binary flag is set to 1 for SID and NO_DATA frames when the codec operates in the DTX mode with LP-CNG encoding (see clause 5.2.2.3.5.5) but only if the bandwidth of the input signal is WB. For all other cases within the DFT stereo mode, the ACELP16k binary flag is set to 1 when the element bitrate $brate_{element}$ is higher or equal to 24.4 kbps and to 0 when the bitrate is lower than 24.4 kbps. In the TD stereo mode

the ACELP16k binary flag is set to 1 for the core-coder in the primary channel when the bitrate is higher or equal to 24.4 kbps. However, when LRTD sub-mode has been selected in the TD stereo encoder the ACELP16k binary flag is set to 1 when the bitrate is higher than 24.4 kbps and not when the bitrate is equal to 24.4 kbps. For all bitrates lower than 24.4 kbps in the TD stereo mode the ACELP16k binary flag is set to 0.

For the encoding of SCE (single-channel element) the ACELP16k binary flag is set to 1 when the element bitrate is higher than or equal to 17 kbps. For all other bitrates it is set to 0.

In case of EVS mono operation the ACELP16k binary flag is set based on the logic described in clause 5.4.4 of [3].

The internal sampling rate of the LP-based core-coder and the MDCT-based core-coder (TCX) is initially set to 16kHz when the ACELP16k binary flag is set to 1. Otherwise, the internal sampling rate is initially set to 12.8 kHz. When the IVAS codec operates in the MDCT stereo mode and the element bitrate is higher than or equal to 64 kbps TCX core-coder is selected and its internal sampling rate is set to 32kHz. Similarly, for the encoding of SCEs, TCX core-coder is selected and its internal sampling rate is set to 32kHz when the core-coder bitrate is higher than 48 kbps. The TCX core is also selected when the IVAS codec operates in the ISM mode (object-based audio) and the core-coder bitrate is higher than 40 kbps and lower than or euqal to 48 kbps. In this case, however, the internal sampling rate of the TCX core-coder is set to 25.6 kHz.

During the DTX operation, the internal sample rate of the core-coder is not allowed to change during NO_DATA frames. The internal sampling rate is also not allowed to change in SID frames following short segments of active frames.5.2.2.3.1.2 Core-coder technology selection

5.2.2.3.1.2.1 Overview

The selection of the core-coder technology in the IVAS codec is based on the mechanism described in clause 5.1.14 in [3]. The core-coder technology is selected from the following list

- ACELP core-coder technology
- GSC core-coder technology
- TCX core-coder technology
- HQ core-coder technology

The selection of the core-coder technology follows the pre-selection mechanism, described in detail in clause 5.2.2.2.12, which is part of the front pre-processing module.

The selection of the core-coder technology is based on the pre-selected core-coder technology which is stored in the form of the S/M binary flags f_{SM1} and f_{SM2}. The selection of the core-coder technology takes into account bitrate limitations, bandwidth limitations, VAD flag and other auxiliary parameters. Table 5.2-7 shows the selection of the core-coder technology based on core-coder bitrate, bandwidth and content type.

Table 5.2-7: Selection of core-coder technology based on bitrate, bandwidth and content type

bandwidth	content	core-coder total bitrate [kbps]					
		13.2	16.4	24.4	32	48	>48
WB	speech	ACELP	ACELP	ACELP	ACELP	ACELP	TCX
	audio	GSC/TCX	GSC/TCX	TCX/HQ	TCX/HQ	TCX/HQ	TCX
	inactive	GSC/TCX	GSC /TCX	GSC /TCX	ACELP/TCX	ACELP/TCX	TCX
SWB	speech	ACELP	ACELP	ACELP	ACELP	ACELP	TCX
	audio	GSC/TCX	GSC/TCX	TCX/HQ	TCX/HQ	TCX/HQ	TCX
	inactive	GSC/TCX	GSC/TCX	GSCI/TCX	ACELP/TCX	ACELP/TCX	TCX
FB	speech				ACELP	ACELP	TCX
	audio				TCX/HQ	TCX/HQ	TCX
	inactive				ACELP/TCX	ACELP/TCX	TCX

Please, note that the core-coder bitrate may be different from the bitrates listed in Table 5.2-6 due to variable bitrate allocation mechanism in the ACELP/TCX coder, described in clauses 5.2.2.3.2 and 5.2.2.3.3.

As can be seen in Table 5.2-6 the ACELP core-coder technology is selected when the content is classified as “speech” which is signalled to the selector by the S/M binary flags. The content is considered “speech” when $f_{SM1} = 0$ and $f_{SM2} = 0$. ACELP core is also selected for the encoding of SID frames and NO_DATA frames in DTX mode. In the

ISM low-rate mode (see clause 5.6.2.3.2) ACELP core is selected for the encoding of inactive frames. In this case, $coder\_type$ is set to INACTIVE.

For "music" content the IVAS codec selects among GSC, TCX or HQ core-coder technologies. The selection between the GSC core-coder technology and the TCX/HQ core-coder technology is based on the values of the f_{SM1} and f_{SM2} S/M binary flags. The GSC core is selected when $f_{SM1} = 1$ and $f_{SM2} = 0$. In case when $f_{SM1} = 1$ and $f_{SM2} = 1$ the IVAS codec selects either the TCX core-coder technology or the HQ core-coder technology.The selection between the TCX core-coder technology and the HQ core-coder technology is based on the output of the HQ classifier, described in detail in clause 5.2.2.3.1.2.4.

For "inactive" content the selection of the core-coder technology is based on the coding mode classification, described in clause 5.2.2.2.10. The coding mode classification is based, among other parameters, on the SAD module, described in clause 5.2.2.2.5 and on the output of the IVAS S/M classifier, described in clause 5.2.2.2.11. For core-coder bitrates below 9 kbps and in cases where LP-CNG type has been chosen by the pre-processor (see clause 5.6 in [3]], GSC core-coder technology is selected for the encoding of inactive content. In all other cases TCX core-coder technology is selected.

The TCX core-coder technology is used for any content at bitrates higher than 48 kbps except of the ISM format coding where it is used for any content higher than 40 kbps. The TCX core-coder technology is also selected when the IVAS codec operates in the MDCT stereo mode. In case TCX core-coder technology has been selected by the logic described so far but the bitrate is lower than 9 kbps GSC core is used instead. In this case f_{SM2} is reset to 0 and $coder\_type$ is set to AUDIO. However, if the IVAS codec operates in the ISM low-rate mode $coder\_type$ is set to INACTIVE instead of AUDIO.

In the special case when the IVAS codec operates in LRTD stereo submode with ACELP core-coder technology running at bitrate lower than or equal to 16.4 kbps, the selection mechanism prevents the IVAS codec from switching directly into DFT stereo mode with TCX core-coder technology. This is to avoid excessive computational complexity. The IVAS codec allows for switching into DFT stereo mode but enforces ACELP core-coder technology in the first frame where the switch occurs. In the successive frames core-coder technology is selected without any restrictions using the logic described so far.

Please, note that the selected core-coder technology may be changed in some other special situations which is described in other clauses in this document.

5.2.2.3.1.2.2 TD/FD BWE technology selection

The ACELP core-coder in the IVAS codec uses the TD and FD bandwidth extension (BWE) technology from the EVS codec. The selection between the TD BWE and the FD BWE is described in detail in clause 5.1.14.4 of [3]. In the IVAS codec, the selection mechanism has been further modified to accomodate the variable-rate of the ACELP core-coder and the new SWB TBE modes at 1.10 kbps and 1.75 kbps, described in clause 5.2.2.3.2.7.

The selection between the TD BWE and the FD BWE technology is based on the characteristics of the input signal, the bandwidth and the core-coder of the low-band signal. Table 5.2-7b lists the bitrates of all TD BWE and FD BWE technologies.

Table 5.2-7b: Bitrates of TD BWE and FD BWE technologies

bandwidth	time-domain	frequency-domain
WB	**WB TBE at 0.35 kbps**	WB BWE at 0.35 kbps
	WB TBE at 1.05 kbps	
SWB	**SWB TBE at 0.95 kbps**	
	SWB TBE at 1.10 kbps	
	SWB TBE at 1.75 kbps	
	SWB TBE at 1.6 kbps	SWB BWE at 1.6 kbps
	SWB TBE at 2.8 kbps	
FB	**FB TBE at 1.8 kbps**	FB BWE at 1.8 kbps
	FB TBE at 3.0 kbps	

For WB signals, the bandwidth extension technology is selected as follows. When the core-coder bitrate is lower than 7.15 kbps the bandwidth extension is done in frequency domain without any side information transmitted in the bitstream. This can be referred to as the WB BWE at 0 kbps. The same technology is applied also in the TD stereo mode for the encoding of the secondary channel. If the core-coder bitrate is higher than or equal to 7.15 kbps and the

binary flag ACELP16k $f_{ACELP16k}$ is set to 0, the bandwidth extension technology is set based on the S/M binary flags f_{SM1} and f_{SM2}. described in clause 5.2.2.2.12.6. When $f_{SM1} = 1$ and $f_{SM2} = 0$, the selected bandwidth extension technology is WB BWE at 0.35 kbps. The same WB BWE technology is selected for inactive frames, i.e. when the coder type is set to INACTIVE. For all other combinations of S/M binary flags f_{SM1} and f_{SM2}, WB TBE technology is selected but only under the condition that the coder type is different than INACTIVE. The bitrate of the selected WB TBE technology is set with the following logic. In case the core-coder bitrate is lower than 9.65 kbps, WB TBE at 0.35 kbps is used. Similarly, when the IVAS codec operates in the TD stereo mode, the primary channel is encoded with WB TBE at 0.35 kbps. For all other situations, WB TBE at 1.05 kbps is used.

For SWB and FB signals, the bandwidth extension technology is selected as follows. The WB BWE technology at 0 kbps, described in the previous parapraph, may also be applied for SWB and FB signals when neither of the following three conditions is fullfilled. The first condition is fullfilled when the core-coder bitrate is higher than or equal to 7.8 kbps. The second condition is specific only to the LRTD stereo sub-mode and it is fullfilled when the core-coder bitrate is higher than or equal to 5 kbps. The third condition is specific to the TD stereo mode (both LRTD sub-mode and regular sub-mode) and it is fullfilled when the core-coder bitrate is higher than or equal to 5 kbps and the CPE bitrate is lower than 16.4 kbps. When all three conditions are fullfilled, the IVAS codec selects one of the following SWB or FB bandwidth extension technologies. Note, that the selected BWE technology is always encoded with non-zero side information that is sent to the decoder in the bitstream.

The selection of the SWB/FB bandwidth extension technology is based on the S/M binary flags f_{SM1} and f_{SM2}. described in clause 5.2.2.2.12.6 and the SWB noisy speech flag, f_{UV_SWB}, described in clause 5.1.13.6.9 of [3]. When $f_{UV\_SWB} = 0$ and, at the same time, $f_{SM1} = 1$ and $f_{SM2} = 0$, the selected bandwidth extension technology is the SWB BWE at 1.6 kbps. In the case of $f_{UV\_SWB} = 0$, the SWB BWE at 1.6 kbps is also selected for all INACTIVE signals. Furthermore, if the input bandwidth is FB, then FB BWE at 1.8 kbps is selected on top of SWB BWE at 1.6 kbps for the encoding of the upper band from 14 to 20 kHz. If none of the above conditions is fullfilled SWB TBE technology is selected. The bitrate of the SWB TBE technology is based on the core-coder bitrate and the state of some auxiliary parameters. The selection of the SWB TBE bitrate is set as follows. The initial bitrate of the SWB TBE is set to 1.6 kbps. The initial bitrate may be modified under some special conditions. In case of SCE encoding, when the binary flag ACELP16k $f_{ACELP16k}$ set to 1, the SWB TBE bitrate is increased to 2.8 kbps but only when the core-coder bitrate is higher than or equal to 24.4 kbps. In the LRTD stereo sub-mode, the bitrate of the SWB TBE is either decreased to 1.1 kbps when the CPE bitrate is lower than 24.4 kbps or increased to 1.75 kbps when the CPE bitrate is higher than or equal to 24.4 kbps. Finally, the SWB TBE bitrate is decreased to 0.95 kbps if the core-coder bitrate is lower than 13.2 kbps. This is the minimum SWB TBE bitrate. If the input bandwidth is FB, then the SWB TBE technology is complemented with the FB TBE technology for the encoding of the upper band from 14 kHz to 20 kHz. The bitrate of the selected FB TBE is set as follows. The initial bitrate of the FB TBE technology is set to 1.8 kbps. In case of SCE encoding, when the binary flag ACELP16k $f_{ACELP16k}$ set to 1, the FB TBE bitrate is increased to 3.0 kbps but only when the core-coder bitrate is higher than or equal to 24.4 kbps.

The IVAS codec also contains the IC-BWE technology, optimized for the encoding of the bandwidth extension of stereo signals. The IC-BWE encoder is described in detail in clause 5.3.2.2.1. The IC-BWE bitrate is set as follows. In the TD stereo mode, the IC-BWE technology is applied only in the regular submode (not LRTD submode). The IC-BWE bitrate is set either to 0.35 kbps when the binary flag ACELP16k $f_{ACELP16k}$ is equal to 1 or to 0.25 kbps when the binary flag ACELP16k $f_{ACELP16k}$ is equal to 0. In the DFT stereo mode, the IC-BWE bitrate is set either to 0.5 kbps when the binary flag ACELP16k $f_{ACELP16k}$ is equal to 1 or to 0.4 kbps when the binary flag ACELP16k $f_{ACELP16k}$ is equal to 0. The bitrate of the IC-BWE may be further increased by 0.5 kbps in case when ACELP core coder has been selected for the encoding of the lower band. However, this increase of bitrate is not applied in the LRTD sub-mode within the TD stereo mode.

5.2.2.3.1.2.3 TCX/HQ MDCT technology selection

For IVAS core-coder nominal bitrates ≥ 16.4 kbps and < 48 kbps, the TCX/HQ MDCT technology is selected based on signal characteristics. This selection in IVAS is based on the EVS "TCX/HQ MDCT technology selection at 24.4 and 32 kbps", see TS 26.445, clause 5.1.14.3 in [3]. For IVAS, the following differences apply:

- In case noisy speech is detected in the pre-processing stage (see 5.2.2.2) and the VAD flag is not set, TCX is selected.

- For the DFT Stereo CPEs, the spectral analysis spectral analysis *X*(*k*) is taken from the DFT Stereo downmix signal *M(k)* (see clause 5.3.2.4.9).

For IVAS core-coder nominal bitrates <16.4 kbps and ≥ 48 kbps, TCX is always selected.

5.2.2.3.1.2.4 HQ classifier

For SWB bitrates at 24.4 and 32, there are 4 modes supported, Transient, Harmonic, HVQ and Generic as described in 5.3.4.2 in [3]. A frame is considered harmonic following the classification described in 5.3.4.2.3 [3]. An HQ classifier is used to switch to Generic mode for harmonic frames if a sparse harmonic structure is not detected in the spectrum, otherwise the classification follows that described in 5.3.4.2.3 [3]. A spectrum sparseness analysis is performed based on obtaining a peakyness measure and a noise band detection measure derived from the MDCT coefficients.

First, the magnitude A_i of a critical frequency region is obtained by

$$A_i = |X(k_{start} + i)|, i = 0, \dots, k_{end} \tag{5.2-75}$$

where $X(k)$ is the MDCTs spectrum computed, k_{start} is the first bin the critical frequency region and k_{end} is the last bin in the critical frequency region. k_{start} and k_{end} are set to 320 and 639 respectively where the input sampling rate is 32 kHz and the frame length is 640. The critical frequency region is the upper half of the MDCT spectrum.

The peakyness measure is obtained by obtaining the crest in accordance with equation (5.2-76) below

$$crest = \frac{\max(A_i)}{\sqrt{\frac{1}{M}\sum_{i=0}^{M-1} A_i^2}} \tag{5.2-76}$$

where $M = k_{end} - k_{start} + 1$ is the number of bins in the critical band. A complimentary peakyness measure pk_{low} is obtained in accordance with equation (5.2-77)

$$pk_{low} = \sum_{i=0}^{M-1} low(A_i) \tag{5.2-77}$$

where

$$low(A_i) = \begin{cases} 1, A_i < 0.1 \max(A_i) \\ 0, A_i \geq 0.1 \max(A_i) \end{cases} \tag{5.2-78}$$

A noise band detection measure is obtained according to equation

$$crest_{mod} = \frac{\max\left(movmean(A_i, W)\right)}{\sqrt{\frac{1}{M}\sum_{i=0}^{M-1} A_i^2}} \tag{5.2-79}$$

where $movmean(A_i, W)$ is a moving mean of the absolute spectrum A_i using a window size W fixed to 21. The moving mean is computed according to equation (5.2-80) below.

$$movmean(A_i, W) = \frac{1}{b-a+1}\sum_{i=a}^{b} A_i(m), \text{ where } \begin{cases} a = \max(0, i - (W-1)/2 \\ b = \min(M-1, i + (W-1)/2 \end{cases} \tag{5.2-80}$$

$crest_{mod}$ gives a measure of local concentration of energy, indicating a noise band in the spectrum. To stabilize the decision, $crest$ and $crest_{mod}$ are low pass filtered according to equation.

$$crest_{LP} = (1-\alpha) \cdot crest + \alpha \cdot crest_{LP}^{-1} \tag{5.2-81}$$

$$crest_{mod,LP} = (1-\alpha) \cdot crest_{mod} + \alpha \cdot crest_{mod,LP}^{-1} \tag{5.2-82}$$

where α is set to 0.97.

The spectrum sparseness analysis obtains a harmonic decision for the current frame in accordance with equation (5.2-83):

$$Harmonic\_decision = \begin{cases} FALSE, \ crest_{LP} > crest_{thr}, crest_{mod,LP} > crest_{mod,thr}, pk_{low} > pk_{low-thr} \\ TRUE, \ otherwise \end{cases} \tag{5.2-83}$$

Table 5.2-8: Thresholds for HQ classifier

threshold	value
$crest_{thr}$	7.0
$crest_{mod,thr}$	2.128
pk_{low_thr}	220

The Generic mode is selected if the condition below is fulfilled, otherwise the mode decision is left unchanged and follows that described in clause 5.3.4.2.3 of [3] to select between HQ_HARMONIC and HQ_HVQ mode.

$$\text{Harmonic \&\& ! Harmonic\_decision.} \quad (5.2\text{-}84)$$

The audio frame is then encoded using the HQ Core encoder with the selected mode.

5.2.2.3.2 LP based coding

5.2.2.3.2.1 Variable bitrate ACELP coding

The ACELP core-coder mode in EVS [3] is mainly based on a constant bitrate principle where a bit-budget to encode a given frame is constant during the encoding. In order to obtain the best possible quality at a given constant bitrate, the bit-budget is carefully distributed among the different coding parts. Specifically, the bit-budget per coding part at a given bitrate is fixed and stored in codec ROM tables.

On the other hand, IVAS is a complex multi-module codec where the bit-budget at a constant bitrate is allocated between different modules based on, for example, a number of input audio channels, audio bandwidth, spatial audio characteristics of input signal, presence of metadata, available codec total bitrate, etc. Then in IVAS, the codec total bit-budget is distributed among the core module tools (see clause 5.2.3) and other different modules like a bandwidth extension (BWE), stereo or spatial parameters module(s), metadata module(s), etc. which are collectively referred to in the further text as “supplementary codec modules”. In order to get the most efficiency and flexibility in IVAS, the allocated bit-budget per supplementary module is variable by default making a demand for the core-codec to support a variable bitrate coding as well. Consequently, the bit-budget allocated to the ACELP core module can fluctuate between a relatively large minimum and maximum bitrate span with a granularity of 1 bit (i.e. 0.05 kbps at a frame length of 20 ms).

Dedicated ROM table entries for all possible ACELP core module bitrates as done in EVS are obviously inefficient in IVAS with variable bitrate coding. Thus, a more efficient and flexible distribution of the bit-budget among the different modules with fine bitrate granularity based on a limited number of intermediate bitrates is employed in IVAS and described in the next clauses.

5.2.2.3.2.2 Bit-budget allocator

5.2.2.3.2.2.1 Overview

Figure 5.2-6 is a block diagram illustrating the bit-budget allocator in the IVAS codec.

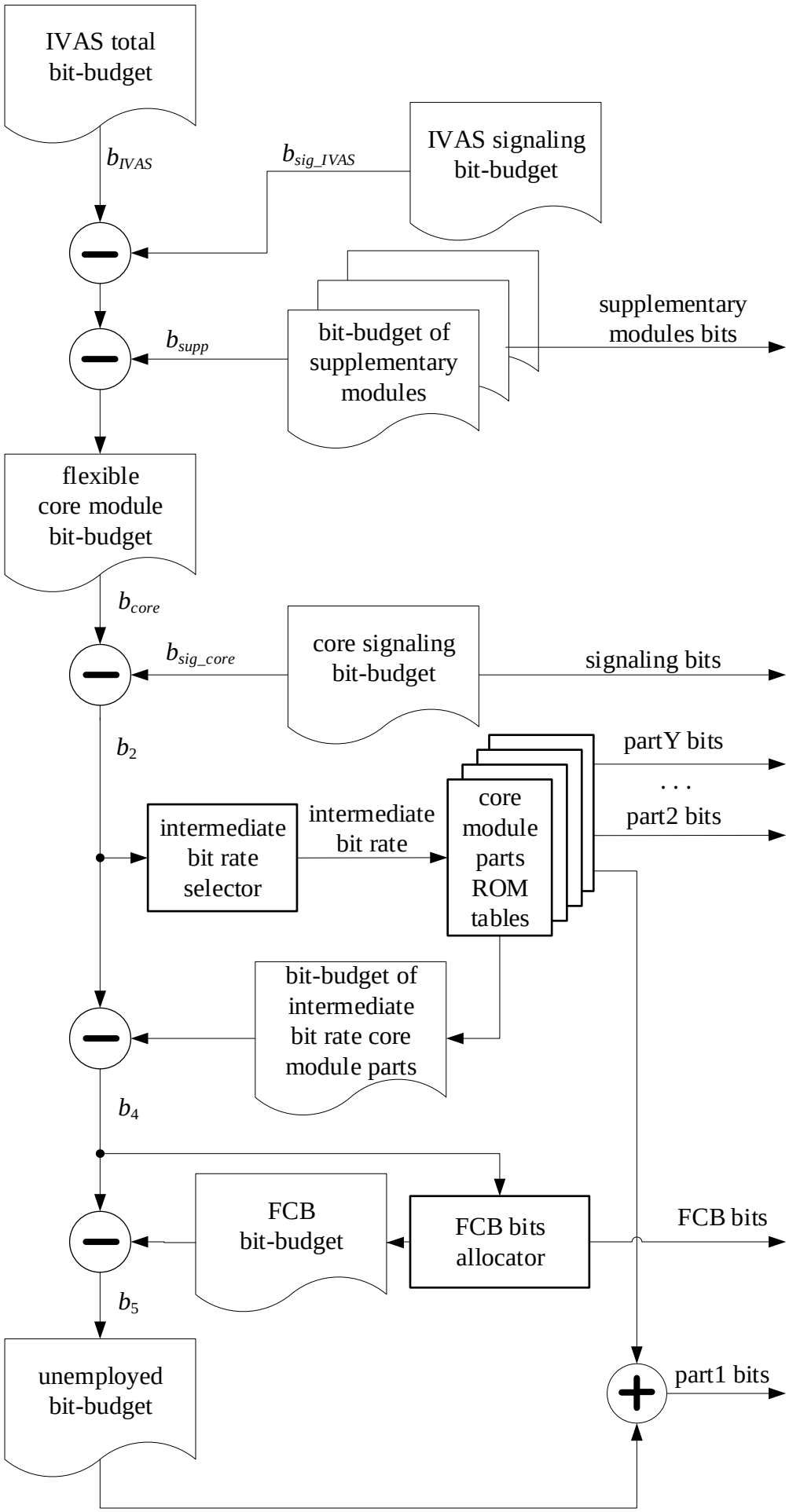

IVAS total bit-budget
b_{IVAS}
b_{sig_IVAS}
IVAS signaling bit-budget
b_{supp}
bit-budget of supplementary modules
supplementary modules bits
flexible core module bit-budget
b_{core}
b_{sig_core}
core signaling bit-budget
signaling bits
b_2
partY bits
. . .
part2 bits
intermediate bit rate selector
intermediate bit rate
core module parts ROM tables
bit-budget of intermediate bit rate core module parts
b_4
FCB bit-budget
FCB bits allocator
FCB bits
b_5
unemployed bit-budget
part1 bits

Figure 5.2-6: Block diagram of the bit-budget allocator

The ACELP bit-budget allocator from Figure 5.2-6 operates on a frame-by-frame basis and consists of several operations described below. Its fluctuating bit-budget is a result of a fluctuating number of bits used for encoding supplementary codec modules while these modules are processed and their bit-budget set before the core-coder bit-budget allocator is run. It is noted that the distribution of bit-budget among the different core module parts is symmetrically done at the encoder and the decoder using the same allocation algorithm.

Referring to Figure 5.2-6, an IVAS total bit-budget b_{IVAS} is allocated to the codec for each frame.

First, there are determined the number of bits (bit-budget) of all supplementary modules, b_{supp}, used for encoding the supplementary codec modules like stereo or spatial information, metadata, or BWE, and the number of bits b_{sig_IVAS} for transmitting IVAS format signalling to the decoder. These two bit-budgets are then subtracted from the IVAS total bit-budget to obtain a bit-budget of the ACELP core module, b_{core}, using the following relation:

$$b_{core} = b_{IVAS} - b_{sig\_IVAS} - b_{supp}. \quad (5.2\text{-}85)$$

Next operation subtracts the bit-budget b_{sig_core} related to the transmission of ACELP core module signalling parameters like core-coder type or audio bandwidth, similarly as done in EVS:

$$b_2 = b_{core} - b_{sig\_core}. \quad (5.2\text{-}86)$$

Then, the bit-budget b_2 is converted in an intermediate bitrate selector from Figure 5.2-6 to bitrate $brate_2$ and its related so-called candidate bitrate is found. Specifically, in IVAS, a predetermined number of twenty-two (22) candidate bitrates is used. These bitrates are as follows (note an overlap with defined EVS core-coder bitrates): 5.00 kbps, 6.15 kbps, 7.20 kbps, 8.00 kbps, 9.60 kbps, 11.60 kbps, 12.15 kbps, 12.85 kbps, 13.20 kbps, 14.80 kbps, 16.40 kbps, 22.60 kbps, 24.40 kbps, 29.00 kbps, 29.20 kbps, 30.20 kbps, 30.40 kbps, 32.00 kbps, 48.00 kbps, 64 kbps, 96 kbps, and 128 kbps. The found intermediate bitrate for core-coder parts is then the nearest higher candidate intermediate bitrate to the ACELP core module bitrate. For example, for $b_2 = 9.00$ kbps bitrate the found intermediate bitrate would be 9.60 kbps using the candidate intermediate bitrates from the list of 22 IVAS candidate bitrates.

Then, for each candidate intermediate bitrate, pre-determined bit-budgets for encoding first parts of the ACELP core module are stored in ROM tables. These ACELP core module first parts comprise the LP filter coefficients, the adaptive codebook, the adaptive codebook gain, and the innovation codebook gain. Note that the ROM tables, one ROM table per one first part parameter, used to allocate the ACELP first parts bit-budgets are ROM tables as used in EVS complemented by five (5) other candidate bitrates. Also note that no bit-budget for encoding the innovation codebook is stored in the ROM tables but its bit-budget is computed as described in the next clause 5.2.2.3.2.2.2.

The bit-budget selector thus allocates for encoding the ACELP core module first parts the bit-budgets stored in the ROM tables and associated to the intermediate bitrate selected by the bit-budget selector.

In the next step, the bit-budget for intermediate bitrate core module first parts is subtracted from the bit-budget b_2. More specifically, there are subtracted from bit-budget b_2 the following bit-budgets: (a) bit-budget b_{LPC} for encoding the LP filter coefficients associated to the candidate intermediate bitrate selected by the bit-budget selector, (b) the sum of the bit-budgets $b_{ACB}[m], m = 0, .., M-1$, of the M subframes associated to the selected candidate intermediate bitrate, (c) the sum of the bit-budgets $b_{gain}[m]$ for quantizing the adaptive and innovation codebook gains of the M subframes associated to the selected candidate intermediate bitrate, (d) the bit-budget, associated to the selected intermediate bitrate, for encoding other ACELP core module first parts that are present like FEC bits or the low-pass adaptive excitation filtering flag. Thus, a remaining bit-budget b_4 still available for encoding the innovation codebook (which relates to the second ACELP core module part) is found as:

$$b_4 = b_2 - b_{LPC} - \sum_{m=0}^{M-1} b_{ACB}[m] - \sum_{m=0}^{M-1} b_{gain}[m] - \cdots. \quad (5.2\text{-}87)$$

5.2.2.3.2.2.2 Innovation codebook bit-budget distribution

The distribution of bit-budget for innovation codebook coding is done in Fixed Codebook (FCB) allocator as part of the bit-budget allocator from Figure 5.2-6. The FCB allocator distributes the remaining bit-budget b_4 from equation (5.2-87) for encoding the innovation codebook (aka second ACELP core module part) between the M subframes of the current frame. Specifically, the bit-budget b_4 is divided into bit-budgets $b_{FCB}[m]$ allocated to the various subframes m. At the same time, there must be kept in mind that only some bit-budget values are supported by the innovative codebook. These supported bit-budgets per subframe are 7, 10, 12, 15, 17, 20, 24, 26, 28, 30, 32, 34, 36, 40, 43, 46, 47, 49, 50, 53, 55, 56, 58, 59, 61, 62, 65, 68, 70, 73, 75, 78, 80, 83, 85, 87, 89, 92, 94, 96, and 98 bits.

The FCB allocator then distributes the bit-budget between subframes in an iterative procedure which divides the bit-budget b_4 between the M subframes as equally as possible while assuming the following principles:

(1) In case the bit-budget b_4 cannot be distributed equally between all the subframes, a highest possible (i.e. a larger) bit-budget is allocated to the first subframe. As an example, if $b_4 = 106$ bits, the FCB bit-budget per 4 subframes is allocated as 28-26-26-26 bits.

(2) If there are more bits available to potentially increase other subframe FCB codebooks, the FCB bit-budget allocated to at least one next subframes after the first subframe is increased. As an example, if $b_4 = 108$ bits, the FCB bit-budget per 4 subframes is allocated as 28-28-26-26 bits. In another example, if $b_4 = 110$ bits, the FCB bit-budget per 4 subframes is allocated as 28-28-28-26 bits, etc.

(3) The bit-budget b_4 is not necessarily distributed as equally as possible between all the subframes but rather to use as much as possible the bit-budget b_4. As an example, if $b_4 = 87$ bits, the FCB bit-budget per 4 subframes is allocated as 26-20-20-20 bits rather than e.g. 24-20-20-20 bits when the principle (3) would not be considered. In another example, if $b_4 = 91$ bits, the FCB bit-budget per 4 subframes is allocated as 26-24-20-20 bits in contradiction to e.g. 24-24-20-20 bits would be allocated if the principle (3) is not considered. Consequently, only 1 bit remains unused when the principle (3) is considered while 3 bits would remain unused otherwise.

(4) In case the bit-budget cannot be equally distributed between all the subframes when encoding using a Transition Coding (TC) mode (see clause 5.2.3.2 in [3]), the largest possible (i.e. larger) bit-budget is allocated to the subframe using a glottal-impulse-shape codebook. As an example, if $b_4 = 122$ bits and the glottal-impulse-shape codebook is used in the third subframe, the FCB bit-budget per 4 subframes is allocated as 30-30-32-30 bits.

(5) If, after applying the principle (4), there are more bits available to potentially increase another FCB codebook in a TC mode frame, the FCB bit-budget allocated to the last subframe is increased. As an example, if $b_4 = 116$ bits and the glottal-impulse-shape codebook is used in the second subframe, the FCB bit-budget per 4 subframes is allocated as 28-30-28-30 bits.

The sum of the bit-budgets (number of bits) $b_{FCB}[m], m = 0, .., M-1$, allocated to the M various subframes for encoding the innovation codebook is thus

$$\sum_{m=0}^{M-1} b_{FCB}[m] \tag{5.2-88}$$

Then, a subtractor at the very bottom of Figure 5.2-6 determines the number of bits b_5 remaining after encoding of the innovation codebook, using the following relation:

$$b_5 = b_4 - \sum_{m=0}^{M-1} b_{FCB}[m]. \tag{5.2-89}$$

Ideally, after encoding of the innovation codebook, the number of remaining bits b_5 is equal to zero. However, it may not be possible to achieve this result because the granularity of the innovation codebook index is greater than 1 (usually 2-3 bits). Consequently, a small number of bits often remain unemployed after encoding of the innovation codebook.

Finally, a bit-budget allocator from Figure 5.2-6 thus assigns the unemployed bit-budget b_5 to increase the bit-budget of one of the ACELP core first module parts. Specifically, the unemployed bit-budget b_5 is used to increase the bit-budget b_{LPC} obtained from the ROM tables, using the following relation:

$$b'_{LPC} = b_{LPC} + b_5. \tag{5.2-90}$$

5.2.2.3.2.2.3 Bit-budget distribution in high bitrate ACELP

At high bitrate ACELP core, there is used a combined algebraic codebook as described in clause 5.2.3.1.6 of [3] which contains a transform-domain AVQ-based codebook. In high bitrate ACELP, bit-budget is allocated to the ACELP core module parts using an extended version of the procedure as described in previous clause 5.2.2.3.2.2. Following this procedure, the sum of the bit-budgets $b_{FCB}[m], m = 0, \ldots, M-1$, from equation (5.2-88) for encoding the FCB codebook in the M subframes should be equal or approach bit-budget b_4. In the high bitrate ACELP, the bit-budgets $b_{FCB}[m]$ are however modest, and the number of unemployed bits b_5 from equation (5.2-89) is relatively high and is used to encode the transform-domain codebook parameters as follows.

First, the sum of the bit-budget $b_{TDgain}[m]$ for encoding the transform-domain pre-quantizer gain (see clause 5.2.3.1.6.2 in [3]) in the M subframes are subtracted from the unemployed bit-budget b_5, using the following relation:

$$b_7 = b_5 - \sum_{m=0}^{M-1} b_{TDgain}[m]. \tag{5.2-91}$$

Then, the remaining bit-budget b_7 is allocated to the AVQ vector quantizer within the transform-domain codebook (see clause 5.2.3.1.6.9 in [3]) and distributed among all M subframes. The bit-budget by subframe of the vector quantizer is denoted as $b_{VQ}[m]$. Giving the AVQ quantization steps, the vector quantizer does not consume all of the allocated bit-budget $b_{VQ}[m]$ leaving a small variable number of bits available in each subframe. These bits are floating bits employed in the following subframe within the same frame. For a better effectiveness of the transform-domain codebook, a slightly higher bit-budget is allocated to the vector quantizer in the first subframe. The implementation of this logic is given in the following pseudocode:

```
b_tmp = floor(b_7 / M);
for( m = 0; m < M; m++ )
{
    b_VQ[m] = btmp;
}
b_VQ[0] = b_tmp + (b_7 – M * b_tmp)
```

where $floor(x)$ denotes the largest integer less than or equal to x and M is the number of subframes in one frame. Bit-budget b_7 is thus distributed equally between all the subframes while the bit-budget for the first subframe is eventually slightly increased by up to $M - 1$ bits. Consequently, in high bitrate ACELP, there are no remaining bits after this operation.

5.2.2.3.2.3 LP coefficients encoding

The encoding of the LP coefficients is performed on the LSF representation. The structure of the quantizer is similar to the EVS lattice based quantizer described in clause 5.2.2.1.4 of [3]. A safety net, predictive, or switched safety-net predictive multi-stage vector quantizer (MSVQ) is used to quantize the full-length frame-end LSF vector for all modes where it is used. In addition to the bitrates supported by the EVS LSF quantizer, in the IVAS codec, due to the fine granularity of the variable bit allocation between the coding blocks, more bitrate values are added for the multiple scale lattice vector quantizer (MSLVQ) LSF quantizer. The corresponding lattice structure structures are defined for the number of bits presented in Table 5.2-9 according to the signal coding type.

Table 5.2-9: Number of bits for which the MSLVQ are defined

Coding type	LSF quantizer - Number of bits	VQ bits	Predictive LSF quantizer	VQ bits
Unvoiced WB	14,15,18,19,25,28		14-28	12
Unvoiced NB	18,19,24,25,29,32		18,19,22,23,24,29,32	8
Voiced WB	17-37	8	9-39	6
Voiced NB	17,18,22,23,24,25,27,29,34,37	8	8,19,23,24,25,27-36,39	6
Generic WB	12-32	9	15-35	6
Generic NB	12,16,19,20,21,22,23,25-32	9	15,19,22-26,28-35	6
Transition WB	17-33	9		
Transition NB	17,18,22,23,24,25,28,31,32	9		
Generic 16k	31,32		26-37	5
Transition 16k	32,33	8		
Audio WB	17-36	4	21-40	0
Audio NB	17,21,22,25,26,28	4	21,25,26,30,31,32	0
Audio WB 16k	26,36		26-37	5
Voiced 16k	22-37	8	24-39	6
Inactive NB			17,21,22,25,26,27,36	5
Inactive WB			17-36	5
Inactive 16k			17,21,22,25,26,27,36	5

The VQ structures are defined by the corresponding unstructured codebooks and the lattice quantizers by a set of 6 scales and 6 numbers of leader vectors defining the structures as presented in clause 5.2.2.1.4 of [3].

To accommodate many more bitrates than are used for EVS, a different representation of the structures is used. For each bitrate of each coding mode there are the lattice structure definition and the lattice scales. One lattice structure definition consists of two groups of 3 integer numbers specifying the number of leader vectors in each of the three lattice truncations of each 8-dimensional half of the LSF vector. The lattice scales consist of two groups of 3 floating point scale values, one for each lattice truncation. In order to represent one lattice structure definition, a vector of two numeric pointers is used, a first pointer and a second pointer. The values of the pointers point to entries in a table comprising all (275) sub vectors defining the possible lattice structures for the lattice quantizers part for the first, and for the second half of the LSF vector. Where the first and second pointers each point to a sub vector half of the LSF vector, each sub vector half combining to give the full vector. The first and second pointers are contained as an entry in a table of pointers. A lattice structure for an 8-dimensional quantizer corresponding to a half of the LSF vector is

represented as a 3-dimensional vector of integers, each integer indicating how many leader classes form each lattice truncation. Because there are more than 256 different lattice structures, and the pointers are represented using 8 bits, pointer values that are larger than 256 are stored as their value minus 256. Following on, if the first pointer value is smaller than the second pointer value then this indicates that the first pointer value has been stored using modulo 256 arithmetic. In this case the first pointer is given by adding 256 to the value of the stored first pointer. However, if the first pointer value is larger (or the same value) as the second pointer value then the value of the first pointer is directly used to access the LSF sub vector.

There is one exception which is handled separately. The numeric pointer [19, 0] has 256 added to both components. It can be handled separately because it is the only one with this value, and its real value is thus [19+256, 0+256].

The lattice scales are stored as 6-dimensional floating point vectors. The bit ranges and values for are defined the lattice structures for each mode are presented in Table 5.2.9, and the encoding is then performed as described for EVS.

5.2.2.3.2.4 Fast algebraic codebook search

The existence of a variable bitrate ACELP coding and a very low bitrate available for core coding in some configurations make a demand for algebraic codebooks that support algebraic codebooks sizes which are not part of the EVS codec. Thus, in addition to the algebraic codebooks as described in clause 5.2.3.1.5 in [3], an additional so-called fast algebraic codebook search is present in IVAS.

The fast algebraic codebook search is employed for subframe lengths $L_{subfr} = 64$ samples and $L_{subfr} = 128$ samples at internal sampling length of 12.8 kHz and its supported configurations are summarized in Table 5.2-10.

Table 5.2-10: Fast algebraic codebook configurations

subframe length	bits/subframe
L_subfr = 64	10, 12, 15, 17
L_subfr = 128	12, 14, 18, 20, 24

The structure of fast algebraic codebook is based on interleaved single-pulse permutation (ISSP) design where the pulse positions are divided into several tracks of interleaved positions. The details about the structure including number of pulses and number of tracks is summarized in Table 5.2-11 for codebooks with $L_{subfr} = 64$ samples resp. in Table 5.2-12 for codebooks with $L_{subfr} = 128$ samples.

Table 5.2-11: Structure of fast algebraic codebook search for $L_{subfr} = 64$ samples

bits/subframe	number of pulses	number of tracks	note
10	2	4	even tracks used
12	2	2	all tracks used
15	3	4	first three tracks used
17	3	4	all tracks used

Table 5.2-12: Structure of fast algebraic codebook search for $L_{subfr} = 128$ samples

bits/subframe	number of pulses	number of tracks	note
12	2	4	even tracks used
14	2	2	all tracks used
18	3	4	first three tracks used
20	3	4	all tracks used
24	4	4	all tracks used

The fundamental principle of the fast algebraic codebook search uses a sequential search of the pulses by maximizing a criterion based on a certain reference signal. The optimum fixed codebook gain, filtered target vector and reference signal are then recomputed after each new pulse is determined.

Similar to clause 5.2.3.1.5.9 in [3], let us define a reference signal $b(n)$ as a weighted sum of the signal $d(n)$ and an ideal excitation signal $r(n)$ as

$$b(n) = \sqrt{\frac{E_d}{E_r}} r(n) + \beta d(n) \quad (5.2\text{-}92)$$

where $E_d = \mathbf{d}^T\mathbf{d}$ is the energy of the signal $d(n)$ and $E_r = \mathbf{r}^T\mathbf{r}$ is the energy of the ideal excitation signal $r(n)$. The value of the scaling factor $\beta = 2.0$.

The signal $d(n)$ in (5.2-92) is the backward filtered target vector defined in Equation (524) of [3], i.e.:

$$d(n) = \sum_{i=n}^{L-1} x_{11}(i)h(i-n), n = 0, \ldots, L-1 \quad (5.2\text{-}93)$$

where $x_{11}(n)$ is an updated target signal computed in Equation (523) of [3] and $h(n)$ is the impulse response of the weighted synthesis filter. Then $n, n = 0, \ldots, L-1$, is a time sample index and L is the subframe length L_{subfr} (the subscript is avoided in this clause for simplicity).

Further, the pulse signs are pre-determined using an approach where the sign of a pulse at specific position is set a priori equal to the sign of the reference signal $b(n)$ at that position. That is, there is defined the sign vector $z_b(n)$ containing the signs of the reference signal $b(n)$.

Further, there is employed an autocorrelation method for error minimization in the fixed codebook search procedure in which the matrix of correlations with elements

$$\varphi(i,j) = \sum_{n=j}^{L-1} h(n-i)h(n-j), i = 0, \ldots, L-1, j = i, \ldots, L-1 \quad (5.2\text{-}94)$$

is reduced to a Toeplitz form by modifying the summation limits in Equation (5.2-94) so that $\varphi(i,j) = \alpha(|i-j|)$, where

$$\alpha(n) = \sum_{i=n}^{L-1} h(i)h(i-n) \quad (5.2\text{-}95)$$

is the correlation vector.

Now, the fast algebraic codebook search of M pulses can be summarized in the following steps:

Step I: The backward filtered target vector $d(n)$, the correlation vector $\alpha(n)$, the reference signal $b(n)$, and the sign vector $z_b(n)$ are computed.

Step II: Find the first pulse: the first pulse is found as the index of maximum absolute value of the reference signal $b(n)$. Using the sign vector $z_b(n)$, this can be expressed as

$$m_0 = \text{index}[\max(z_b(n)b(n))] \quad (5.2\text{-}96)$$

$$s_0 = z_b(m_0) \quad (5.2\text{-}97)$$

where m_0 is the index of the first pulse position and s_0 its sign.

Step III: Find following pulses: The other pulses are searched sequentially for $j = 1, \ldots, M-1$. The search of each new pulse starts with computing the fixed codebook gain g_c and the update of the reference signal $b(n)$. The fixed codebook gain for the previously found pulses (pulses m_0, …, m_{j-1}) is given by

$$g_c^{(j-1)} = \frac{g_N^{(j-1)}}{g_D^{(j-1)}} \quad (5.2\text{-}98)$$

where numerator and denominator are expressed as

$$g_N^{(j-1)} = g_N^{(j-2)} + s_{j-1}d(m_{j-1}) \quad (5.2\text{-}99)$$

and

$$g_D^{(j-1)} = g_D^{(j-2)} + \alpha(0) + 2\sum_{i=0}^{j-2} s_i s_{j-1}\alpha(|m_i - m_{j-1}|) \quad (5.2\text{-}100)$$

with the initialization $g_N^{-1} = 0$ and $g_D^{-1} = 0$.

The target signal is then updated by subtracting the contributions of the found pulses from the original target signal $x_{11}(n)$. This can be written as

$$x'_{11}(n) = x_{11}(n) - g_c^{(j-1)} \sum_{i=0}^{j-1} s_i h(n-m_i) \quad (5.2\text{-}101)$$

and substituting $x'_{11}(n)$ from (5.2-101) in (5.2-92) and using (5.2-95) we get an update of the signal $d(n)$, i.e.

$$\acute{d}(n) = d(n) - g_c^{(j-1)} \sum_{i=0}^{j-1} s_i \alpha(|n - m_i|) \quad (5.2\text{-}102)$$

Next, the reference signal $b(n)$ is updated as

$$\acute{b}(n) = \sqrt{\frac{E_d}{E_r}} r(n) + \beta \acute{d}(n) \quad (5.2\text{-}103)$$

and the new pulse m_j is found similarly to (5.2-96) and (5.2-97) as

$$m_j = \text{index}\left[\max\left(z_b(n)\acute{b}(n)\right)\right] \quad (5.2\text{-}104)$$

$$s_j = z_b(m_j) \quad (5.2\text{-}105)$$

Step IV: Compute the fixed codevector and the filtered fixed codevector.

Giving the use of mentioned ISSP design, the steps II to IV are repeated. Let's first suppose that the number of tracks is equal to the number of searched pulses M (one pulse per track) while it is then extended for other configurations. The first iteration is initialized in step II by assigning m_0 to T_0, m_1 to T_0, …, m_{M-1} to T_{M-1}, where T_z is the track number. The procedure is then repeated from step II to step IV by assigning the pulses in the initialization to different tracks and computing equations (5.2-101) to (5.2-105) for $n \in T_z$ only. The number of iterations is equal to M. Finally, the set of pulses selected in the iteration that maximizes the criterion from (526) in [3] is chosen.

5.2.2.3.2.5 AVQ Bit Savings Encoder

5.2.2.3.2.5.1 Overview

Like EVS [3], the time domain or transform-domain coefficients are split into sub-frames, prior to the AVQ quantization. In every sub-frame, a bit budget allocated to the AVQ subframe is composed of a first number of bits which is a sum of a fixed bit-budget and a floating number of bits. AVQ subframes usually does not consume all the allocated bits, leaving a variable number of bits available in each subframe. These bits are floating bits employed in the following AVQ sub-frame. The floating number of bits is equal to 0 in the first sub-frame and the floating bits resulting from AVQ in the last sub-frame in each frame remain unused when coding WB signals or are re-used in coding of upper band. Allocated bits for each AVQ subframe is used for quantizing subframe coefficients, which are split into 8 consecutive sub vectors of 8 coefficients each. The sub-vectors are quantized with an 8-dimensional multi-rate algebraic vector quantizer (AVQ). The parameter of each sub vector sv consist of the codebook number n_{sv} and a corresponding code vector index v_{sv}. The codebook number is in the set of integers {0, 2, 3, 4, 5, 6, 7, 8, ...} and the size of its unary code representation is n_{sv} bits and the size of each index v_{sv} is given by $4n_{sv}$ bits, equalling to $5n_{sv}$ bits per sub-vector with exception of the case of codebook number with the integer 0 that requires 1 bit. Based on the number of bits-available for encoding a sub-vector in the AVQ, one of two encoding methods is selected for encoding the sub-vector. The one is encoding the codebook number of the sub-vector (i.e., the original AVQ encoding [3]), and the other is encoding "unused" bits which is obtained by computing the difference between the number of bits allocated to the AVQ subframe and the number of bits consumed by the AVQ parameters. If the number of bits for encoding the "unused" bits is smaller than the number of bits for encoding the codebook number, the total number of AVQ bits can be reduced and saved. Such condition can be determined using the number of bits available for encoding the sub-vector in the AVQ. Detailed algorithm will be described in the following sub clauses. Figure 5.2-7 shows an example of an AVQ encoder for encoding DCT coefficients. The AVQ bit-saving algorithm is integrated in the code converter block in the figure.

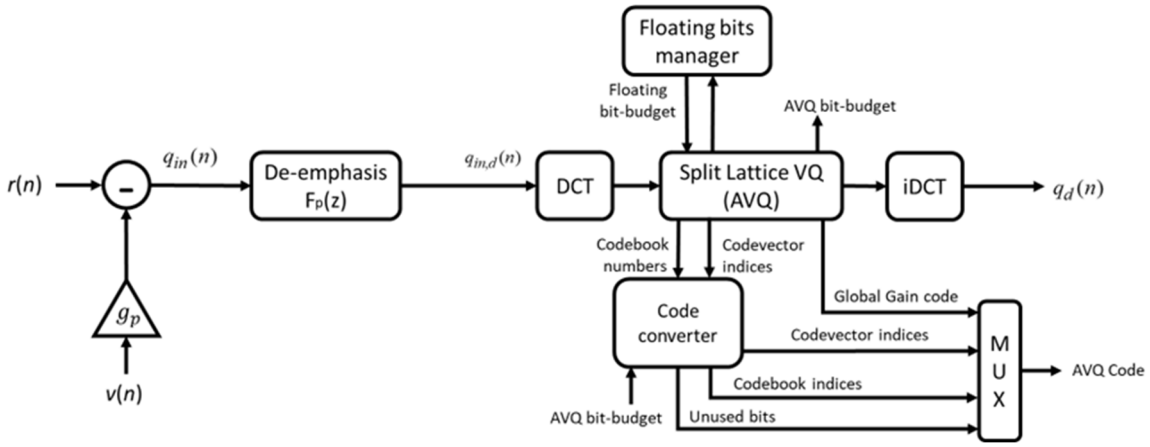


Figure 5.2-7: Overview of AVQ Bit Savings Encoder

5.2.2.3.2.5.2 Code Converter in Encoder

5.2.2.3.2.5.2.1 Fundamentals

Code converter converts a codebook number (indicator) to unused bits based on the information obtained from the AVQ block and the AVQ bit-budget. Figure 5.2-8 shows a basic functional block diagram of the code converter. Codebook number separator selects a codebook number for a predetermined position (see sub-clause Selection of Targeted Sub-vector for details) and inputs to the number of unused bits calculator. The number of usable bits calculator utilizes the remaining $sv's$ and its corresponding parameters to compute the number of usable bits by subtracting the bits necessary to encode the sv parameters. The number of unused bits calculator computes the unused bits by computing the difference between the usable bits obtained from the usable bits calculator and the number of bits needed for coding the pre-determined sv parameters. The number of unused bits encoder encodes the number of unused bits and outputs encoded code to the multiplexer. The number of usable bits may be used for encoding the number of unused bits in some special cases. Although the basic principle is simple, there are many special cases which must be addressed in its implementation. Detailed processing steps are described in the following clause.

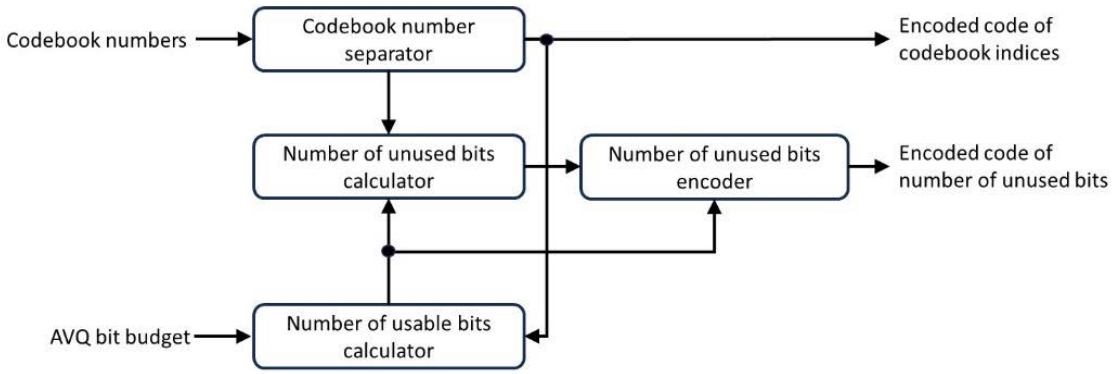


Figure 5.2-8: Functional Block Diagram of Code Converter

5.2.2.3.2.5.2.2 Selection of Targeted Sub-Vector

Figure 5.2-9 shows the steps for selection of targeted sub-vector from a predetermined candidate. In the first stage of the code converter, a targeted sub-vector is selected among the 8 AVQ sub-vectors. There are two predetermined candidates, 3rd or 8th sub-vector. If the AVQ bit budget exceeds threshold Th (i.e., 85 bits), the 3rd sub-vector is selected, otherwise code conversion is applied to the 8th sub-vector parameters.

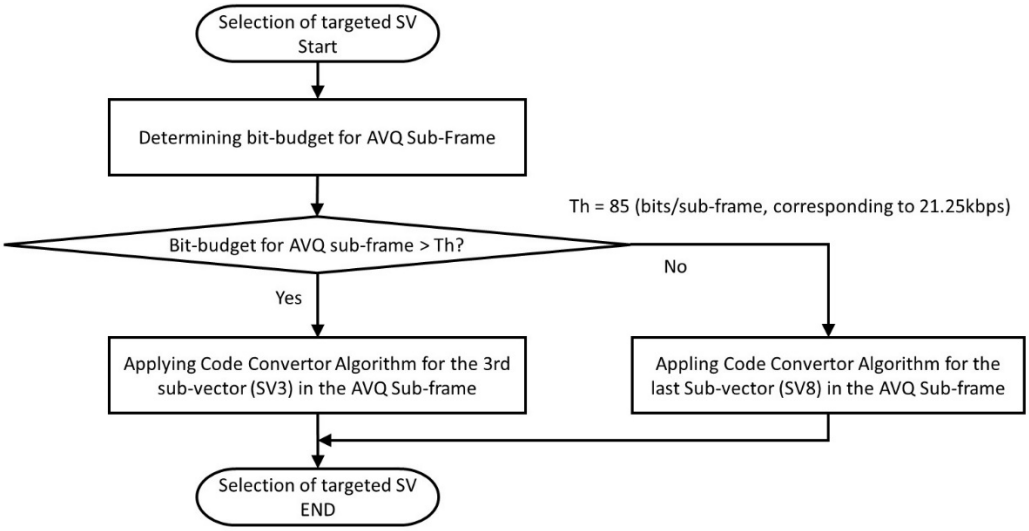


Figure 5.2-9: Selection of Targeted Sub-Vector

5.2.2.3.2.5.2.3 Case of Targeting the 3rd Sub-Vector

Figure 5.2-10 shows code conversion processing flow for the case of targeting the 3rd sub-vector. In the figure, Th1=30 bits, Th2=9 bits and Th3=79 bits.

If the 3rd sub-vector is selected as the targeted sub-vector, the eight sub-vectors are classified into two groups, Group1 and Group2. Group1 contains first two sub vectors, and Group2 contains remaining six sub-vectors from $sv3$ to $sv8$. The number of bits consumed by the Group1 for encoding group 1 parameters is determined according to

$$Bits_{Gp1} = \sum_{j=1}^{2} \begin{cases} 1 & n_{sv}(j) = 0 \\ 5 * n_{sv}(j) & n_{sv}(j) \geq 2 \end{cases} \quad (5.2\text{-}106)$$

where $n_{sv}(j)$ is the code book number for sub-vector of index *j*

Bits available for encoding Group2 $sv's$ is calculated according to

$$Bits_{Gp2} = Bit_{Budget_{AVQ}} - Bits_{Gp1} \quad (5.2\text{-}107)$$

Where $Bit_{Budget_{AVQ}}$ is the bit budget for encoding the AVQ sub-frame which is an integer value.

In the next step, the number of bits available for encoding SVs in Group2 ($Bits_{Gp2}$) is compared to a threshold (Th1). When the $Bits_{Gp2}$ is less than threshold, targeted SV for code conversion is changed from $sv3$ to $sv8$ and encoding of SVs from $sv3\ to\ sv7$ is performed based on the clause 5.2.3.1.6.9 of TS 26.445 [3]. The number of bits available for encoding $sv8$ is calculated based on the bits consumed for encoding $sv3\ to\ sv7$ and from $Bits_{Gp2}$. Code conversion is applied for $sv8$ when the bits available is not less than threshold Th2, instead of encoding codebook index, the number of unused bits is encoded, and encoded information (unused bits index and codevector index) is outputted.

In the case where the number of bits available for encoding SVs in Group2 is not less than 30 bits: Encoding order of SVs in Group2 is determined as sv4 to sv8 and sv3. Encoding of sv4 to sv8 is performed based on the clause 5.2.3.1.6.9 of TS 26.445 [3]. In each encoding cycle, the number of bits used for each encoding is determined, and the number of bits available for encoding remaining SVs in Group2 is updated. The updated number of bits available for encoding remaining SVs in Group2 is compared to a threshold Th1. If the number of bits available for encoding remaining SVs in Group2 is not sufficiently represented, order of encoding for the remaining SVs in Group2 is redetermined as sv3 followed by the other remaining SVs and for other case i.e., sufficient bits available for representing the SVs, sv4 to sv8 is encoded first followed by code conversion to sv3

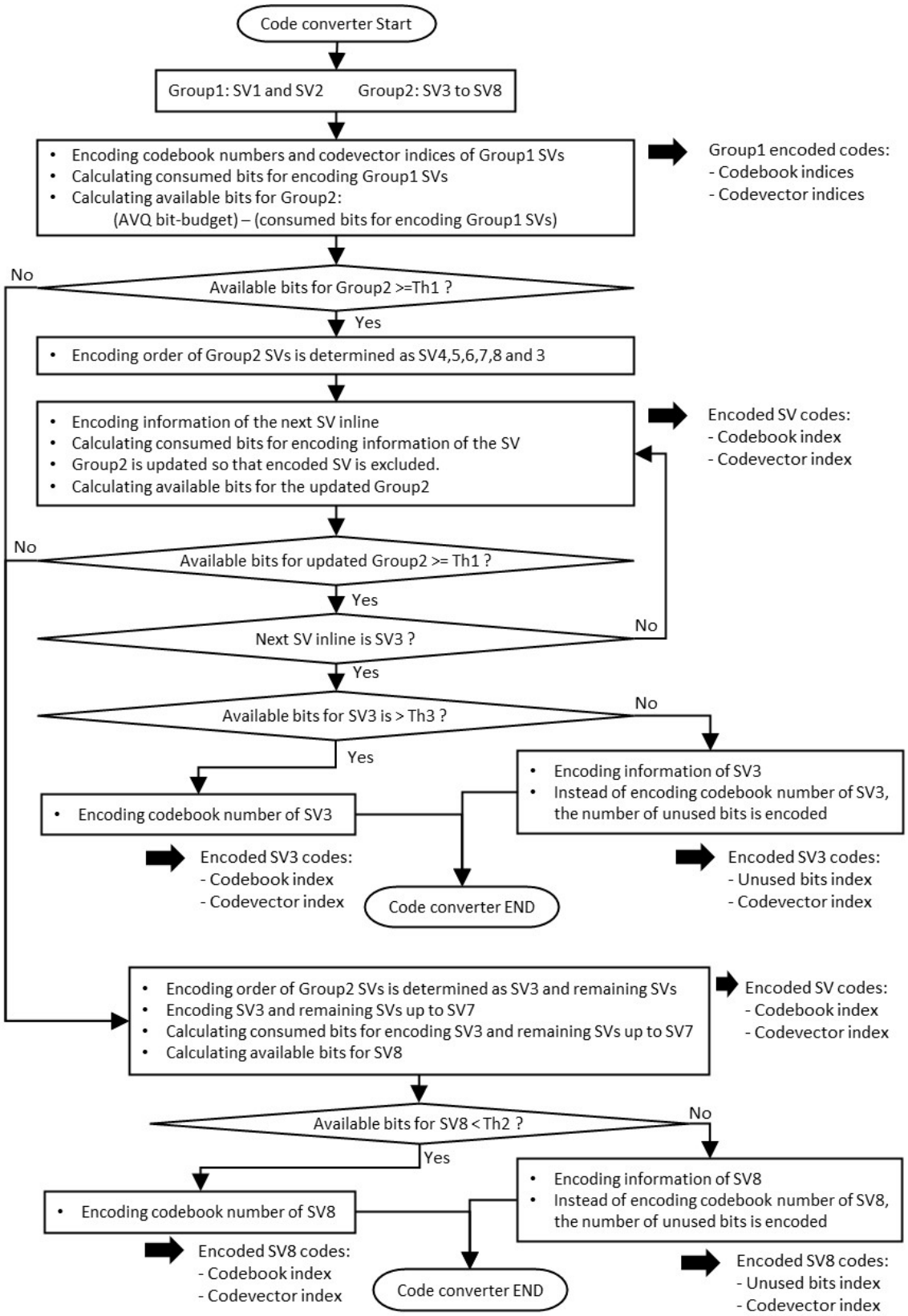


Figure 5.2-10: Flow of code conversion (in the case of targeting SV3)

5.2.2.3.2.5.2.4 Case of Targeting the 8th Sub-Vector

Figure 5.2-11 shows code conversion processing flow for the case of targeting the 8th sub-vector. In the figure, Th2=9 bits and Th3=79 bits.

If the 8th sub-vector is selected as the targeted sub-vector, $sv1\ to\ sv7$ are encoded using AVQ, and encoded information (codebook indices and codevector indices) is outputted. The number of bits available for encoding $sv8$ is calculated according to

$$Bits_{sv8} = Bit_{Budget_{AVQ}} - \sum_{j=1}^{7} \begin{cases} 1 & n_{sv}(j) = 0 \\ 5 * n_{sv}(j) & n_{sv}(j) \geq 2 \end{cases} \quad (5.2\text{-}108)$$

Code conversion is not applied to $sv8$ when fewer bits or extremely large number of bits available for representing $sv8$ (i.e., $Bits_{sv8} < Th2\ or > Th3$), otherwise instead of encoding codebook index, the number of unused bits is encoded, and encoded information (unused bits index and code vector index) is outputted.

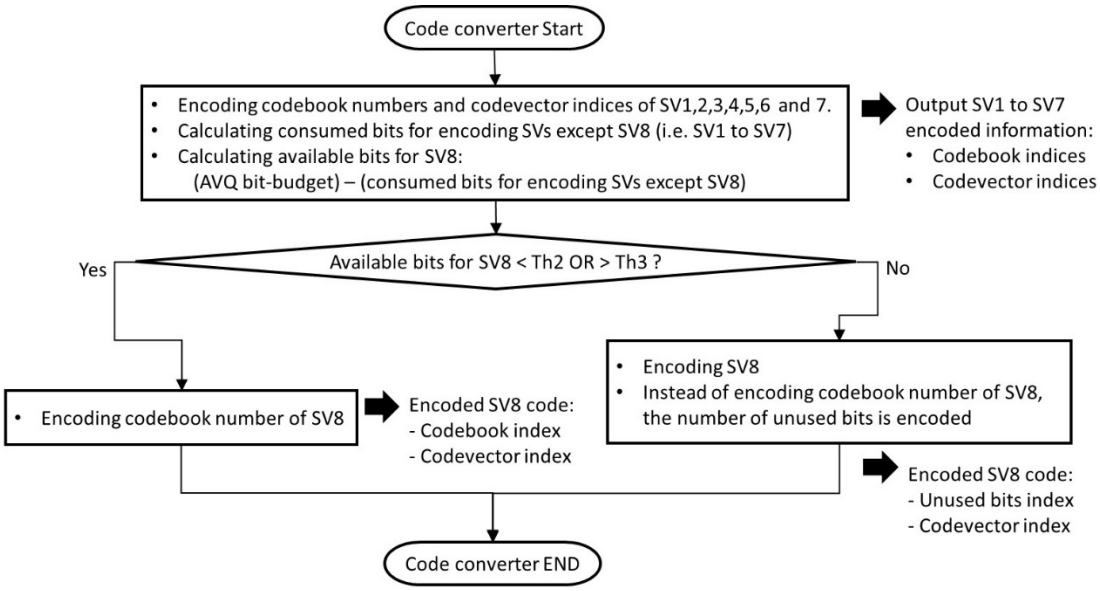


Figure 5.2-11: Flow of code conversion (in the case of targeting SV8)

5.2.2.3.2.5.3 Unused Bit Encoding

Figure 5.2-12 shows a flow diagram of the number of unused bits encoding. In the figure, sv_d represents a targeted SV, which is either of sv_3 or sv_8. For encoding of unused bits, Remainder bits, RB, is computed according to

$$RB = Bits_{svd} \,\%\, 5 \quad (5.2\text{-}109)$$

Before computation of unused bits for the sv_d, one overflow bit is added to $Bits_{svd}$ when overflow is detected (i.e., RB=4), for the remaining possible values of RB, RB bits are subtracted from the usable bits ($Bits_{svd}$) before computing unused bits for the sv_d

$$bits_{unused} = Bits_{svd} - 5 * n_{sv}(d), d = 3\ or\ 8 \quad (5.2\text{-}110)$$

Finally, the number of unused bits is encoded, and the encoding process is completed.

For the case of sv_d is sv_3, due to re-ordering of sub-vectors some of the sub-vectors in Group2 with 0 value is also encoded which may lead to inadequate bits for representing sv_d, to avoid those cases following additional operations are performed before encoding unused bits.

$$Condition1:\ NCNV > 0\ (or, quantized\ SV8\ becomes\ null\ vector.)$$
$$Condition2:\ RB + NCNV \geq 4$$

Where NCNV is the number of consecutive null vectors.

When the above condition detect possibility of wasting bit(s) because of encoding null vector(s), the number of available bits ($\mathrm{Bits}_{\mathrm{svd}}$) for encoding sv_3 is updated by additionally adding $(5 - \mathrm{RB})$ bits, or NCNV bits, or $(\mathrm{NCNV} + 1)$ bits. Number of bits added in this process depends on the result of $(\mathrm{RB} + \mathrm{NCNV})\ \%5$ as shown in Figure 5.2-13

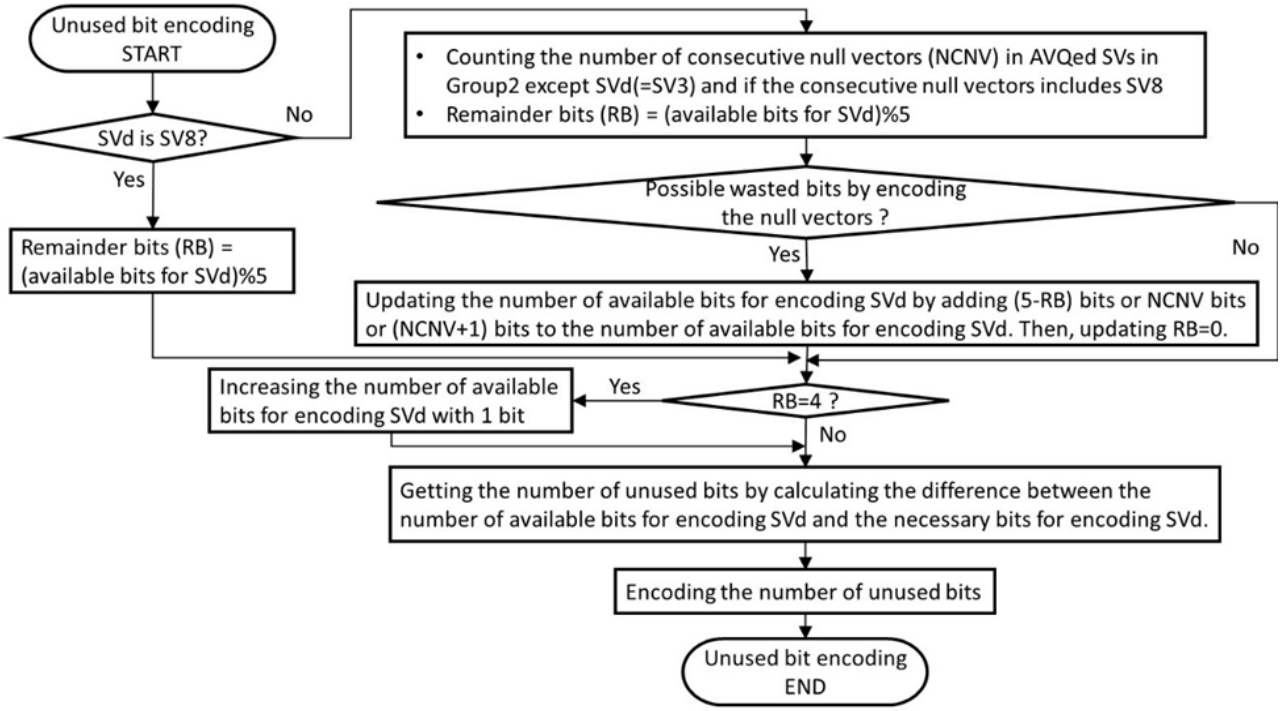


Figure 5.2-12: Flow of Encoding the number of unused bits

Finally, the number of unused bits is encoded according to Table 5.2.2-1, and the encoding process is completed. This table can be customized based on the number of usable bits, meaning codes for unused bits exceeding the usable bits are not necessary and may be removed, and the stop bit "0" of the longest codeword for the unused bit can be omitted. To ensure bit efficiency, typically this stop bit omitted code is used when the codebook indicator is 0 (i.e codebook is Q0), because the unused bits become the largest in that case.

Table 5.2.2-1: Correspondence between number of unused bits and an unused bit encoding code

Unused bits	0-4	5-9	10-14	15-19	..
Codeword	10	0	110	1110	..
Number of bits	2	1	3	4	

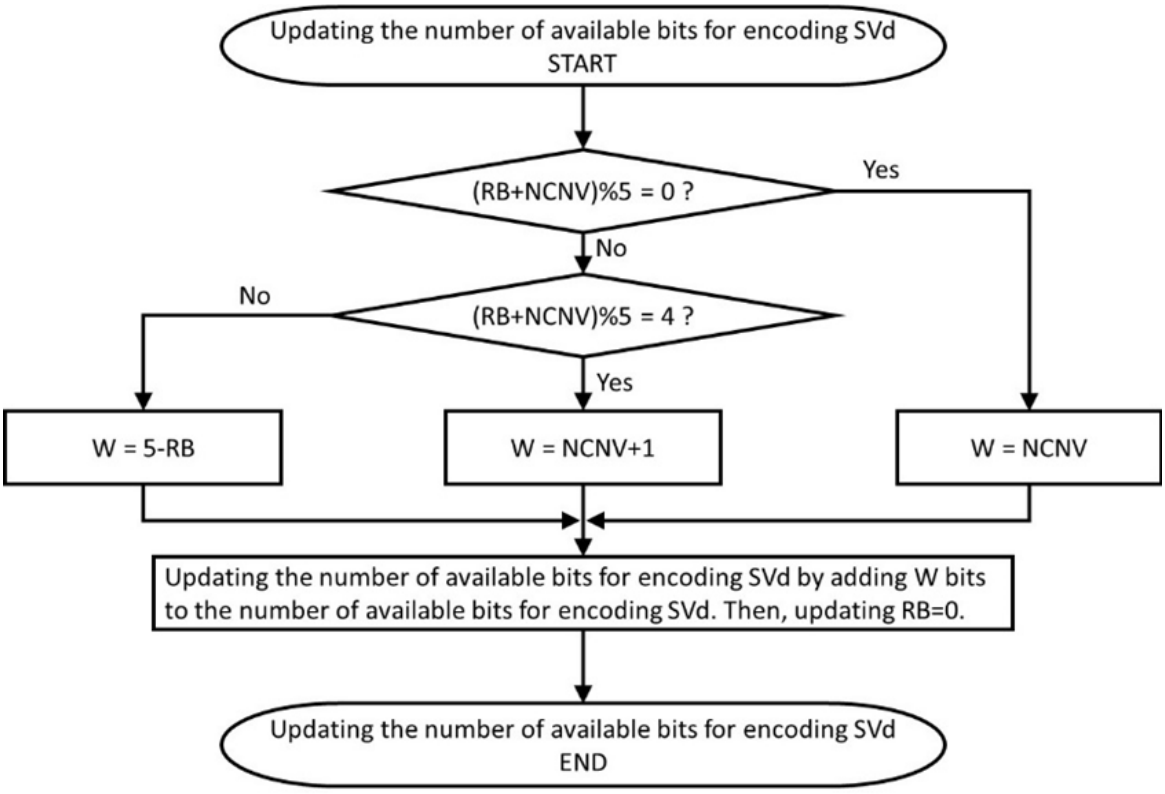


Figure 5.2-13: Process of updating the number of available bits for encoding SV3

5.2.2.3.2.6 GSC core coding

5.2.2.3.2.6.1 GSC encoder for signals that are classified as neither speech nor music

When the output class of the IVAS S/M classifier d_{SMC} defined in eq. (5.2-43) is UNCLEAR it means that the classifier is unsure whether the signal in the current frame shall be encoded with the time-domain encoder or the frequency-domain encoder. In such case the GSC encoder is selected and optimized accordingly. The GSC encoder is described in clause 5.2.3.5 of [3]. As part of the GSC encoder one of four sub-modes is selected and the selection is transmitted to the decoder using 2 bits. Let the selected sub-mode of the GSC encoder be denoted as F_{tfsm}.

The selection of the GSC sub-mode is done as follows. If the available bit rate is below or equal to 9.2 kbps or if the input format is MONO, then EVS-based GSC encoder is selected, and the input signal is encoded using the method described in clause 5.2.3.5 of [3]. When EVS-based GSC encoder is selected the index F_{tfsm} is set to 0. Otherwise, one of the three GSC sub-modes is selected based on the following logic. Note, that each of the three GSC sub-modes is derived from the EVS-based GSC encoder, described in clause 5.2.3.5 of [3], with various optimizations described in detail below.

The first GSC sub-mode. for which F_{tfsm} is set to 1. is designed to improve the encoding of segments with weak speech characteristics that wouldn't not be optimally encoded with the ACELP encoder. In the first GSC sub-mode the number of subframes for the time-domain contribution of the excitation signal varies depending on the bit budget available. If the available bit budget is less than 15 kbps, then the time-domain contribution of the excitation signal will use two subframes each having the length of 128 samples. If the available bit budget is higher or equal to 15 kbps four subframes will be used, each having the length of 64 samples.

The second GSC sub-mode, for which F_{tfsm} is set to 2, is used in frames with high likelihood of onset. Similarly, to the first GSC sub-mode, if the available bit budget is lower than 15 kbps, then the time-domain contribution of the excitation signal uses two subframes each of which has the length of 128 samples. If the available bit budget is higher or equal to 15 kbps four subframes will be used, each having the length of 64 samples. Furthermore, the frequency-domain contribution of the excitation signal is more focused on the low-frequency bands.

Note, that the time-domain contribution of the excitation signal in the first and the second GSC sub-mode consists of the algebraic part and the fixed part in each encoded subframe.

The third sub-mode, for which F_{tfsm} is set to 3 is selected when the speech-music classifier points to a signal having certain music characteristics. In such case, the time-domain contribution of the excitation signal uses four subframes each having the length of 64 samples. The time-domain contribution of the excitation signal is made by the algebraic part only. The fixed part is omitted. The bits that would normally be allocated to the fixed part of the excitation signal are conceded to the frequency-domain contribution. The decision tree in Figure 5.2-14 describes the mechanism for the selection of the GSC sub-mode.

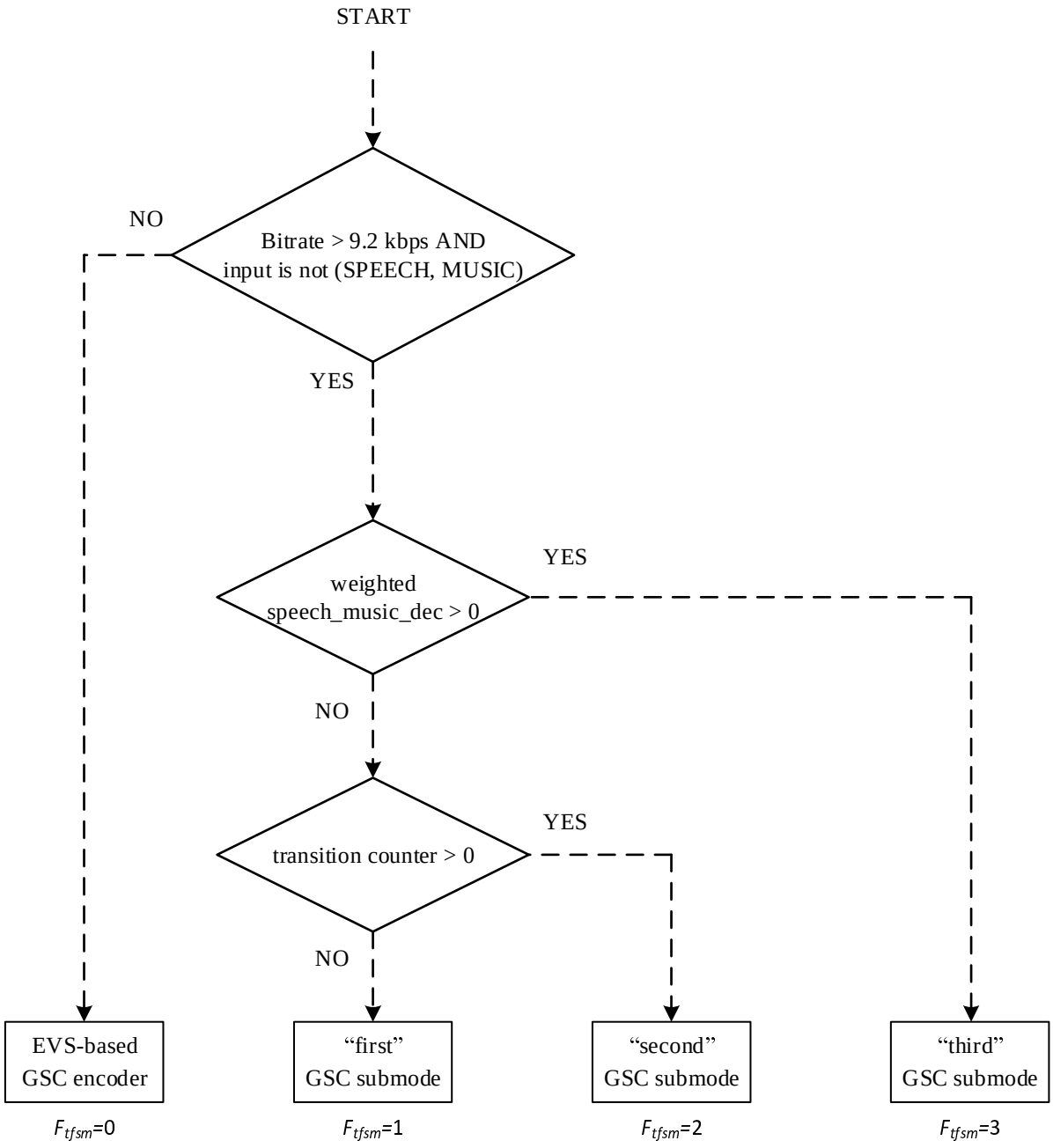


Figure 5.2-14: GSC sub-mode selection

5.2.2.3.2.6.2 Excitation coding in GSC sub-modes

The encoding of the excitation signal in segments that are classified by the speech/music classifier as neither speech nor music is done with one of the GSC sub-modes described in clause 5.2.2.3.2.6. The excitation coding is based on the same technology from the EVS codec, described in clause 5.2.3.5 of [3]. Note, that the GSC coder combines the time-domain and frequency-domain approach to excitation coding. The modifications introduced to the EVS-based GSC encoder in any of the GSC sub-modes have been designed to optimize the quality of input signals that are neither

speech nor music. The modifications to the EVS-based GSC encoder are shown in the schematic diagram in Figure 5.2-15.

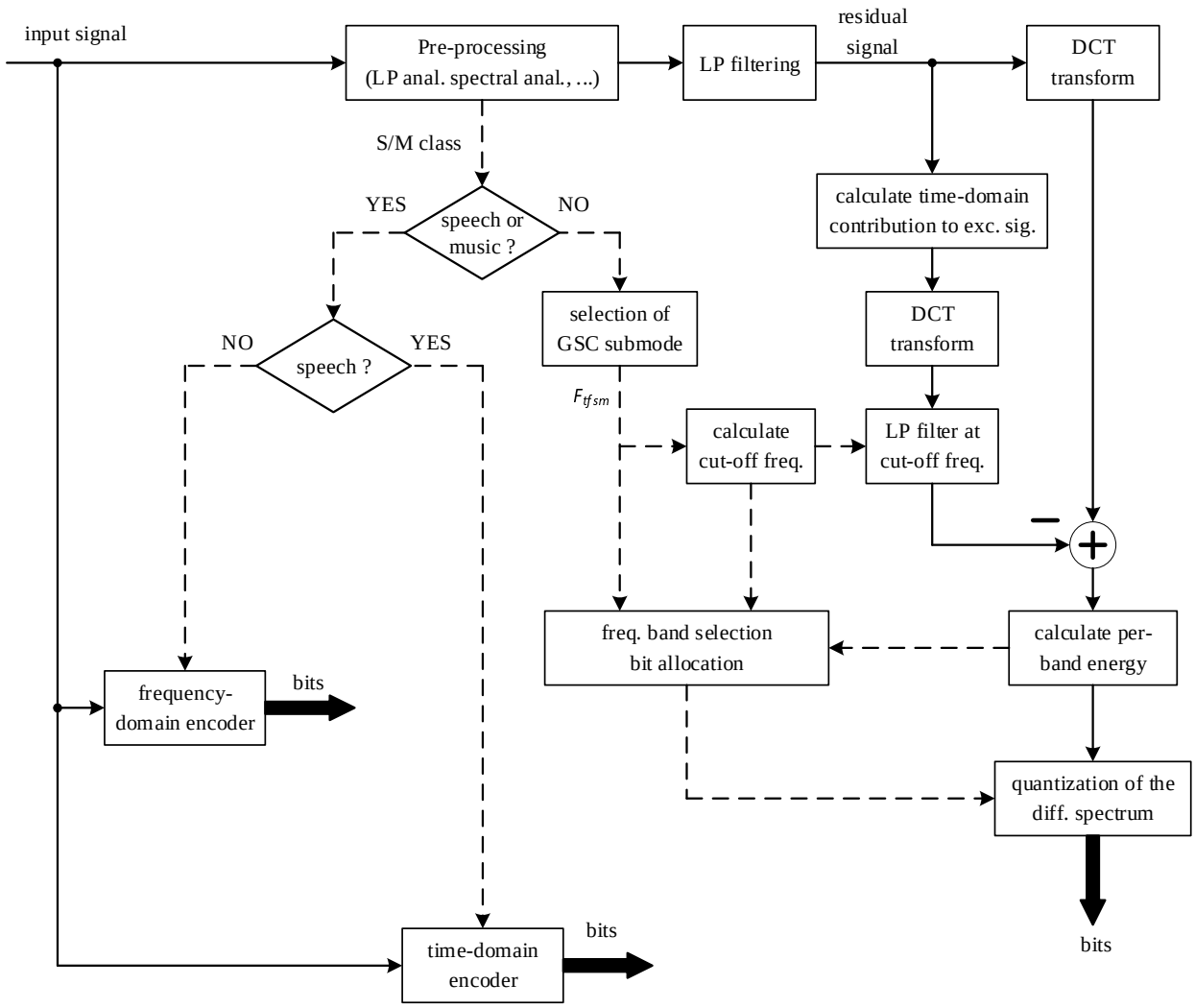


Figure 5.2-15: Modifications to the EVS-based GSC encoder

The time-domain contribution to the excitation signal is calculated using 1, 2 or 4 subframes using the adaptive codebook part and possibly also the fixed codebook part. Then, a cut-off frequency for the time-domain contribution is calculated based on the selected GSC sub-mode. The calculation of the cut-off frequency is done as described in clause 5.2.3.5.6 of [3]. The first estimate of the cut-off frequency, f_{tc1}, is calculated as per equation (648) of [3]. When one of the GSC sub-modes is selected with $F_{tfsm} > 0$, the average normalised correlation $\bar{C}_{c2}$, calculated in equation (646) of [3], is doubled before determining the first estimate of the cut-off frequency f_{tc1}. This is done to increase the weight of the time-domain contribution.

The frequency spectrum of the residual signal and the frequency spectrum of the time-domain contribution of the excitation signal is partitioned into spectral bands as defined in equation (642) of [3]. This results in the following set of frequency bins, $L_f(i)$, for $i = 0, \ldots, 15$ where each value corresponds to the last frequency bin of the i −th frequency band. That is

$$L_f(i) \in (175,375,775,1175,1575,1975,2375,2775,3175,3575,3975,4375,4775,5175,5575,6375) \quad (5.2\text{-}111)$$

The minimum value of the cut-off frequency is equivalent to the last frequency bin of the 7th band which is 2775 Hz. The cut-off frequency is then set as the maximum of the following three values

$$f_{tc} = \max(L_f(i_8), 2775, f_{tc1}) \quad (5.2\text{-}112)$$

where i_8 is the index of the 8th pitch harmonic as described in clause 5.2.3.5.6 of [3]. Therefore, the value $L_f(i_8)$ is the last frequency of the frequency band in which the 8th pitch harmonic is located. The cut-off frequency f_{tc} is quantized with a 4-bit quantizer using the procedure described in clause 5.2.3.5.6 of [3].

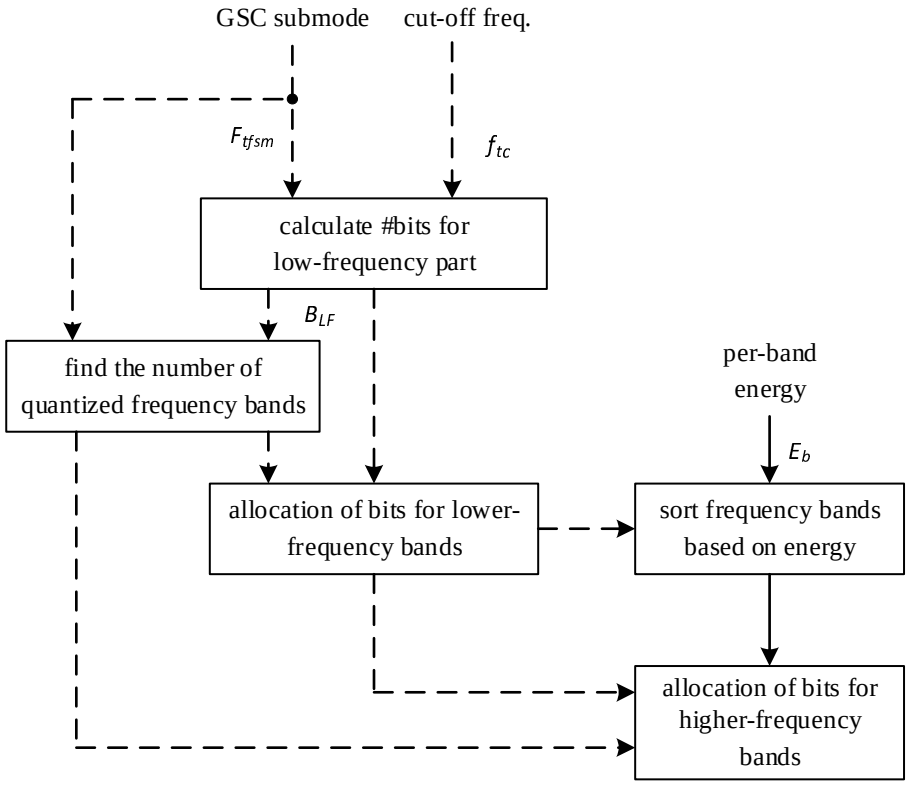


Figure 5.2-16: Bit allocation mechanism for the frequency-domain part of the GSC encoder

After the time-domain contribution has been computed and quantized in one of the GSC sub-modes the remaining bit budget is allocated to the frequency-domain quantization. The mechanism of bit allocation is depicted in the schematic diagram in Figure 5.2-16. The quantization of the differential residual spectrum is performed in a per-band manner. For the purposes of quantization, the residual spectrum is partitioned into 16 frequency bands each of which has 16 frequency bins.

The spectral content is divided into the lower part containing the first five spectral bands (up to 2 kHz) and the upper part. The available bitrate is distributed unequally among the lower part and the upper part. The parameter P_{Blf} represents the ratio of bits allocated to the lower part of the spectrum. The parameter P_{Blf} is calculated based on the quantized cut-off frequency $\hat{f}_{tc}$. That is

$$P_{Blf0} = \frac{1}{100}\left(-0.125 \cdot \frac{L_f(\hat{f}_{tc})}{25} + 76\right) \quad (5.2\text{-}113)$$

where $L_f(\hat{f}_{tc})$ is the quantized value of the last frequency bin corresponding to the cut-off frequency. The ratio P_{Blf0} is then limited to the interval between 0.5 and 0.75 using the following relation

$$P_{Blf} = \max\left(min\left(P_{Blf0}, 0.75\right), 0.5\right) \quad (5.2\text{-}114)$$

The number of bits allocated to the lower part is then calculated as

$$B_{LF} = P_{Blf} \cdot B_F \quad (5.2\text{-}115)$$

where B_F represents the number of bits available for the quantization of the frequency-domain contribution.

The portion of bits allocated to the lower part in equation (5.2-115) is then adjusted based on the GSC sub-mode flag F_{tfsm}. In case of the first GSC sub-mode, where F_{tfsm} is equal to 1, there is a high likelihood that the current frame contains a temporal attack with a complex spectral shape. In such case the portion of bits allocated to the lower part is increased by 10%. In case of the second GSC sub-mode, where F_{tfsm} is equal to 2, the spectral content might contain

regular harmonics with simple spectral shape. In such case the portion of bits allocated to the lower part is decreased by 10%.

Another parameter affecting the distribution of bits between the lower part and the upper part of the differential residual spectrum is the maximum number of quantized frequency bands, denoted as N_{Bmx}. At the internal sampling rate of 12.8 kHz the total number of frequency bands is $N_{tt} = 16$. The maximum number of quantized frequency bands N_{Bmx} is initialized to 10. In case of the first GSC sub-mode, where F_{tfsm} is equal to 1, the maximum number of quantized frequency bands N_{Bmx} is set to 9. In case of the second GSC sub-mode, where F_{tfsm} is equal to 2, the maximum number of quantized frequency bands N_{Bmx} is set to 13.

The maximum number of quantized frequency bands is then adjusted using a factor N_{Badj} which is calculated based on the total bitrate available for the encoding of the input signal of the core-coder B_T. The adjustment factor is calculated as follows

$$N_{Badj} = \begin{cases} 0.0125 \cdot B_F - 0.75 & F_{tfsm} = 1 \text{ AND } B_T < 15000, \\ 0.02 \cdot B_F - 1.2 & F_{tfsm} \neq 2 \text{ AND } B_T > 20000, \\ 1 & \text{otherwise} \end{cases} \quad (5.2\text{-}116)$$

where B_T is the total bitrate available to the GSC encoder.

The maximum number of quantized frequency bands is then adjusted with

$$N_{Bmx} \leftarrow \max\left(\min\left(\left\lfloor N_{Bmx} \cdot N_{Badj} + 0.5 \right\rfloor, N_{tt}\right), 5\right) \quad (5.2\text{-}117)$$

Note, that the $\lfloor \cdot \rfloor$ function cuts the fractional part of the value, rounding it to the nearest lower integer value.

The maximum number of quantized frequency bands N_{Bmx} may be further modified by considering the amount of bits needed for the quantization of high-frequency bands. Since important information is usually present in the mid-frequency band the first band following the last low-frequency band, i.e. the 6th band, has a similar number of bits as the last low-frequency band, denoted as m_b.

Let the desired number of bits in the last encoded frequency band be denoted as m_p. The desired number of bits m_p is initialized to the value of 4.5. This is sufficient for the encoding of a single spectral peak per frequency band. If the available bitrate B_T is greater or equal to 15 kbps, then the desired number of bits m_p is set to 9 allowing for the encoding of two spectral peaks per frequency band. However, if the available bitrate B_T is below 15 kbps and the third GSC sub-mode has been selected, then the desired number of bits m_p is set to 6.75, allowing the encoding of a single spectral peak with higher precision. The adjusted maximum number of quantized frequency bands is then calculated as follows

$$N'_{Bmx} = \min\left(N_{Bmx}, 5 + \frac{B_F - P_{Blf} \cdot B_F}{0.5 \cdot (m_p + m_b)}\right) \quad (5.2\text{-}118)$$

where the constant 5 represents the minimum number of quantized frequency bands. After the calculation of the maximum number of quantized frequency bands N_{Bmx} and the its adjusted value N'_{Bmx} the GSC encoder ensures that m_p remains lower than or equal to m_b. This is done as follows

$$m_p \leftarrow \min\left(m_p, m_b\right) \quad (5.2\text{-}119)$$

If the adjusted maximum number of quantized frequency bands N'_{Bmx} is smaller than the initially calculated value of N_{Bmx} some bits are re-allocated from the higher-frequency band to the lower-frequency band. This is done as follows

$$B_{LF} = P_{Blf} \cdot B_F + \left(0.5 \cdot \left(m_p + m_b\right) \cdot \left(N_{Bmx} - N'_{Bmx}\right)\right) \quad (5.2\text{-}120)$$

where B_{LF} corresponds to the number of bits allocated to the lower part of the freqeuncy spectrum, i.e. the five lower-frequency bands.

Once the bit distribution between the lower part and the higher part of the frequency spectrum is finished the frequency bands are sorted in decreasing order based on their importance. This is done in two stages. In the first stage, higher-frequency bands having lower energy than both their neighbouring bands are identified. The identified frequency bands will receive only a minimum amount of bits even if the available bit budget is high. This is done as follows

$$P_{pb}(i) = \begin{cases} 1 & \text{if } E(i-1) > E(i) < E(i+1) \quad \text{for } i = 6, \ldots, N'_{Bmx} - 2 \\ 2 & \text{otherwise} \end{cases} \quad (5.2\text{-}121)$$

where $E(i)$ denotes the per-band energy and the value of $P_{pb}(i)$ indicates the importance of the *i*th band. The value of "1" serves as an indicator of energy dips and the value of "2" serves as an indicator of a energy peaks.

In the second stage, higher-frequency bands are sorted in decreasing order based on their energy. The sorting of higher-frequency bands can be described with an iterative algorithm. Let us denote the indices of the sorted array of energy values as $E_{P_{max}}(j)$, where $j = 0, \dots, N'_{Bmx} - 1$. Since the sorting process is done only on higher-frequency bands the indices corresponding to the lower-frequency bands are set as follows

$$E_{P_{max}}(j) = j \qquad \text{for } 0 \le j < 5 \tag{5.2-122}$$

The iterative sorting process starts by setting j to 5 and repeating the following pseudo-code until $j = N'_{Bmx} - 1$.

$$\begin{aligned} &i_{max} = \arg\max\big(E(i)\big) \quad \text{where } i \in \langle 5, \dots, N'_{Bmx} - 1\rangle \\ &E_{P_{max}}(j) = i_{max} \\ &E(i_{max}) = 0 \\ &j \leftarrow j + 1 \end{aligned} \tag{5.2-123}$$

The bits allocated to the lower part of the frequency spectrum B_{LF} are distributed among the first five lower-frequency bands in a linearly decreasing order. The first band receives 23% of B_{LF} and the very last band receives 17% of B_{LF}. The per-band bit distribution in the lower part can be described using the following relation

$$B_p(i) = (0.23 - i \cdot 0.015) \cdot B_{LF} \qquad \text{for } 0 \le i < 5 \tag{5.2-124}$$

Then, small amount of bits is allocated to the frequency bands having lower energy than both their neighbouring bands. This is done with

$$B_p(i) = m_p \qquad \text{for } 5 \le i \le N'_{Bmx} \text{ where } P_{pb}(i) = 1 \tag{5.2-125}$$

The remaining bits are then distributed over the rest of the higher-frequency bands in a linearly decreasing order but considering the ordering of bands based on their energy. This is done to allocate more bits to the higher-energy bands and less bits to the lower-energy bands. The bit allocation in the rest of the higher-frequency bands can be described with

$$\begin{gathered} B_p\left(E_{P_{max}}(j)\right) = \left(\frac{2\cdot(B_F - B_{LF})}{N'_{Bmx}-5}\right) - m_p - \left(\frac{2\cdot(B_F - B_{LF}) - 2\cdot m_p\cdot\left(N'_{Bmx}-5\right)}{\left(N'_{Bmx}-5\right)^2}\right)\cdot\left(E_{P_{max}}(j) - 5\right) \\ \text{for } j = 5, \dots, N'_{Bmx} - 1 \text{ where } P_{pb}\left(E_{P_{max}}(j)\right) = 2 \end{gathered} \tag{5.2-126}$$

In case there are some remaining bits after the bitrate allocation process above is completed, these bits will be added to the bitrate of the lower-frequency bands. The re-allocation of the remaining bits is done iteratively by adding one bit per frequency band. The re-allocation process starts from the 5th band and goes backwards to the first band. The re-allocation process is repeated until all remaining bits are spent. After the bit allocation process the per-band bit distribution $B_p(.)$ and the order of sorted frequency bands $E_{P_{max}}(.)$ are both passed to the GSC quantizer described in clause 5.2.3.5.9 of [3].

5.2.2.3.2.7 Bandwidth extension in time domain

5.2.2.3.2.7.1 Overview

The time-domain bandwidth extension (TBE) module codes the signal content beyond the range of frequencies that are coded by the low band core encoder. In the IVAS codec the TBE module is based on the identically titled module from the EVS codec described in detail in clause 5.2.6.1 of [3]. This clause describes only the details of the SWB TBE encoder which has been optimized for the coding of cross-talk content.

In immersive conversational scenarios the IVAS codec often processes input signals with cross-talk content. Since the SWB TBE technology in the EVS codec has been optimized mainly for single-talk content the quality of cross-talk speech may be severely impacted. In the SWB TBE encoder the high-band excitation signal is not encoded directly. Instead, the low-band excitation signal (0 – 8 kHz), calculated in the ACELP coder, is upsampled and extended up to 16 kHz and used as a replacement of the high-band excitation signal. For example, if there the spectral characteristics of the low-band signal are predominantly voiced and the characteristics of the high-band signal are predominantly unvoiced the synthesized signal might be distorted.

5.2.2.3.2.7.2 Low-band excitation signal

The low-band excitation signal, calculated in the ACELP core, is transformed with a non-linear processor as described in clause 5.2.6.1.7 in [3]. The non-linearly transformed excitation signal in time domain is then spectrally flipped as defined in eq. (701) in [3] reproduced here for convenience

$$\varepsilon_{flipped}(n) = (-1)^n \varepsilon_{NL}(n), \quad \text{for } n = 1, \dots, N_{32k} - 1 \tag{5.2-127}$$

where $\varepsilon_{NL}(n)$ is the non-linearly transformed excitation signal and N_{32k} is the frame length. Note that both the non-linearly transformed excitation signal $\varepsilon_{NL}(n)$ and the flipped signal $\varepsilon_{flipped}(n)$ are sampled at 32 kHz where $N_{32k} = 640$.

The flipped signal is then decimated by 2 with a series of two all-pass filters as described in clause 5.2.6.1.9. The output signal, sampled at 16 kHz, is denoted as

$$\varepsilon_{16k}(n), \quad \text{for } n = 1, \dots, N - 1 \tag{5.2-128}$$

where $N = 320$ is the frame length.

The high band target signal is generated using the procedure described in clause 5.2.6.1.1 of [3]. Let the high-band target signal be denoted as

$$s_{HB}(n), \quad \text{for } n = 1, \dots, N - 1 \tag{5.2-129}$$

5.2.2.3.2.7.3 High-band residual signal

The high-band target signal is generated by the SWB TBE encoder as described in clause 5.2.6.1.1 in [3]. The high-band target signal is sampled at 16 kHz regardless of the bitrate of the ACELP core and its spectral content is flipped. Therefore, the first bin of the high-band target spectrum corresponds to the last bin of the spectrum of the input signal and vice-versa.

Following the procedure described in clause 5.2.6.1.1 of [3] LP filter coefficients are estimated from the high-band target signal in four consecutive subframes where each subframe has the length of 80 samples. The high-band LP filter coefficients are calculated using the Levinson-Durbin algorithm. Let's denote the high-band LP filter coefficients as

$$a_j^{HB}(k), \quad k = 0, \dots, P, \ j = 0, \dots, 3 \tag{5.2-130}$$

where $P = 10$ is the order of the high-band LP filter and $j = 0, \dots, 3$ is the subframe index. Note, that the first LP coefficient in each subframe is unitary, i.e. $a_j^{HB}(0) = 1$.

The high-band residual signal is then generated by filtering the high-band target signal with the high-band LP filter. This is done as follows

$$r_{HB}(n + j \cdot 80) = s_{HB}(n + j \cdot 80) - \sum_{k=1}^{P} a_j^{HB}(k) s_{HB}(n + j \cdot 80 - k), \quad j = 0, \dots, 3, \ n = 0, \dots, 80 - 1 \tag{5.2-131}$$

Note, that the first *P* samples of the high-band residual signal are calculated from the high-band target signal in the previous frame. This is indicated by the negative index in $s_{HB}(-k), k = 1, \dots, P$ in the summation term. The negative indices refer to the samples of the high-band target signal from the end of the previous frame.

The high-band residual signal calculated in eq. (5.2-131) is used to calculate the high-band autocorrelation function and the high-band voicing factor. The high-band autocorrelation function is not calculated directly on the high-band residual signal. Direct calculation of the high-band autocorrelation function requires significant computational resources. Furthermore, the dynamics of the high-band residual signal is generally low and the spectral flipping process often leads to smearing the differences between voiced and unvoiced signals. To avoid these problems the high-band autocorrelation function is estimated on the temporal envelope of the high-band residual signal in the downsampled domain.

The temporal envelope of the high-band residual signal is calculated by filtering the high-band residual signal with a sliding moving-average (MA) filter with *M* = 20 taps. This is done as follows

$$R_{TD}(n) = \frac{1}{M} \sum_{k=-M/2}^{M/2} |r_{HB}(n + k)|, \quad n = 0, \dots, N - M - 1 \tag{5.2-132}$$

where the negative samples $k = -M/2, \dots, -1$ in $r_{HB}(k)$ refer to the values of the high-band residual signal in the previous frame. In mode-switching scenarios it may happen that the high-band residual signal in the previous frame is not calculated and the values are unknown. In that case, the first $M/2$ values of $r_{HB}(k)$ are replicated and used as a replacement of the samples $k = -M/2, \dots, -1$ of $r_{HB}(k)$. The last M values of $R_{TD}(n)$ in the current frame are approximated by means of IIR filtering. This is done as follows

$$R_{TD}(n) = 0.05 \cdot r_{HB}(n) + 0.95 \cdot R_{TD}(n-1), \quad n = N-M, \dots, N-1 \tag{5.2-133}$$

The calculation of the temporal envelope of the high-band residual signal is illustrated in Figure 5.2-17.

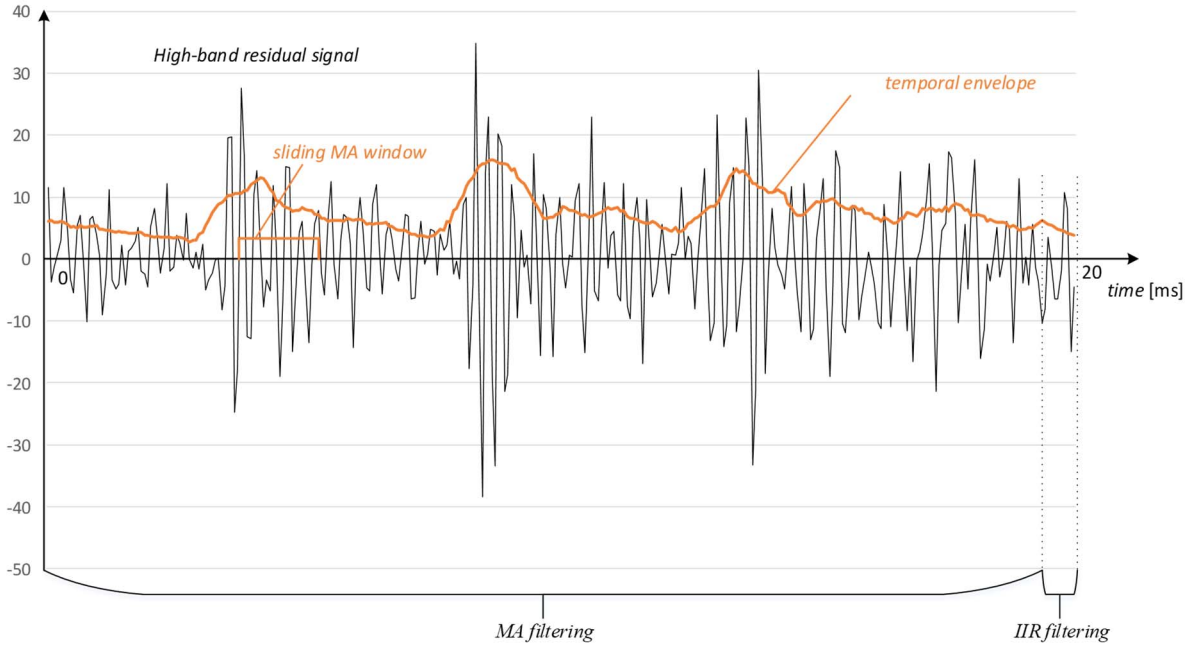


Figure 5.2-17: Temporal envelope of the high-band residual signal

The temporal envelope is then down-sampled by a factor of 4. This is done as follows

$$R_{4kHz}(n) = R_{TD}(4n), \quad n = 0, \dots, \frac{N}{4} - 1 \tag{5.2-134}$$

The down-sampled temporal envelope is then divided into four consecutive segments and the mean value of each segment is calculated as follows

$$\bar{R}_k = \frac{16}{N} \sum_{n=0}^{\frac{N}{16}-1} R_{4kHz}\left(k \cdot \frac{N}{16} + n\right), \quad k = 0, \dots, 3 \tag{5.2-135}$$

where *k* is the index of the segment.

The mean values are all limited by the maximum value of 1.0. The mean values are then used to calculate the segmental normalization factors. This is done with

$$\eta_k = \frac{1}{\bar{R}_k}, \quad k = 0, \dots, 3 \tag{5.2-136}$$

The segmental normalization factors are linearly interpolated in the entire interval of the current frame in the following way

$$\begin{aligned} &\gamma(n) = \eta_{-1} + \frac{32n}{N} \cdot (\eta_0 - \eta_{-1}), && n = 0, \dots, \frac{N}{32} - 1 \\ &\gamma\left(\frac{N}{32} + k \cdot \frac{N}{16} + n\right) = \eta_{k-1} + \frac{16n}{N} \cdot (\eta_k - \eta_{k-1}), && k = 0, \dots, 2, n = 0, \dots, \frac{N}{16} - 1 \\ &\gamma(n) = \eta_3, && n = \frac{7N}{32}, \dots, \frac{N}{4} - 1 \end{aligned} \tag{5.2-137}$$

The interpolation process is illustrated in Figure 5.2-18.

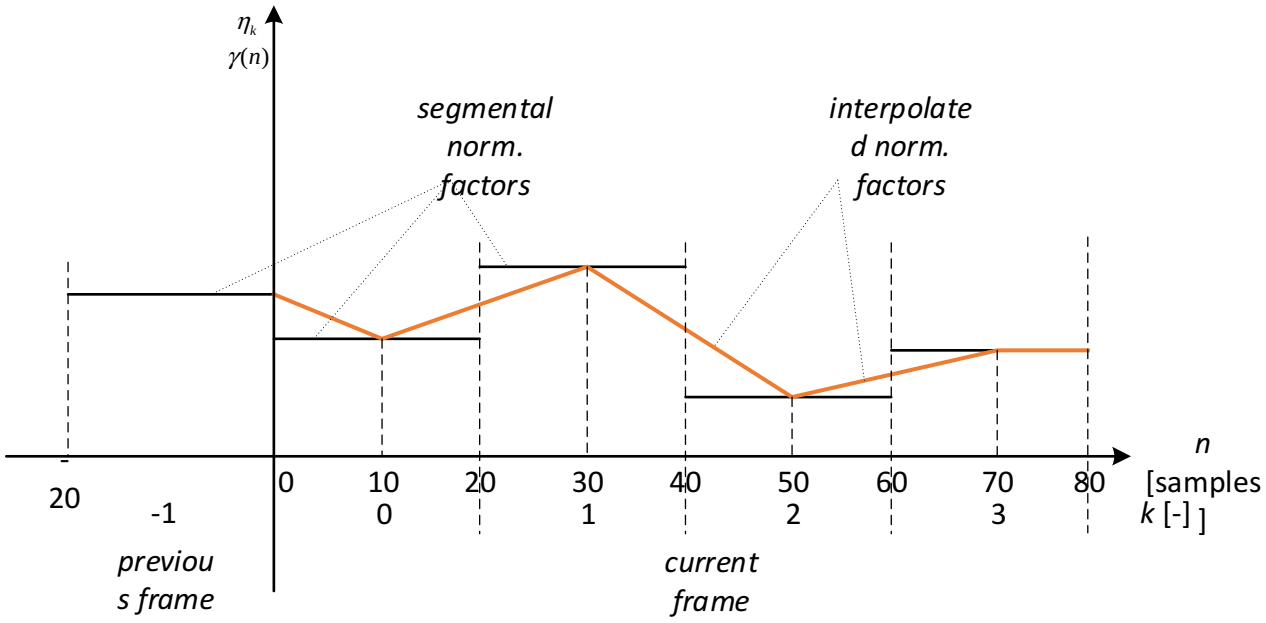


Figure 5.2-18: Interpolation of segmental normalization factors

Note that η_{-1} refers to the last segmental normalization factor in the previous frame. Therefore, η_{-1} is updated with η_3 after the interpolation process in each frame. The down-sampled temporal envelope is then normalized with the interpolated factors as follows

$$R_\gamma(n) = R_{4kHz}(n) \cdot \gamma(n), \quad n = 0, \dots, \frac{N}{4} - 1 \tag{5.2-138}$$

Finally, the global mean value of the normalized envelope is subtracted from $R_\gamma(n)$ which completes the normalization process. The global mean value is computed with

$$\bar{R}_\gamma = \frac{1}{\frac{N}{4}} \sum_{n=0}^{\frac{N}{4}-1} R_\gamma(n) \tag{5.2-139}$$

and the subtraction is done as follows

$$R_{norm}(n) = R_\gamma(n) - \bar{R}_\gamma, n = 0, \dots, \frac{N}{4} - 1 \tag{5.2-140}$$

It is useful to estimate the tilt of the temporal envelope. This is done by fitting a linear curve to the segmental means calculated in eq. (5.2-135) with the linear least squares (LLS) method. The tilt of the temporal envelope is equal to the slope of the linear curve. The linear curve calculated with the LLS method is defined as follows

$$y_k^{[LLS]} = a \cdot k + b, \quad k = 0, \dots, 3 \tag{5.2-141}$$

The objective is to minimize the sum of squared differences between $y_k^{[LLS]}$ and $y_k^{[LLS]}$ for all $k = 0, \dots, 3$. That is

$$(a_{LLS}, b_{LLS}) = \min_{a,b} \left[\sum_{k=0}^{3} \left(\bar{R}_k - y_k^{[LLS]} \right)^2 \right] \tag{5.2-142}$$

The optimal slope a_{LLS} is calculated by solving the minimization problem. That is

$$a_{LLS} = \frac{4 \sum_{k=0}^{3} k \cdot \bar{R}_k - 6 \sum_{k=0}^{3} \bar{R}_k}{4 \cdot 14 - 6 \cdot 6} = \frac{1}{5} \sum_{k=0}^{3} k \cdot \bar{R}_k - \frac{3}{10} \sum_{k=0}^{3} \bar{R}_k \tag{5.2-143}$$

The high-band autocorrelation function is calculated based on the normalized temporal envelope in the following way

$$X_{corr}(k - 40) = \frac{1}{max\left(E_f, E_f^{[-1]}\right)} \sum_{n=0}^{N/4-1} R_{norm}(n) R_{norm}(n - k), \quad k = 40, \dots, 80 - 1 \tag{5.2-144}$$

where E_f is the energy of the normalized temporal envelope in the current frame and $E_f^{[-1]}$ is the energy of the normalized temporal envelope in the previous frame. The energy is calculated with

$$E_f = \sum_{n=0}^{\frac{N}{4}-1}\left(R_{norm}(n)\right)^2 + 1.0 \quad (5.2\text{-}145)$$

In case of mode switching the factor in front of the summation term in eq. (5.2-144) is set to $1/E_f$ because the energy of the normalized temporal envelope in the previous frame is unknown.

The voicing of the high-band residual signal is closely related to the variance of the high-band autocorrelation function calculated with

$$\sigma_{corr} = \frac{1}{40}\sum_{k=0}^{40-1}\left(X_{corr}(k)\right)^2 \quad (5.2\text{-}146)$$

To improve the discriminative potential of the voicing parameter for the VOICED/UNVOICED decision the variance is multiplied with the maximum value of the high-band autocorrelation function. That is

$$v_{mult} = \sigma_{corr} \cdot \max_k\left(X_{corr}(k)\right) \quad (5.2\text{-}147)$$

The result is then transformed with the sigmoid function to limit its dynamic range. This is done as follows

$$v_{HB} = \frac{2.0}{1.0+exp(-\beta \cdot v_{mult})} - 1.0 \quad (5.2\text{-}148)$$

where the factor β is estimated experimentally and set to the constant value of 25.0. The high-band voicing parameter v_{HB} calculated above is then limited to lie within the range of $\langle 0.0;\ 1.0\rangle$.

5.2.2.3.2.7.4 Excitation mixing factor

The SWB TBE technology in the 3GPP EVS codec [3] uses the low-band excitation signal described in clause 5.2.2.3.2.7.2 as a predictor of the high-band residual signal described in clause 5.2.2.3.2.7.3. At lower bitrates of the IVAS codec below 24.4 kbps the SWB TBE encoder takes 19 bits to encode the spectral envelope and the energy of the high-band signal. With the frame length of 20 ms this results in the bitrate of 0.95 kbps. At bitrates higher than 24.4 kbps the SWB TBE encoder takes 32 bits to encode the spectral shape and the energy of the high-band signal. This results in the bitrate of 1.6 kbps.

The prediction of the high-band residual signal $\mathrm{r_{HB}(n)}$ defined in eq. (5.2-131) from the low-band excitation signal $\varepsilon_{16k}(\mathrm{n})$ defined in eq. (5.2-128) is done in the following way. The IVAS codec contains a pseudo-random generator with uniform distribution. Let us denote the pseudo-random signal in the current frame generated with the IVAS pseudo-random generator as

$$\mathrm{w}_{pseudo}(\mathrm{n}) \in \mathrm{U}[-32767; 32768],\ \ n = 0, \dots, N-1 \quad (5.2\text{-}149)$$

The pseudo-random signal has zero mean and non-zero variance of $\sigma_{pseudo} = 1.14e + 11$. Note, that the variance is a sample variance calculated over an exemple of 100 frames. The power of the low-band excitation signal is calculated as

$$E_{LB} = \sum_{n=0}^{N-1}\left(\varepsilon_{16k}(n)\right)^2 \quad (5.2\text{-}150)$$

The power of the random signal is normalized to the power of the low-band residual signal using the following scaling

$$w_{rand}(n) = \sqrt{\frac{E_{LB}}{\sigma_{pseudo}}} \cdot w_{pseudo}(n),\ \ n = 0, \dots, N-1 \quad (5.2\text{-}151)$$

Mixing the low-band excitation signal $\varepsilon_{16k}(n)$ with the normalized pseudo-random signal $w_{white}(n)$ yields the total excitation signal. That is

$$u_0(n) = g_\varepsilon \cdot \varepsilon_{16k}(n) + g_{white} \cdot w_{rand}(n),\ \ n = 0, \dots, N-1 \quad (5.2\text{-}152)$$

The gains of the excitation signals, g_ε and g_w, may be calculated by minimizing the mean square error (MSE) between the mixed output excitation signal and the high-band residual signal. Minimizing the MSE error directly on the excitation signals sampled at 16 kHz would require substantial amount of computational resources. To reduce the complexity the calculation of optimal gains is done on the temporal envelopes of the excitation signals in the downsampled domain. This is illustrated in the schematic diagram in Figure 5.2-19.

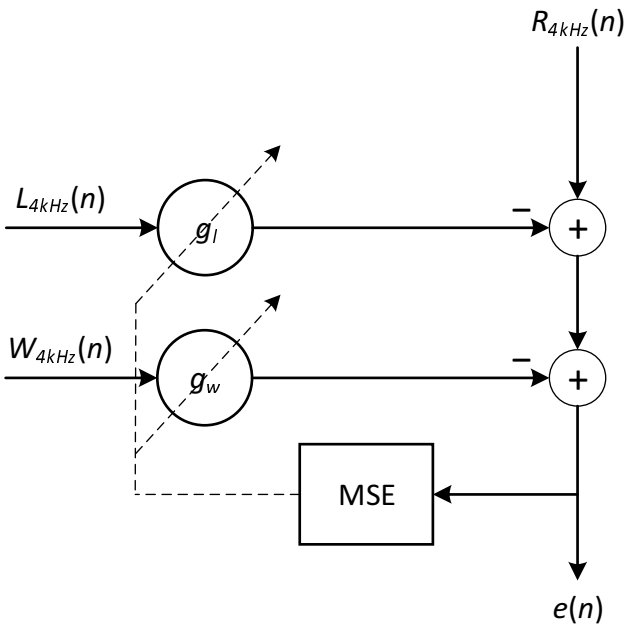


Figure 5.2-19: Mixing temporal envelopes of the low-band and random excitation signals

The temporal envelope of the high-band residual signal, downsampled to 4 kHz, is denoted as $R_{4kHz}(n)$ and is calculated in eq. (5.2-134) in clause 5.2.2.3.2.7.3. The temporal envelope of the pseudo-random signal $w_{white}(n)$, downsampled to 4 kHz, is calculated using the same procedure. Let the temporal envelope of the pseudo-random signal, downsampled to 4 kHz, be denoted as

$$W_{4kHz}(n), \quad n = 0, \ldots, \frac{N}{4} - 1 \tag{5.2-153}$$

Similarly, the temporal envelope of the low-band excitation signal, downsampled to 4 kHz is calculated using the same procedure as described in clause 5.2.2.3.2.7.3 and denoted as

$$L_{4kHz}(n), \quad n = 0, \ldots, \frac{N}{4} - 1 \tag{5.2-154}$$

The minimization of the MSE error in Figure 5.2-19 can be mathematically expressed as

$$(g_l^*, g_w^*) = \min_{g_l g_w} \left[\sum_{n=0}^{\frac{N}{4}-1} \left(R_{4kHz} - g_l \cdot L_{4kHz}(n) - g_w \cdot W_{4kHz}(n) \right)^2 \right] \tag{5.2-155}$$

where (g_l^*, g_w^*) are the optimal gains. The solution is given by

$$\begin{aligned} g_l^* &= \frac{c_3 \cdot c_4 - 2.0 \cdot c_1 \cdot c_2}{4.0 \cdot c_0 \cdot c_2 - c_4 \cdot c_4} \\ g_w^* &= \frac{c_1 \cdot c_4 - 2.0 \cdot c_0 \cdot c_3}{4.0 \cdot c_0 \cdot c_2 - c_4 \cdot c_4} \end{aligned} \tag{5.2-156}$$

where the values $c_0, \ldots, c_4$ are calculated with

$$\begin{aligned} c_0 &= \sum_{n=0}^{\frac{N}{4}-1} \left(L_{4kHz}(n) \right)^2 & c_3 &= -2.0 \sum_{n=0}^{\frac{N}{4}-1} R_{4kHz}(n) \cdot W_{4kHz}(n) \\ c_1 &= -2.0 \sum_{n=0}^{\frac{N}{4}-1} L_{4kHz}(n) \cdot R_{4kHz}(n) & c_4 &= 2.0 \sum_{n=0}^{\frac{N}{4}-1} L_{4kHz}(n) \cdot W_{4kHz}(n) \\ c_2 &= \sum_{n=0}^{\frac{N}{4}-1} \left(W_{4kHz}(n) \right)^2 & c_5 &= \sum_{n=0}^{\frac{N}{4}-1} \left(R_{4kHz}(n) \right)^2 \end{aligned} \tag{5.2-157}$$

The minimum MSE error energy (excess error) can be expressed as

$$E_{err} = (g_l^*)^2 c_0 + g_l^* c_1 + (g_w^*)^2 c_2 + g_w^* c_3 + g_l^* g_w^* c_4 + c_5 \tag{5.2-158}$$

For further processing the optimal gains are scaled in such a way that the gain associated with the temporal envelope of the low-band excitation signal is unitary. This is done as follows

$$g_{ln} = 1.0$$
$$g_{wn} = \frac{g_w^*}{g_l^* + g_w^*} \tag{5.2-159}$$

The re-scaling of gains saves the amount of information that needs to be transmitted from the encoder to the decoder. Only one gain parameter, g_{wn}, is quantized and transmitted in the bitstream instead of two gain parameters. Therefore, the re-scaling of gains reduces bit consumption and simplifies the quantization process. The re-scaling of gains affects the energy of the mixed temporal envelopes. In general, the energy of the mixed temporal envelopes does not match the energy of the temporal envelope of the high-band residual signal. This is not a problem since the SWB TBE encoder calculates the global gain and subframe gains containing the information about the total energy of the high-band signal. The calculation of subframe gains and the global gain is described in clause 5.2.6.1.14 and clause 5.2.6.1.15 of [3], respectively.

The normalized gain g_{wn} is limited by the maximum threshold of 1.0 and the minimum threshold of 0.0. The normalized gain is quantized using a 3-bit uniform scalar quantizer as follows

$$idx_g = \lfloor 7g_{wn} + 0.5 \rfloor \tag{5.2-160}$$

and the resulting index is limited to the interval $\langle 0; 7 \rangle$. The index idx_g is transmitted in the SWB TBE bitstream together with the other indices of the SWB TBE encoder at 0.95 kbps or 1.6 kbps. The decoded gain is calculated with

$$f_{mix} = \frac{idx_g}{7} \tag{5.2-161}$$

For further processing the decoded gain f_{mix} will be referred to as the high-band excitation mixing factor.

Using the mixing factor f_{mix} the low-band excitation signal $\varepsilon_{16k}(n)$, sampled at 16 kHz, and the normalized pseudo-random noise signal $w_{white}(n)$, sampled at 16 kHz, are mixed together. That is

$$u_1(n) = f_{mix} \cdot \varepsilon_{16k}(n) + (1 - f_{mix}) \cdot w_{rand}(n), \quad n = 0, \ldots, N-1 \tag{5.2-162}$$

The energy of the pseudo-random noise signal $w_{rand}(n)$ is normalized to the energy of the low-band excitation signal in eq. (5.2-151). However, the energy of the low-band excitation signal $\varepsilon_{16k}(n)$ varies in each frame and the fluctuations may eventually generate audible artifacts at frame borders. To smoothen the transitions between consecutive frames the energy of the low-band excitation signal is linearly interpolated between the previous frame and the current frame. This is done by calculating an interpolation factor

$$\zeta_w(n) = \sqrt{\frac{E_{LB}^{[-1]}}{E_{LB}}} + \frac{2n}{N}\left(1.0 - \sqrt{\frac{E_{LB}^{[-1]}}{E_{LB}}}\right), \quad n = 0, \ldots, \frac{N}{2} - 1 \tag{5.2-163}$$

where $E_{LB}^{[-1]}$ is the energy of the low-band excitation signal in the previous frame. The interpolation factor is then applied as follows

$$\begin{aligned} u_2(n) &= \zeta_w(n) \cdot f_{mix} \cdot \varepsilon_{16k}(n) + (1 - f_{mix}) \cdot w_{rand}(n), \quad n = 0, \ldots, \frac{N}{2} - 1 \\ u_2(n) &= f_{mix} \cdot (n) \cdot \varepsilon_{16k}(n) + (1 - f_{mix}) \cdot w_{rand}(n), \quad n = \frac{N}{2}, \ldots, N-1 \end{aligned} \tag{5.2-164}$$

For further smoothing of the transitions between consecutive frames the mixing factor f_{mix} is also linearly interpolated between the previous frame and the current frame. This is done as follows

$$\beta_{mix}(n) = f_{mix}^{[-1]} + \frac{2n}{N}\left(f_{mix} - \beta_{mix}(-1)\right), \quad n = 0, \ldots, \frac{N}{2} - 1 \tag{5.2-165}$$

where $f_{mix}^{[-1]}$ is the value of the mixing factor in the previous frame. By replacing f_{mix} with the linearly-interpolated mixing factor $\beta_{mix}(n)$ in the first half of the current frame in eq. (5.2-164) we get the final expression for mixing the low-band excitation signal $\varepsilon_{16k}(n)$ and the pseudo-random noise signal $\mathrm{w_{rand}}(n)$. That is

$$\begin{aligned} u(n) &= \zeta_w(n) \cdot \beta_{mix}(n) \cdot \varepsilon_{16k}(n) + \left(1 - \beta_{mix}(n)\right) \cdot w_{rand}(n), \quad n = 0, \ldots, \frac{N}{2} - 1 \\ u(n) &= f_{mix} \cdot (n) \cdot \varepsilon_{16k}(n) + (1 - f_{mix}) \cdot w_{rand}(n), \quad n = \frac{N}{2}, \ldots, N-1 \end{aligned} \tag{5.2-166}$$

5.2.2.3.2.7.5 High-band synthesis

The LP coefficients calculated on the high-band target signal by means of LP analysis in eq. (5.2-130) are converted in the SWB TBE encoder into LSF parameters and quantized. This is described in clause 5.2.6.1.3 in [3]. At 0.95 kbps the SWB TBE encoder uses 8 bits to quantize the LSF indices. At 1.6 kbps the SWB TBE encoder uses 21 bits to quantize the LSF indices.

At 1.75 kbps the SWB TBE encoder uses 20 bits to quantize the LSF indices when operating in the LRTD coding mode and when the super high band SHB SWB TBE spectral envelope is quantized using three quantization stages.

Since the spectral content of the high-band target for SWB TBE is reversed (or flipped) (see clause 5.2.6.1.1 in [3]), the order of the SHB LSF vector is initially reversed to also be in an increasing order of magnitude. The mean LSF value of each LSF is then removed from the corresponding component of the LSF vector. This is performed for all components of the LSF vector to give a mean removed LSF vector. The order of the SHB LSF vector is 10. The 10th-order target vector $\tilde{L}$ is obtained with:

$$\tilde{L}_i = (0.5f - L_i) - L_i^{mean}, i = 0,..,shb\_order - 1$$

where $\tilde{L}_i$ is a component of the vector $\tilde{L}$. In the first quantization stage a first sub vector $\tilde{L}^1$ is determined as having the first 6 LSF coefficients of the target LSF vector $\tilde{L}$

$$\tilde{L}^1 = \tilde{L}_i \quad , i = 0, \ldots, 5$$

The first sub vector is quantized with 4-bit weighted VQ using a codebook from `cb_LSF_BWE`. to give a quantized first sub vector $\hat{L}^1$. A residual sub vector is then determined by subtracting the quantized first sub vector from the full-length target vector, and because $\hat{L}^1$ is of order 6, the quantized first sub vector is extended with a number of zero value components to give a zero extended quantized first sub vector. In this case, the number of zero values is two, so that the extended quantized first sub vector is the same order as the 8th order vectors used for the calculations in the second quantization stage.

A residual LSF sub vector $\tilde{L}^2$ of order 8 is then formed by subtracting the first 8 components of the target LSF vector $\tilde{L}_i$ from the corresponding 8 components of the zero extended quantized first sub vector $\hat{L}^1$

$$\tilde{L}^2 = \tilde{L}_i - \hat{L}^1 \ , i = 0, \ldots, 7$$

The residual LSF sub vector $\tilde{L}^2$ is then quantized with Lattice VQ using 15 bits to the give the quantized residual LSF sub vector $\hat{L}^2$. The lattice VQ structure is defined similarly to the LVQ structure used for the LPC coefficients in LSF representation from clause 5.2.2 in [3], but for only one 8-dimensional vector. Consequently, for each considered number of bits of the LVQ, there are the corresponding number of leaders in each of the 3 lattice truncations and the corresponding scales.

The quantized first sub vector $\hat{L}^1$ from the first quantization stage is combined with the quantized residual LSF sub vector $\hat{L}^2$ from the second stage of quantization to give the quantized combined LSF sub vector $\hat{L}^{12}$

$$\hat{L}^{12} = \hat{L}^1 + \hat{L}^2$$

Finally, the last two LSF vector coefficients $\tilde{L}_{8,9}$ are predicted based on the LSF vector coefficients of the quantized combined LSF sub vector $\hat{L}^{12}$ using a chosen set of stored pre-determined optimal predictor coefficients $\tilde{\rho}$, where $\tilde{\rho}$ denotes a vector of 8 predictor coefficients. In all there are two different predictor coefficient sets of which one is selected by determining the set which provides a minimum mean square error between the quantized LSF vector $\hat{L}^3$ and the original target LSF vector. The predictor coefficient sets are stored in the table `LastCoefPred_1bit` and each predictor coefficient set consists of vector of 8 prediction coefficients $\widetilde{\rho_8}$ for predicting the 9th LSF coefficient and a vector of 8 prediction coefficients $\widetilde{\rho_9}$ for predicting the10th LSF coefficient. A single bit is used to signal the selected predictor coefficient set. The 10th order quantized LSF vector $\hat{L}^3$ is formed by concatenating the quantized combined LSF sub vector $\hat{L}^{12}$ with the predicted LSF coefficients $\hat{L}^3{}_{8,9}$, where each predicted LSF coefficient is predicted according to

$$\hat{L}_8^3 = \widetilde{\rho_8} * \hat{L}^{12} \text{ and } \hat{L}_9^3 = \widetilde{\rho_9} * \hat{L}^{12}$$

and where $*$ denotes the vector scalar product operator.

The final full length (10th order) quantized SHB LSF vector $\hat{L}$ can be calculated by adding the quantized LSF vector $\hat{L}^3$ with the mean vector

$$\hat{L} = \hat{L}^3 + L^{mean}$$

The resulting LSF vector $\hat{L}$ is then sorted to increasing order of magnitude and reversed back to the original flipped format, for use in SHB synthesis.

The quantized LSF indices are then decoded and converted back to LP coefficients as described in clause 5.2.6.1.4 in [3]. Let the decoded LP coefficients of the high-band target signal be denoted as

$$a_{qj}^{HB}(k), \quad k = 0, \ldots, P, \quad j = 0, \ldots, 3 \tag{5.2-167}$$

where $P = 10$ is the order of the LP filter and j is teh subframe index. Note, that the first decoded LP coefficient in each subframe is unitary, i.e. $a_{qj}^{HB}(0) = 1.0$ for $j = 0, \ldots, 3$. The decoded LP coefficients are then used to filter the mixed excitation signal $u(n)$ calculated in eq. (5.2-166). That is

$$y_{HB}(n + j \cdot 80) = u(n + j \cdot 80) - \sum_{k=1}^{P} a_{qj}^{HB}(k) y_{HB}(n + j \cdot 80 - k), \quad j = 0, \ldots, 3, \; n = 0, \ldots, 80 - 1 \tag{5.2-168}$$

5.2.2.3.2.7.6 Temporal gain/shape parameters

The spectral shape of the high-band target signal is encoded with the quantized LSF coefficients as described in clause 5.2.2.3.2.7.5. The SWB TBE encoder estimates temporal subframe gains in four consecutive subframes using the procedure described in clause 5.2.6.1.14 in [3].

Let the temporal gains estimated by the SWB TBE encoder be denoted as

$$g_{HB}(j), \quad j = 0, \ldots, 3 \tag{5.2-169}$$

Note, that the estimated temporal subframe gains are normalized to unit energy.

The temporal tilt of subframe gains is calculated by means of linear least squares (LLS) interpolation. This is done by fitting a linear curve to the subframe gains in four consecutive subframes and calculating its slope. The LLS interpolation process is illustrated in Figure 5.2-20.

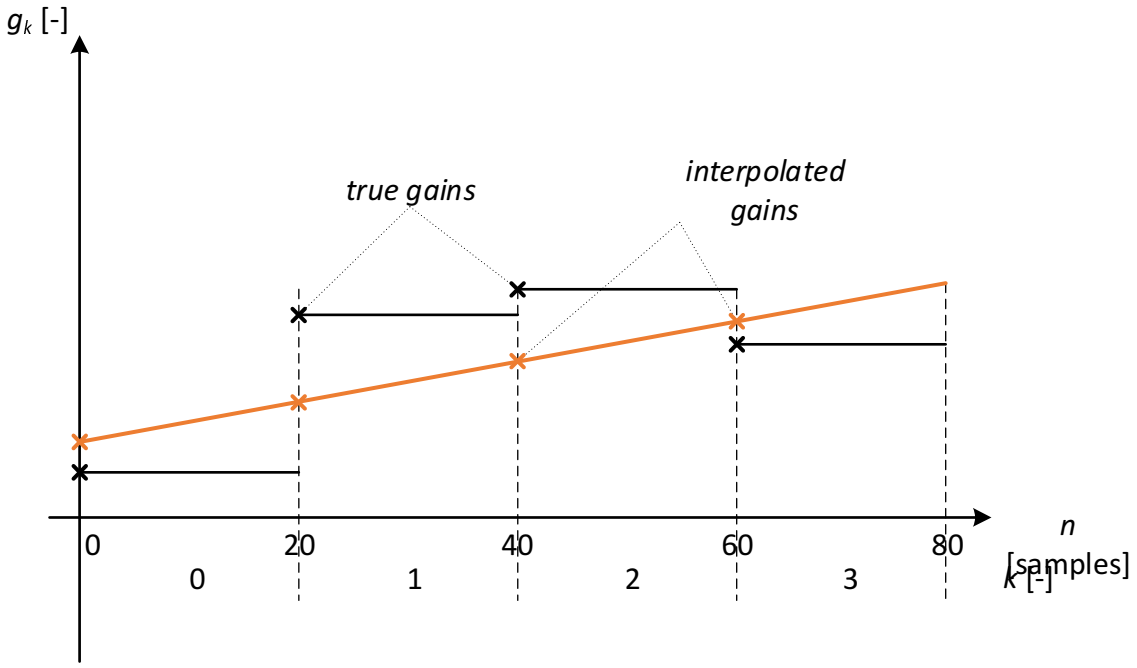


Figure 5.2-20: Interpolation of temporal subframe gains

Let the linear curve calculated with the LLS interpolation method be defined as

$$g_k^{[LLS]} = c \cdot k + d, \quad k = 0, \ldots, 3 \tag{5.2-170}$$

The objective of the LLS interpolation method is to minimize the sum of squared differences between $g_k^{[LLS]}$ and g_k for all $k = 0, \ldots, 3$. That is

$$(c_{LLS}, d_{LLS}) = \min_{c,d} \left[\sum_{k=0}^{3} \left(g_k - g_k^{[LLS]} \right)^2 \right] \quad (5.2\text{-}171)$$

The optimal slope c_{LLS} determines the tilt of the temporal gains. The optimal slope is calculated with

$$g_{tilt} = c_{LLS} = \frac{4 \sum_{k=0}^{3} k \cdot g_k - 6 \sum_{k=0}^{3} g_k}{4 \cdot 14 - 6 \cdot 6} = \frac{1}{5} \sum_{k=0}^{3} k \cdot g_k - \frac{3}{10} \sum_{k=0}^{3} g_k \quad (5.2\text{-}172)$$

The temporal subframe gains g_k are mixed with the interpolated subframe gains $g_k^{[LLS]}$ when the following condition is true

$$v_{HB} < 0.4 \text{ AND } idx_g \geq 5 \text{ AND } |g_{tilt}| < 0.2 \quad (5.2\text{-}173)$$

where v_{HB} is the voicing of the high-band residual signal calculated in eq. (5.2-148) and idx_g is the index of the normalized gain, calculated in eq. (5.2-160). The mixing of temporal subframe gains done as follows

$$\bar{g}_k = (1 - \kappa) \cdot g_k + \kappa \cdot g_k^{[LLS]}, \quad k = 0, \ldots, 3 \quad (5.2\text{-}174)$$

where the weight κ is proportional to the voicing of the high-band residual signal, calculated in eq. (5.2-148). The weight κ is calculated as

$$\kappa = \frac{-0.95}{0.4} (v_{HB} - 0.4) \quad (5.2\text{-}175)$$

and limited to the maximum value of 1.0 and the minimum value of 0.0.

The smoothed temporal subframe gains are quantized using the gain-shape quantizer, described in clause 5.2.6.1.14.2 in [3], with 5 bits. Let the quantized temporal subframe gains be denoted as

$$\hat{g}_k, \quad k = 0, \ldots, 3 \quad (5.2\text{-}176)$$

After the quantization process the temporal subframe gains are interpolated using the same LLS method as described in eqs. (5.2-170) to (5.2-172). Let the interpolated quantized gains in four sonsecutive subframes be denoted as

$$\hat{g}_k^{[LLS]}, \quad k = 0, \ldots, 3 \quad (5.2\text{-}177)$$

and the tilt of the quantized subframe gains as $\hat{g}_{tilt}$. The quantized subframe gains are smoothed when the following condition is true

$$idx_g \geq 6 \text{ AND } \left| \hat{g}_{tilt}^{[LLS]} \right| < 0.12 \quad (5.2\text{-}178)$$

The smoothing of the quantized subframe gains is done by means of averaging with the interpolated gains $\hat{g}_k^{[LLS]}$. That is

$$\tilde{g}_k = 0.7 \cdot \hat{g}_k + 0.3 \cdot \hat{g}_k^{[LLS]}, \quad k = 0, \ldots, 3 \quad (5.2\text{-}179)$$

The SWB TBE encoder quantizes and encodes the global energy of the synthesized high-band signal in the form of the frame gain. The calculation of the frame gain is described in clause 5.2.6.1.15 of [3]. The frame gain g_f is calculated by means of energy matching between the LP-filtered high-band signal $y_{HB}(n)$, defined in eq. (5.2-168) and the high-band target signal, described in clause 5.2.6.1.1 of [3]. The LP-filtered high-band signal is first scaled with the quantized subframe gains. This is done as follows

$$\tilde{y}_{HB}(n + k \cdot 80) = \tilde{g}_k \cdot y_{HB}(n + k \cdot 80), \quad k = 0, \ldots, 3, \quad n = 0, \ldots, 80 - 1 \quad (5.2\text{-}180)$$

The frame gain is then calculated using the energy-matching procedure described in clause 5.2.6.1.15 of [3]. The calculated frame gain may be modified under some specific conditions. For example, when the voicing in the high-band signal v_{HB} is high but the output excitation signal is dominated by the pseudo-random component the frame gain is attenuated with

$$g_f \leftarrow f_{att} \cdot g_f, \quad \text{if } v_{HB} > 0.1 \text{ AND } E_{err} > 5.0 \quad (5.2\text{-}181)$$

where E_{err} is the MSE excess error energy calculated in eq. (5.2-158) and f_{att} is the attenuation factor calculated as

$$f_{att} = 1.0 - 0.04(E_{err} - 5.0) \quad (5.2\text{-}182)$$

Further modifications to the frame gain parameter under some specific conditions are described in clause 5.2.6.15 of [3]. After the modifications the frame gain parameter is quantized in the log domain using 5-bit scalar quantizer. Note, that the lowest quantization point is -1.0 and the quantization step is set to 0.15.

Finally, the synthesized high-band signal is calculated with

$$\tilde{s}_{HB}(n) = g_f \cdot \tilde{y}_{HB}(n),\ n = 0, \ldots, N-1 \quad (5.2\text{-}183)$$

5.2.2.3.2.8 Multimode FD bandwidth extension coding

[TBD]

5.2.2.3.3 MDCT based coding

5.2.2.3.3.1 General

The IVAS MDCT based coding follows the MDCT Coding Mode in EVS, as described in clause 5.3 of [3]. Differences compared to EVS are described in the subsequent clauses.

5.2.2.3.3.2 Variable bitrate MDCT coding

The HQ Core has been adapted for IVAS operation to handle varying core-coder bitrate >= 16.4 kbps and < 48 kbps. The coding mode has a variable rate sharing internally between the envelope encoding and the fine structure encoding, where the fine structure encoding is done using the remaining bit budget after the envelope has been encoded. This functionality directly supports a variable bit rate for the HQ Core by allowing the fine structure bit rate to fluctuate. A special handling is implemented for the sub-mode HVQ described in clause 5.3.4.2.5 of [3] where the maximum number of peaks is calculated based on the core bit rate R_{core} according to.

$$number\_peaks = \left\lfloor \frac{6R_{core} - 17200}{7600} \right\rfloor \quad (5.2\text{-}184)$$

This coincides with the number of peaks for 24.4 kbps and 32 kbps in EVS. The variable bit rate between the discrete number of pulses is absorbed by the residual coding stage of the HVQ mode.

5.2.2.3.3.3 TCX entropy coding

The TCX entropy coding in IVAS follows the TCX entropy coding in EVS as described in clause 5.3.3.2.8 of [3]. For lower complexity, the low-level arithmetic coding routines are however replaced by an optimized arithmetic encoding routine (range coder).

The following pseudo-code describes the optimized arithmetic encoding routine, which is used for coding any symbol associated with a probability model. For the IVAS operation modes, the routine `rc_uni_enc_encode_symbol_fast()` replaces the routine `ar_encode()`as described in in clause 5.3.3.2.8 of [3]. The probability model is represented by a cumulative frequency table *cum_freq[]* and a symbol frequency table *sym_freq[]*. The symbol frequency table *sym_freq[]* can be derived from the cumulative frequency table *cum_freq[]* by simply determining the delta of two neighbored cumulative frequencies: *sym_freq[i] = cum_freq[i+1] - cum_freq[i]*. The variable *tot_shift* indicates the total frequency as a power of 2.

```
void rc_uni_enc_init( )
{
    uint32_t rc_low = 0;
    uint32_t rc_range = 0xFFFFFFFF;
    int16_t rc_cache = -1;
    int16_t rc_carry = 0;
    int16_t rc_carry_count = 0;

    int16_t byte_count = 0;
    int16_t last_byte_bit_count = -1;
}

void rc_uni_enc_encode_fast( uint16_t cum_freq, uint16_t sym_freq, uint16_t tot_shift )
{
    uint32_t r, tmp;
```

```
    r = rc_range >> tot_shift;
    tmp = r * cum_freq;

    rc_low += tmp;
    if ( rc_low < tmp )
    {
        rc_carry = 1;
    }

    rc_range = r * sym_freq;

    /* rc_range was shifted right by up to 16, so at most two renormalizations are needed */
    if ( rc_range < 0x01000000 )
    {
        rc_range <<= 8;
        rc_uni_enc_shift( );
        if ( rc_range < 0x01000000 )
        {
            rc_range <<= 8;
            rc_uni_enc_shift( );
        }
    }

    return;
}

rc_uni_enc_encode_symbol_fast( uint16_t symbol, uint16_t cum_freq[], uint16_t sym_freq[],
uint16_t tot_shift )
{
    rc_uni_enc_encode_fast( cum_freq[symbol], sym_freq[symbol], tot_shift );
}

rc_uni_enc_finish( )
{
    int16_t total_bit_count;
    uint32_t val, mask;
    int16_t bits;

    bits = norm_l( rc_range >> 24 ) - 22;
    bits++;

    mask = 0xFFFFFFFFu >> bits;
    val = ( rc_low + mask ) & ~mask;
    if ( val < rc_low )
    {
        rc_carry = 1;
    }

    rc_low = val;
    while ( bits > 0 )
    {
        rc_uni_enc_shift( );
        bits -= 8;
    }
    bits += 8;

    if ( rc_carry_count > 0 )
    {
        if ( rc_cache >= 0 )
        {
            byte_buffer[byte_count++] = (uint8_t) ( rc_cache + rc_carry );
        }

        while ( rc_carry_count > 1 )
        {
            byte_buffer[byte_count++] = (uint8_t) ( rc_carry + 0xFF );
            rc_carry_count--;
        }

        byte_buffer[byte_count++] = (uint8_t) ( ( rc_carry + 0xFF ) & ( 0xFFu << ( 8 - bits ) ) );
        last_byte_bit_count = bits;
    }
    else
    {
        byte_buffer[byte_count++] = (uint8_t) ( ( rc_cache + rc_carry ) & ( 0xFFu << ( 8 - bits ) )
);
        last_byte_bit_count = bits;
```

```
    }
    total_bit_count = ( ( byte_count - 1 ) << 3 ) + last_byte_bit_count;

    return total_bit_count;
}

int16_t rc_uni_enc_virtual_finish( )
{
    return ( ( byte_count + rc_carry_count ) << 3 ) +
           norm_l( rc_range >> 24 ) - 13 - 8 * ( (uint16_t) rc_cache >> 15 );
}

void rc_uni_enc_shift( )
{
    if ( ( rc_low < 0xFF000000u ) || rc_carry )
    {
        if ( rc_cache >= 0 )
        {
            byte_buffer[byte_count++] = (uint8_t) ( rc_cache + rc_carry );
        }

        while ( rc_carry_count > 0 )
        {
            byte_buffer[byte_count++] = (uint8_t) ( rc_carry + 0xFF );
            rc_carry_count--;
        }

        rc_cache = (int16_t) ( rc_low >> 24 );
        rc_carry = 0;
    }
    else
    {
        rc_carry_count++;
    }
    rc_low <<= 8;

    return;
}

rc_uni_enc_encode_bits( uint16_t value, int16_t bits )
{
    uint32_t tmp;

    rc_range >>= bits;
    tmp = rc_range * value;

    rc_low += tmp;
    if ( rc_low < tmp )
    {
        rc_carry = 1;
    }

    if ( rc_range < 0x01000000 )
    {
        rc_range <<= 8;
        rc_uni_enc_shift( );
        if ( rc_range < 0x01000000 )
        {
            rc_range <<= 8;
            rc_uni_enc_shift( );
        }
    }

    return;
}
```

5.2.2.3.3.4 Intelligent Gap Filling (IGF)

5.2.2.3.3.4.1 General

The general functionality of IGF is described in clause 5.3.3.2.11 of [3]. In the following all changes and additions that were made for IVAS are described.

For IVAS, activation of IGF and selection of individual configuration is not dependent on fixed bitrates, but instead bitrate intervals are used depending on the selected bandwidth for processing, the IVAS format and the type and number of IVAS elements (SCEs/CPEs). For modes where multiple elements are coded, the IGF configuration is called separately for each element. Accordingly, the bitrates given to the IGF configuration do not correspond to the overall bitrate but are bitrates specified for each element.

The maximum configuration bitrates for which IGF is active for SCEs and CPEs are shown in Table 5.2-13 and Table 5.2-14.

Table 5.2-13: IGF application modes – IVAS SCE

Mode	Bitrate
WB	≤ 9.6 kbps
SWB	≤ 64 kbps
FB	≤ 128 kbps

Table 5.2-14: IGF application modes – IVAS CPE

Mode	Bitrate
WB	≤ 13.2 kbps
SWB	≤ 96 kbps
FB	≤ 128 kbps

Note, that some low bitrate modes do not allow FB operation. Accordingly, IGF is only applied up to SWB for such cases. See Table 4.2-2 for details on the supported audio bandwidth per input format.

5.2.2.3.3.4.2 IGF Damping

5.2.2.3.3.4.2.1 Introduction

Typically, signal spectra become more noise-like toward higher frequencies. Therefore, the IGF source regions, which are located in a lower part of the spectrum than the higher-frequency target regions they are used to fill, are often too tonal compared to the original high-frequency spectrum. This is remedied by whitening of the tiles (as described in clause 5.3.3.2.11.6 of [3]). In much rarer cases, however, it can also happen that a very noisy source spectrum is copied to a much more tonal target spectrum. In the most extreme cases, pure background noise might be copied to a place where the original signal has a tonal component, like an overtone. This background noise will be scaled to the level of the tonal component by the transmitted scale-factor (which adjusts the spectral envelope of the source tile to the level of the target tile) inside the according scale-factor band (SFB). This will result in a noise band in the high-frequency spectrum which can be very audible and detrimental to the overall quality. An illustration of this effect is shown in Figure 5.2-21.

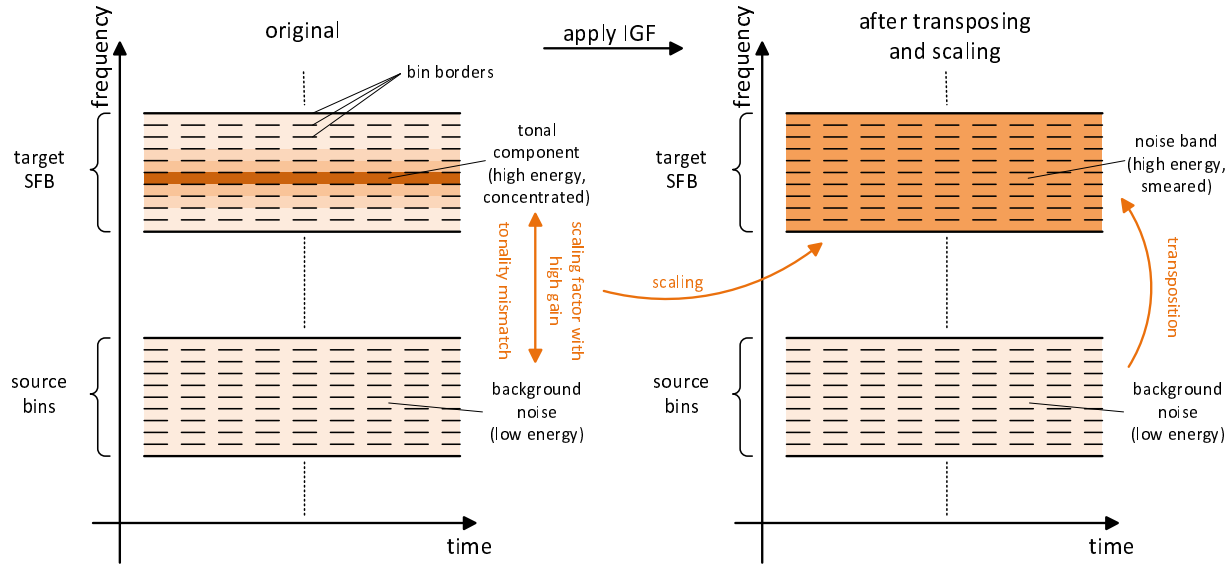


Figure 5.2-21: Illustration of noise bands due to tonality mismatch

To improve quality in such cases, a scale-factor damping is applied at the IGF encoder. The idea is to detect cases where such a mismatch occurs by analyzing tonality in both source and target SFBs and use the results to calculate a compensation value depending on the severity of the mismatch. With this compensation value and the original scale factor, a reduced scale-factor is calculated for the target SFB which will be transmitted in the bitstream and which will lead to a significantly reduced noise band when applied during the decoder IGF processing. There are no additional processing steps at the decoder, the technique is therefore encoder-only. A block diagram of the IGF encoder with the necessary adaptations is shown in Figure 5.2-22 and the individual processing steps are described in more detail in the following.

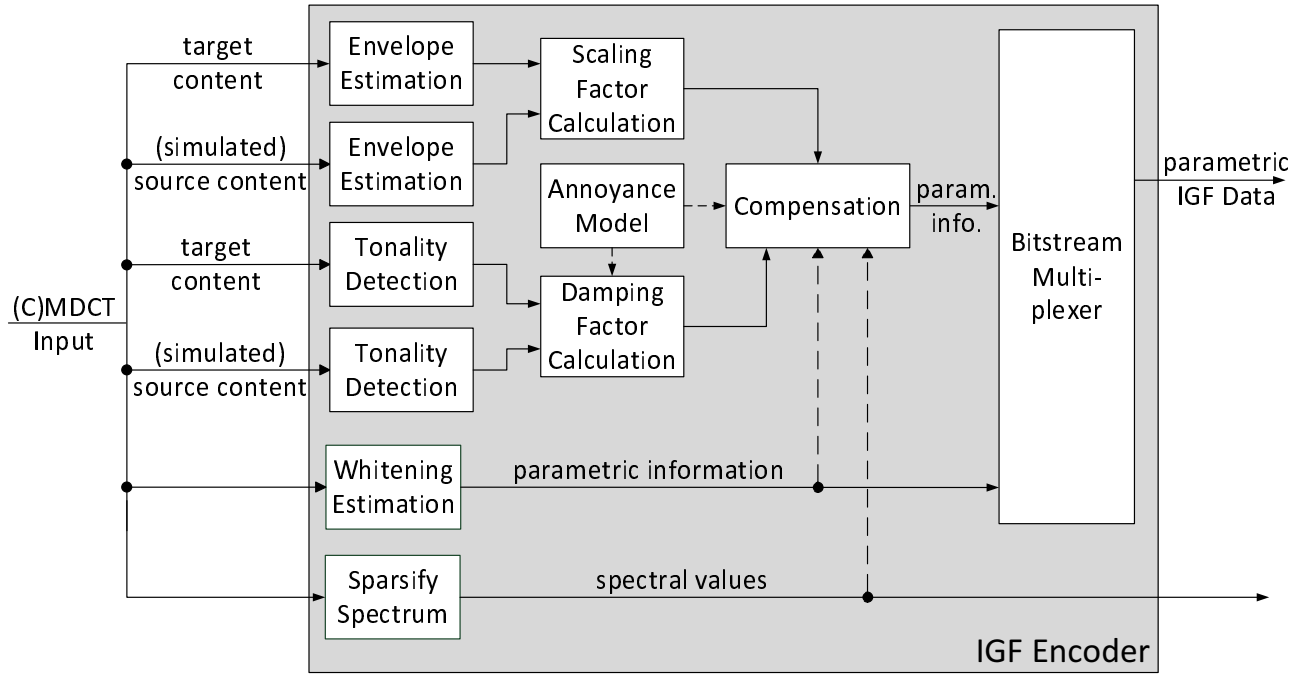


Figure 5.2-22: IGF encoder with added damping

5.2.2.3.3.4.2.2 Tonality mismatch detection

In a first step, those SFBs, where a tonality mismatch might cause noise band artefacts, have to be identified. In order to do so, the tonality in each SFB of the IGF range and the corresponding source bands has to be determined. The tonality is calculated in the same manner as it is already done for IGF Whitening using both the spectral flatness measure (SFM) and the crest factor. Spectral flatness and crest factor calculations are described in clauses 5.3.3.2.11.1.3 and

5.3.3.2.11.1.4 of [3], respectively. Smoothed spectral flatness values SFM_{src} and SFM_{dest} for source and target SFBs are obtained by following the calculation of *s(k)* in step 1) of clause 5.3.3.2.11.6.3 of [3]. However, in this case the values are calculated per SFB *b* and not per tile *k*.

Now the difference in tonality between source and destination is calculated:

$$SFM_{diff} = SFM_{src} - SFM_{dest} \tag{5.2-185}$$

Positive values of this difference indicate that the copied-up source SFB is noisier than the target SFB. Such an SFB becomes a likely candidate for damping.

In order to avoid damping in some unwanted cases, e.g. band-limitation inside an SFB, possibly affected SFBs the spectral tilt of the energy in all target bands with positive SFM differences is calculated as:

$$slope = \left(\sum_i x_i P_i - \frac{1}{e-b}\sum_i x_i \sum_i P_i\right)\left(\sum_i {x_i}^2 - \frac{1}{e-b}\left(\sum_i x_i\right)^2\right)^{-1}, \quad i = b, \dots, e-1 \tag{5.2-186}$$

with x as the bin number, P the TCX power spectrum, the start line and e the stop line of the current SFB.

However, a tonal component close to a border of an SFB might also cause a steep tilt, but should still be subjected to damping. To separate these two cases, another shifted SFM calculation is performed for bands with steep tilt.

The threshold for the slope value is defined as

$$thresh_{tilt} = \frac{60}{e-b} \tag{5.2-187}$$

with division by the SFB width *e-b* as normalization.

If there is a strong decline slope $< -thresh_{tilt}$, the frequency region for which SFM is calculated will be shifted down by half the width of the SFB; for a strong incline slope $> thresh_{tilt}$ it is shifted up. In this way, tonal components that are supposed to be damped can still be correctly detected due to low SFM while for higher SFM values damping will not be applied. The threshold here is defined as the value 0.04, where damping is only applied if the shifted SFM falls below the threshold.

5.2.2.3.3.4.2.3 Perceptual annoyance model

To ensure application within reasonable bounds, damping should only be used if the target SFB is indeed very tonal. So only whenever both

$$SFM_{diff} > 0 \tag{5.2-188}$$

and

$$SFM_{dest} < 0.1 \tag{5.2-189}$$

hold, damping should be applied.

Additionally, the amount of damping is dependent on the tonal-to-noise ratio in the SFB. For SFBs with a higher ratio, i.e. lower background noise, more damping is applied, and vice versa.

For the calculation of this tonal-to-noise ratio the squared TCX power spectral values P of all bins i in an SFB are summed up and divided by the width of the SFB (given by start line b and stop line e) to get the average energy of the band. This average is subsequently used to normalize all the energies in the band.

$$P_{norm,k} = \sqrt{P_k}\left(\frac{1}{e-b} * \sum_{sb=b}^{e-1}\sqrt{P}_{sb}\right)^{-1}, \qquad k = b, \dots, e-1 \tag{5.2-190}$$

All bins with a normalized energy $P_{norm,k}$ below 1 are then summed up and counted as the noise part P_{noise} while everything above a threshold of 1 + adap with

$$adap = \frac{e-b}{40} \tag{5.2-191}$$

is counted as the tonal part P_{tonal}. From the tonal and the noise part the final tonal-to-noise log-ratio is computed:

$$tonalToNoise = 20 * \log_{10}\left(\frac{P_{tonal}}{P_{noise}}\right) \tag{5.2-192}$$

5.2.2.3.3.4.2.4 Calculation of damping factor and compensation of tonality mismatch

Now the damping factor d_{curr} is calculated as

$$d_{curr} = \left(\frac{SFM_{dest}}{SFM_{src}}\right)^{\alpha} + \beta, \quad (5.2\text{-}193)$$

where α and β are damping adjustment parameters that are calculated as

$$\alpha = min\left(\frac{320}{e-1}, 1.25\right) \quad (5.2\text{-}194)$$

with e is the stop line of the current SFB and

$$\beta = \begin{cases} 0.1 * \left((10 + adap) - tonalToNoise\right), & \text{if } \left((10 + adap) - tonalToNoise\right) > 0 \\ 0, & else \end{cases} \quad (5.2\text{-}195)$$

where $adap$ is again calculated as

$$adap = \frac{e-b}{40} \quad (5.2\text{-}196)$$

The parameter α decreases with frequency in order to apply less damping for higher frequencies while β is used to further reduce the strength of the damping if the tonal-to-noise ratio of the SFB falls below a threshold. The more significantly it falls below this threshold, the more the damping is reduced.

As damping is only activated within certain constraints, it is necessary to apply smoothing in order to prevent abrupt on/off transitions. To realize this, several smoothing mechanisms are active.

Directly after a transient, a core switch to TCX or an undamped previous frame damping is only gradually applied to avoid extreme energy drops after high-energy transients. Furthermore, a forgetting factor in the form of an IIR filter is utilized to also take the results of previous frames into account.

All smoothing techniques are comprised in the following formula:

$$d = min\left(\frac{d_{curr} + d_{prev}}{2} + 0.1 * smooth, 1\right), \quad (5.2\text{-}197)$$

where d_{prev} is the damping factor of the previous frame. If damping was not active in the previous frame d_{prev} is overwritten with d_{curr} but limited to a minimum of 0.1. The variable smooth is an additional smoothing factor that will be set to 2 during transient frames (flag isTransient active) or after core switches (flag isCelpToTCX active), to 1 if in the previous frame damping was inactive. In each frame with damping the variable will be decreased by 1, but may not fall below 0. If a power spectrum is not available or if the frame is transient, no damping will be applied.

In the final step, the damping factor *d* is multiplied with the scaling gain:

$$g_{damped} = g * d \quad (5.2\text{-}198)$$

The resulting compensated gain factor is then transmitted in the bitstream as part of the core-encoded IGF side data.

5.2.2.3.3.4.3 Stabilization of IGF Whitening levels

5.2.2.3.3.4.3.1 General

In some cases, the calculated IGF whitening levels *currWLevel(k)* for each tile *k* (as described in clause 5.3.3.2.11.6.3 of [3]) can fluctuate quite strongly by changing their value every few frames. This leads to an unstable listening impression detrimental to the overall quality. Therefore, additional stability mechanisms are introduced for IVAS as described below.

5.2.2.3.3.4.3.2 Margin-based Whitening thresholds and minimum deviation from past SFM mean

In cases where $currWLevel(k) \neq prevWLevel(k)$, switching to the new whitening level $currWLevel(k)$ is allowed if the spectral flatness *SFM* of tile *k* (see calculation of $s(k)$ in clause 5.3.3.2.11.6.3 of [3]) lies outside a margin of ± 0.15 around the whitening thresholds ThM_k and ThS_k (see mapping in clause 5.3.3.2.11.1.5 of [3]).

If it falls inside either margin, as in

$$ThM_k - 0.15 < SFM_k < ThM_k + 0.15 \quad (5.2\text{-}199)$$

or

$$ThM_s - 0.15 < SFM_k < ThM_s + 0.15 \quad (5.2\text{-}200)$$

switching is prohibited if additionally the deviation of SFM_k from the mean $SFM_{mean,k}$ of the *SFM* values of the n_{prev} previous frames is sufficiently small:

$$currWLevel(k) = \begin{cases} prevWLevel(k), & SFM_{mean,k} < 0.2 \\ currWLevel(k), & else \end{cases} \quad (5.2\text{-}201)$$

$SFM_{mean,k}$ is calculated for each tile *k*:

$$SFM_{mean,k} = \frac{\sum_{i=0}^{nprev-1} SFM_{prev,k}(i)}{n_{prev}} \quad (5.2\text{-}202)$$

where n_{prev} is the number of available past values $SFM_{prev,k}$ in the previous 5 frames. If past SFM values are not available for one the past 5 frames - due to the frame being ACELP or transient – these frames will be excluded and n_{prev} reduced accordingly.

5.2.2.3.3.4.3.3 Oscillating pitch handling

If the pitch of a signal is not stable but oscillates rapidly (e.g. a voice or instrument employing vibrato) overtones of the signal might constantly cross tile borders. This can strongly affect the SFM values in the two tiles between which the overtone oscillates and, in the worst case, can lead to constantly changing whitening levels. If such a case is detected, whitening is turned off for both tiles.

The detection is done for pairs of tiles *k* and *k+1*. First, the change in *SFM* for both tiles is calculated over the 3 previous frames (provided and SFM value is available for all 3 frames) by summing up the SFM differences:

$$SFM_{diff,k} = \sum_{i=0}^{2}\big(SFM_k(i) - SFM_k(i+1)\big) \quad (5.2\text{-}203)$$

and

$$SFM_{diff,k+1} = \sum_{i=0}^{2}\big(SFM_{k+1}(i) - SFM_{k+1}(i+1)\big) \quad (5.2\text{-}204)$$

If either

$$SFM_{diff,k} < 0 \;\&\&\; SFM_{diff,k+1} > 0 \quad (5.2\text{-}205)$$

or

$$SFM_{diff,k} > 0 \;\&\&\; SFM_{diff,k+1} < 0 \quad (5.2\text{-}206)$$

holds – indicating that the tonality changes in opposite directions between the tiles – and the difference between $SFM_{diff,k}$ and $SFM_{diff,k+1}$ is sufficiently large as in

$$\left|SFM_{diff,k} - SFM_{diff,k+1}\right| > 0.5 \quad (5.2\text{-}207)$$

and if $currWLevel(k) = 0$ and $currWLevel(k+1) \neq 0$, or vice versa, is true – meaning whitening is off only in one of the tiles – the whitening will be turned off in both tiles:

$$currWLevel(k) = currWLevel(k+1) = 0 \quad (5.2\text{-}208)$$

5.2.2.3.3.4.3.4 Whitening hangover

Additionally, to the stabilization methods described above, a hangover of 2 frames is employed. This means that any conditions for a switch of the whitening level have to be fulfilled for 3 frames in a row before the switch is allowed in the third frame.

5.2.2.3.3.4.4 Whitening processing for CPE bitrate range of]32.0-48.0] kbps

In cases where the IVAS element is a CPE and the configuration bitrate falls in the]32.0-48.0] kbps range, a different processing is done to determine the whitening levels.

Instead of calculating a separate whitening level for each tile, this is done only for the first tile. The whitening levels of the other tiles are set to the same value. Additionally, only whitening levels 0 and 1 – indicating no whitening and mid-whitening, respectively – are allowed.

The calculation of this one whitening level based on the first tile is also done differently from other modes. Instead of setting the level by comparing the SFM value of the target tile to fixed thresholds (see clause 5.3.3.2.11.1.5 of [3] and clause 5.2.2.3.3.4.5 in the present document), the level is determined adaptively by comparing the SFM value of the target tile against the SFM value in the source tile that is used for copy-up. This comparison further depends on the decision of the speech-music classifier, described in clause 5.2.2.2.11.

If the signal is classified as **music**, the whitening level will be set to 0 – indicating no whitening – for

$$SFM_{tar} \leq SFM_{src} + 0.5 \quad (5.2\text{-}209)$$

and to 1 – indicating mid-whitening – for

$$SFM_{tar} > SFM_{src} + 0.5 \quad (5.2\text{-}210)$$

If the signal is classified as **speech**, the whitening level will be set to 0 – indicating no whitening – for

$$SFM_{tar} \leq SFM_{src} + 0.1 \quad (5.2\text{-}211)$$

and to 1 – indicating mid-whitening – for

$$SFM_{tar} > SFM_{src} + 0.1 \quad (5.2\text{-}212)$$

To avoid redundancies, only this one level is transmitted in the bitstream.

5.2.2.3.3.4.5 IVAS IGF whitening thresholds and mapping

The EVS IGF whitening mapping and thresholds are described in clause 5.3.3.2.11.1.5 of [3]. For IVAS, different tables for the whitening thresholds are used depending on whether an SCE or a CPE element is processed, see tables 5.2-15 and 5.2-16.

For the whitening mapping described in step 3) of clause 5.3.3.2.11.6.3 of [3] and shown per bitrate in Table 99, a new table depending on the IVAS bitrate intervals for both SCE and CPE elements is shown in table 5.2-17.

Note, that the bitrate given to the IGF configuration depends on the IVAS format and is not always the same as the overall bitrate.

Table 5.2-15: Thresholds for whitening – IVAS SCE

Bitrate	Mode	nT	ThM	ThS
≤ 9.6 kbps	WB	2	0.36, 0.36	1.41, 1.41
≤ 9.6 kbps	SWB	4	0.89, 0.89, 0.80 ,0.80	1.25, 1.25, 1.19, 1.19
]9.6-13.2] kbps	SWB	6	1.05, 1.05, 1.10, 1.10, 1.05, 1.05	1.70, 1.70, 1.65, 1.65, 1.60, 1.50
]13.2-16.4] kbps	SWB	7	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20
]16.4-24.4] kbps	SWB	8	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20
]24.4-32.0] kbps	SWB	3	0.91, 0.85, 0.85	1.34, 1.35, 1.35
]32.0-48.0] kbps	SWB	1	1.15	1.19
]48.0-64.0] kbps	SWB	1	1.15	1.19
≤ 16.4 kbps	FB	9	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20, 0.65, 0.65
]16.4-24.4] kbps	FB	10	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20, 0.65, 0.65
]24.4-32.0] kbps	FB	4	0.78, 0.31, 0.34, 0.34	1.49, 1.38, 0.65, 0.65
]32.0-48.0] kbps	FB	1	0.80	1.0
]48.0-64.0] kbps	FB	1	0.80	1.0
]64.0-96.0] kbps	FB	1	0	2.82
]96.0-128.0] kbps	FB	1	0	2.82

Table 5.2-16: Thresholds for whitening – IVAS CPE

Bitrate	Mode	nT	ThM	ThS
≤ 9.6 kbps	WB	2	0.80, 0.75	1.50, 1.45
]9.6-13.2] kbps	WB	2	0.90, 0.85	1.60, 1.50
≤ 9.6 kbps	SWB	4	0.89, 0.89, 0.80 ,0.80	1.25, 1.25, 1.19, 1.19
]9.6-13.2] kbps	SWB	6	1.05, 1.05, 1.10, 1.10, 1.05, 1.05	1.70, 1.70, 1.65, 1.65, 1.60, 1.50
]13.2-16.4] kbps	SWB	7	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20
]16.4-24.4] kbps	SWB	8	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20
]24.4-32.0] kbps	SWB	3	0.91, 0.85, 0.85	1.34, 1.35, 1.35
]32.0-48.0] kbps	SWB	6	adaptive	adaptive
]48.0-64.0] kbps	SWB	4	1.00, 1.00, 1.20, 1.25	1.50, 1.50, 1.60, 1.60
]64.0-80.0] kbps	SWB	2	1.20, 1.25	1.60, 1.60
]80.0-96.0] kbps	SWB	1	1.15	1.19
≤ 16.4 kbps	FB	9	1.20, 1.20, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.70, 1.70, 1.70, 1.70, 1.55, 1.45, 1.20, 0.65, 0.65
]16.4-24.4] kbps	FB	10	1.20, 1.20, 1.15, 1.15, 1.10, 1.05, 1.00, 0.90, 0.34, 0.34	1.80, 1.80, 1.80, 1.80, 1.65, 1.55, 1.45, 1.20, 0.65, 0.65
]24.4-32.0] kbps	FB	4	0.78, 0.31, 0.34, 0.34	1.49, 1.38, 0.65, 0.65
]32.0-48.0] kbps	FB	7	adaptive	adaptive
]48.0-64.0] kbps	FB	5	1.00, 1.00, 1.20, 1.25, 0.75	1.50, 1.50, 1.60, 1.60, 0.75
]64.0-80.0] kbps	FB	3	1.20, 1.25, 0.75	1.60, 1.60, 1.00
]80.0-96.0] kbps	FB	2	0.91, 0.85	1.34, 1.35
]96.0-128.0] kbps	FB	1	0	2.82

Note, that for CPEs within bitrate interval]32.0-48] kbps, the whitening is not determined via fixed thresholds but calculated adaptively as described in clause 5.2.2.3.3.4.4.

Table 5.2-17: Whitening mapping – IVAS

Bitrate	mode	mapping
≤ 13.2 kbps	WB	apply
≤ 9.6 kbps	SWB	apply
]9.6-13.2] kbps	SWB	NOP
]13.2-32.0] kbps	SWB	apply
]32.0-96.0] kbps	SWB	NOP
≤ 32.0 kbps	FB	apply
]32.0-128.0] kbps	FB	NOP

5.2.2.3.3.4.6 IVAS IGF scale factor tables

The EVS IGF scale factor tables are described in 5.3.3.2.11.1.7, table 94 of [3]. For IVAS, different tables are used depending on whether an SCE or a CPE element is processed, see tables 5.2-18 and 5.2-19. For the configuration of CPE elements inside the]32.0-48.0] kbps bitrate interval the number of bands is reduced by half for TCX10 frames.

Note, that the bitrate given to the IGF configuration depends on the IVAS format and is not always the same as the overall bitrate.

Table 5.2-18: Scale factor band offset table – IVAS SCE

Bitrate	Mode	Number of bands (nB)	Scale factor band offsets (t[0],t[1],…,t[nB])
≤ 9.6 kbps	WB	3	164, 186, 242, 320
≤ 9.6 kbps	SWB	4	200, 264, 344, 440, 566
]9.6-13.2] kbps	SWB	6	228, 264, 308, 360, 420, 488, 566
]13.2-16.4] kbps	SWB	7	256, 288, 328, 376, 432, 496, 576, 640
]16.4-24.4] kbps	SWB	8	256, 284, 320, 360, 404, 452, 508, 576, 640
]24.4-32.0] kbps	SWB	8	256, 284, 318, 358, 402, 450, 508, 576, 640
]32.0-48.0] kbps	SWB	3	512, 534, 576, 640
]48.0-64.0] kbps	SWB	3	512, 534, 576, 640
≤ 16.4 kbps	FB	9	256, 288, 328, 376, 432, 496, 576, 640, 720, 800
]16.4-24.4] kbps	FB	10	256, 284, 320, 360, 404, 452, 508, 576, 640, 720, 800
]24.4-32.0] kbps	FB	10	256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800
]32.0-48.0] kbps	FB	4	512, 584, 656, 728, 800
]48.0-64.0] kbps	FB	4	512, 584, 656, 728, 800
]64.0-96.0] kbps	FB	2	640, 720, 800
]96.0-128.0] kbps	FB	2	640, 720, 800

Table 5.2-19: Scale factor band offset table – IVAS CPE

Bitrate	Mode	Number of bands (nB)	Scale factor band offsets (t[0],t[1],…,t[nB])
≤ 9.6 kbps	WB	2	196, 248, 320
]9.6-13.2] kbps	WB	2	228, 268, 320
≤ 9.6 kbps	SWB	4	200, 264, 344, 440, 566
]9.6-13.2] kbps	SWB	6	228, 264, 308, 360, 420, 488, 566
]13.2-16.4] kbps	SWB	7	256, 288, 328, 376, 432, 496, 576, 640
]16.4-24.4] kbps	SWB	8	256, 284, 320, 360, 404, 452, 508, 576, 640
]24.4-32.0] kbps	SWB	8	256, 284, 318, 358, 402, 450, 508, 576, 640
]32.0-48.0] kbps	SWB TCX20	6	360, 392, 424, 464, 508, 560, 640
	SWB TCX10	3	360, 424, 508, 640
]48.0-64.0] kbps	SWB	7	400, 424, 448, 476, 508, 540, 576, 640
]64.0-80.0] kbps	SWB	4	464, 496, 532, 576, 640
]80.0-96.0] kbps	SWB	3	512, 536, 576, 640
≤ 16.4 kbps	FB	9	256, 288, 328, 376, 432, 496, 576, 640, 720, 800
]16.4-24.4] kbps	FB	10	256, 284, 320, 360, 404, 452, 508, 576, 640, 720, 800
]24.4-32.0] kbps	FB	10	256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800
]32.0-48.0] kbps	FB TCX20	8	360, 392, 424, 464, 508, 560, 640, 720, 800
	FB TCX10	4	360, 424, 508, 640, 800
]48.0-64.0] kbps	FB	9	400, 424, 448, 476, 508, 540, 576, 640, 720, 800
]64.0-80.0] kbps	FB	6	464, 496, 532, 576, 640, 720, 800
]80.0-96.0] kbps	FB	5	512, 536, 576, 640, 720, 800
]96.0-128.0] kbps	FB	2	640, 720, 800

5.2.2.3.3.4.7 IVAS IGF source band mapping

The EVS IGF source band mapping is described in clause 5.3.3.2.11.1.8 of [3]. For IVAS, different mappings are used depending on whether an SCE or a CPE element is processed, see new tables for minimal source bands (tables 5.2-20 and 5.2-21) and mapping functions (tables 5.2-22 and 5.2-23).

Note, that the bitrate given to the IGF configuration depends on the IVAS format and is not always the same as the overall bitrate.

Table 5.2-20: IGF minimal source subband – IVAS SCE

Bitrate	Mode	minSrc
≤ 9.6 kbps	WB	30
≤ 9.6 kbps	SWB	32
]9.6-13.2] kbps	SWB	32
]13.2-16.4] kbps	SWB	32
]16.4-24.4] kbps	SWB	32
]24.4-32.0] kbps	SWB	32
]32.0-48.0] kbps	SWB	64
]48.0-64.0] kbps	SWB	64
≤ 16.4 kbps	FB	32
]16.4-24.4] kbps	FB	32
]24.4-32.0] kbps	FB	32
]32.0-48.0] kbps	FB	64
]48.0-64.0] kbps	FB	64
]64.0-96.0] kbps	FB	64
]96.0-128.0] kbps	FB	64

Table 5.2-21: IGF minimal source subband – IVAS CPE

Bitrate	Mode	minSrc
≤ 9.6 kbps	WB	32
]9.6-13.2] kbps	WB	32
≤ 9.6 kbps	SWB	32
]9.6-13.2] kbps	SWB	32
]13.2-16.4] kbps	SWB	32
]16.4-24.4] kbps	SWB	32
]24.4-32.0] kbps	SWB	32
]32.0-48.0] kbps	SWB	48
]48.0-64.0] kbps	SWB	48
]64.0-80.0] kbps	SWB	64
]80.0-96.0] kbps	SWB	64
≤ 16.4 kbps	FB	32
]16.4-24.4] kbps	FB	32
]24.4-32.0] kbps	FB	32
]32.0-48.0] kbps	FB	48
]48.0-64.0] kbps	FB	48
]64.0-80.0] kbps	FB	64
]80.0-96.0] kbps	FB	64
]96.0-128.0] kbps	FB	64

Table 5.2-22: Mapping functions for IVAS SCE

Bitrate	Mode	nT	mapping Function
≤ 9.6 kbps	WB	2	*m2a*
≤ 9.6 kbps	SWB	4	*m4b*
]9.6-13.2] kbps	SWB	6	*m2a*
]13.2-16.4] kbps	SWB	7	*m3b*
]16.4-24.4] kbps	SWB	8	*m3c*
]24.4-32.0] kbps	SWB	3	*m3c*
]32.0-48.0] kbps	SWB	1	*m1*
]48.0-64.0] kbps	SWB	1	*m1*
≤ 16.4 kbps	FB	9	*m4c*
]16.4-24.4] kbps	FB	10	*m4d*
]24.4-32.0] kbps	FB	4	*m4*
]32.0-48.0] kbps	FB	1	*m1*
]48.0-64.0] kbps	FB	1	*m1*
]64.0-96.0] kbps	FB	1	*m1*
]96.0-128.0] kbps	FB	1	*m1*

Table 5.2-23: Mapping functions for IVAS CPE

Bitrate	Mode	nT	mapping Function
≤ 9.6 kbps	WB	2	*m1*
]9.6-13.2] kbps	WB	2	*m1b*
≤ 9.6 kbps	SWB	4	*m4b*
]9.6-13.2] kbps	SWB	6	*m2a*
]13.2-16.4] kbps	SWB	7	*m3b*
]16.4-24.4] kbps	SWB	8	*m3c*
]24.4-32.0] kbps	SWB	3	*m3c*
]32.0-48.0] kbps	SWB TCX20	6	*m2c*
	SWB TCX10	3	*m3g*
]48.0-64.0] kbps	SWB	4	*m2d*
]64.0-80.0] kbps	SWB	2	*m1c*
]80.0-96.0] kbps	SWB	1	*m1d*
≤ 16.4 kbps	FB	9	*m4c*
]16.4-24.4] kbps	FB	10	*m4d*
]24.4-32.0] kbps	FB	4	*m4*
]32.0-48.0] kbps	FB TCX20	7	*m3e*
	FB TCX10	4	*m4e*
]48.0-64.0] kbps	FB	5	*m3f*
]64.0-80.0] kbps	FB	3	*m2e*
]80.0-96.0] kbps	FB	2	*m2f*
]96.0-128.0] kbps	FB	1	*m1*

In the following the newly added mapping functions are given. For mappings reused from EVS, please refer to clause 5.3.3.2.11.1.8 of [3]. New tables for tile number and tile width are given in 5.2-24 and 5.2-25. For the configuration of CPE elements inside the]32.0-48.0] kbps bitrate interval different mappings are used for TCX20 and TCX10 frames.

Note, that for some IVAS modes (e.g. SWB for]16.4-24.4] kbps range) the number of tiles was increased with tiles consisting of only SFB. In these cases, some tiles are often contiguous and not overlapping, so they can be grouped together in one case of the according mapping function. Accordingly, there the number of cases per mapping function does not correspond to the number of tiles

The mapping function $m1b$ is defined with:

$$m1b \coloneqq minSB + tF(80, f) + \big(x - t(0)\big) \quad for \quad t(0) \le x < t(2) \tag{5.2-213}$$

The mapping function $m1c$ is defined with:

$$m1c \coloneqq minSB + tF(212, f) + \big(x - t(0)\big) \quad for \quad t(0) \le x < t(4) \tag{5.2-214}$$

The mapping function $m1d$ is defined with:

$$m1d \coloneqq minSB + tF(200, f) + \big(x - t(0)\big) \quad for \quad t(0) \le x < t(3) \tag{5.2-215}$$

The mapping function $m2c$ is defined with:

$$m2c \coloneqq \begin{cases} minSb + tF(120, f) + \big(x - t(0)\big) & for \quad t(0) \le x < t(4) \\ minSb + tF(140, f) + \big(x - t(4)\big) & for \quad t(4) \le x < t(nB) \end{cases} \tag{5.2-216}$$

The mapping function $m2d$ is defined with:

$$m2d \coloneqq \begin{cases} minSb + tF(80, f) + \big(x - t(0)\big) & for \quad t(0) \le x < t(4) \\ minSb + tF(144, f) + \big(x - t(4)\big) & for \quad t(4) \le x < t(nB) \end{cases} \tag{5.2-217}$$

The mapping function $m2c$ is defined with:

$$m2e \coloneqq \begin{cases} minSb + tF(212, f) + \big(x - t(0)\big) & for \quad t(0) \le x < t(4) \\ minSb + tF(200, f) + \big(x - t(4)\big) & for \quad t(4) \le x < t(nB) \end{cases} \tag{5.2-218}$$

The mapping function $m2f$ is defined with:

$$m2f := \begin{cases} minSb + tF(200, f) + \left(x - t(0)\right) & for \quad t(0) \le x < t(3) \\ minSb + tF(240, f) + \left(x - t(4)\right) & for \quad t(3) \le x < t(nB) \end{cases} \tag{5.2-219}$$

The mapping function $m3e$ is defined with:

$$m3e := \begin{cases} minSb + tF(120, f) + \left(x - t(0)\right) & for \quad t(0) \le x < t(4) \\ minSb + tF(140, f) + \left(x - t(4)\right) & for \quad t(4) \le x < t(6) \\ minSb + tF(140, f) + \left(x - t(6)\right) & for \quad t(6) \le x < t(nB) \end{cases} \tag{5.2-220}$$

The mapping function $m3f$ is defined with:

$$m3f := \begin{cases} minSb + tF(80, f) + \left(x - t(0)\right) & for \quad t(0) \le x < t(4) \\ minSb + tF(144, f) + \left(x - t(4)\right) & for \quad t(4) \le x < t(6) \\ minSb + tF(160, f) + \left(x - t(6)\right) & for \quad t(6) \le x < t(nB) \end{cases} \tag{5.2-221}$$

The mapping function $m3g$ is defined with:

$$m3g := \begin{cases} minS + \left(x - t(0)\right) & for \quad t(0) \le x < t(1) \\ minSb + tF(40, f) + \left(x - t(1)\right) & for \quad t(1) \le x < t(2) \\ minSb + tF(80, f) + \left(x - t(2)\right) & for \quad t(2) \le x < t(nB) \end{cases} \tag{5.2-222}$$

The mapping function $m4b$ is defined with:

$$m4b(x) := \begin{cases} minSb + \left(x - t(0)\right) & for \quad t(0) \le x < t(1) \\ minSb + tF(32, f) + \left(x - t(1)\right) & for \quad t(1) \le x < t(2) \\ minSb + tF(46, f) + \left(x - t(2)\right) & for \quad t(2) \le x < t(3) \\ minSb + tF(40, f) + \left(x - t(3)\right) & for \quad t(3) \le x < t(nB) \end{cases} \tag{5.2-223}$$

The mapping function $m4c$ is defined with:

$$m4c(x) := \begin{cases} minSb + \left(x - t(0)\right) & for \quad t(0) \le x < t(4) \\ minSb + tF(48, f) + \left(x - t(4)\right) & for \quad t(4) \le x < t(6) \\ minSb + tF(64, f) + \left(x - t(6)\right) & for \quad t(6) \le x < t(7) \\ minSb + \left(x - t(7)\right) & for \quad t(7) \le x < t(nB) \end{cases} \tag{5.2-224}$$

The mapping function $m4d$ is defined with:

$$m4d(x) := \begin{cases} minSb + \left(x - t(0)\right) & for \quad t(0) \le x < t(4) \\ minSb + tF(32, f) + \left(x - t(4)\right) & for \quad t(4) \le x < t(7) \\ minSb + tF(64, f) + \left(x - t(7)\right) & for \quad t(6) \le x < t(8) \\ minSb + \left(x - t(8)\right) & for \quad t(8) \le x < t(nB) \end{cases} \tag{5.2-225}$$

The mapping function $m4e$ is defined with:

$$m4e(x) := \begin{cases} minSb + \left(x - t(0)\right) & for \quad t(0) \le x < t(1) \\ minSb + tF(40, f) + \left(x - t(1)\right) & for \quad t(1) \le x < t(2) \\ minSb + tF(80, f) + \left(x - t(2)\right) & for \quad t(2) \le x < t(3) \\ minSb + tF(140, f) + \left(x - t(3)\right) & for \quad t(3) \le x < t(nB) \end{cases} \tag{5.2-226}$$

Table 5.2-24: Number of tiles *nT* and tile width *wT* – IVAS SCE

Bitrate	Mode	*nT*	*wT*
≤ 9.6 kbps	WB	2	*t(2)-t(0), t(nB)-t(2)*
≤ 9.6 kbps	SWB	4	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(nB)-t(3)*
]9.6-13.2] kbps	SWB	6	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(nB)-t(5)*
]13.2-16.4] kbps	SWB	7	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(nB)-t(6)*
]16.4-24.4] kbps	SWB	8	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(nB)-t(7)*
]24.4-32.0] kbps	SWB	3	*t(4)-t(0), t(7)-t(4), t(nB)-t(7)*
]32.0-48.0] kbps	SWB	1	*t(nB)-t(0)*
]48.0-64.0] kbps	SWB	1	*t(nB)-t(0)*
≤ 16.4 kbps	FB	9	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(nB)-t(9)*
]16.4-24.4] kbps	FB	10	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(9)-t(8), t(nB)-t(9)*
]24.4-32.0] kbps	FB	4	*t(4)-t(0), t(6)-t(4), t(9)-t(6), t(nB)-t(9)*
]32.0-48.0] kbps	FB	1	*t(nB)-t(0)*
]48.0-64.0] kbps	FB	1	*t(nB)-t(0)*
]64.0-96.0] kbps	FB	1	*t(nB)-t(0)*
]96.0-128.0] kbps	FB	1	*t(nB)-t(0)*

Table 5.2-25: Number of tiles *nT* and tile width *wT* – IVAS CPE

Bitrate	Mode	*nT*	*wT*
≤ 9.6 kbps	WB	2	*t(1)-t(0), t(nB)-t(1)*
]9.6-13.2] kbps	WB	2	*t(1)-t(0), t(nB)-t(1)*
≤ 9.6 kbps	SWB	4	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(nB)-t(3)*
]9.6-13.2] kbps	SWB	6	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(nB)-t(5)*
]13.2-16.4] kbps	SWB	7	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(nB)-t(6)*
]16.4-24.4] kbps	SWB	8	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(nB)-t(7)*
]24.4-32.0] kbps	SWB	3	*t(4)-t(0), t(7)-t(4), t(nB)-t(7)*
]32.0-48.0] kbps	SWB TCX20	6	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(nB)-t(5)*
	SWB TCX10	3	*t(1)-t(0), t(2)-t(1), t(nB)-t(2)*
]48.0-64.0] kbps	SWB	4	*t(2)-t(0), t(4)-t(2), t(6)-t(4), t(nB)-t(6)*
]64.0-80.0] kbps	SWB	2	*t(2)-t(0), t(nB)-t(2)*
]80.0-96.0] kbps	SWB	1	*t(nB)-t(0)*
≤ 16.4 kbps	FB	9	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(nB)-t(9)*
]16.4-24.4] kbps	FB	10	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(7)-t(6), t(8)-t(7), t(9)-t(8), t(nB)-t(9)*
]24.4-32.0] kbps	FB	4	*t(4)-t(0), t(6)-t(4), t(9)-t(6), t(nB)-t(9)*
]32.0-48.0] kbps	FB TCX20	7	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(4)-t(3), t(5)-t(4), t(6)-t(5), t(nB)-t(6)*
	FB TCX10	4	*t(1)-t(0), t(2)-t(1), t(3)-t(2), t(nB)-t(3)*
]48.0-64.0] kbps	FB	5	*t(2)-t(0), t(4)-t(2), t(6)-t(4), t(7)-t(6), t(nB)-t(7)*
]64.0-80.0] kbps	FB	3	*t(2)-t(0), t(4)-t(2), t(nB)-t(4)*
]80.0-96.0] kbps	FB	2	*t(3)-t(0), t(nB)-t(3)*
]96.0-128.0] kbps	FB	1	*t(nB)-t(0)*

5.2.2.3.3.4.8 IVAS IGF – TCX transitions

For EVS all allowed combinations of transitions and their according transition factors and window lengths are detailed in clause 5.3.3.2.11.2 of [3], Table 97. For IVAS, the same information based on IGF bitrate intervals is given in Table 5.2-26. This information holds for both IVAS SCE and CPE elements.

Note, that the bitrate given to the IGF configuration depends on the IVAS format and is not always the same as the overall bitrate.

Table 5.2-26: TCX transitions, transition factor *k*, window length *n*

Bitrate / Mode	*isTCX10*	*isTCX20*	*isCelpToTCX*	Transition factor *f*	Window length *n*
≤ 13.2 kbps	false	true	false	1.00	320

WB	false	true	true	1.25	400
≤ 32.0 kbps SWB	false	true	false	1.00	640
	false	true	true	1.25	800
]32.0-96.0] kbps SWB	false	true	false	1.00	640
	false	true	true	1.00	640
	true	false	false	0.50	320
]16.4-32.0] kbps FB	false	true	false	1.00	960
	false	true	true	1.25	1200
]48.0-128.0] kbps FB	false	true	false	1.00	960
	false	true	true	1.00	960
	true	false	false	0.50	480

5.2.2.3.4 Switching coding modes

5.2.2.3.4.1 Transitions between ACELP and HQ MDCT

The transitions between ACELP and HQ Core in IVAS operations has been harmonized to be more similar to the transitions between ACELP and TCX Core. When switching from HQ MDCT to ACELP, the transition is based on MC2 as described in clause 5.4.2.2 of [3]. Similar to MC2, a synthesis at the ACELP sampling rate sr_{celp} is generated by performing an additional MDCT synthesis and overlap-add operation corresponding to clause 5.3.3.2.12.1. in [3] using the portion of the spectrum corresponding to sr_{celp}. This synthesis is stored in the ACELP excitation memory. Upon a switch from HQ MDCT to ACELP, the content of the excitation memory is filtered through the pre-emphasis filter and the decoded filter coefficients of the current ACELP frame to form the excitation memory for the current frame. If the frame is not classified as voiced, indicated by that the preliminary coder type is not VOICED, the CELP coder in the current frame is forced to operate in Transition coding (TC) mode as described in clause 5.1.13 of [3]. The transition from ACELP to HQ Core in IVAS operation is done similar to the transition from ACELP to TCX as described in clause 5.4.3.1 of [3]. This means the HQ Core frames are extended in the same way as the TCX frames to cover the missing overlap-add from previous frame.

5.2.2.3.4.2 Switching from TCX/HQ to ACELP in DFT-based stereo

The time-domain bandwidth extension process SWB TBE is defined for ACELP mode in clause 5.2.6.2 of [3]. When IVAS is operating in ACELP coding mode, the SWB TBE target signal denoted $s_{HB}(n)$ 5.2.6.1.1 of [3], is updated with the high band down-mix signal which is generated by a STFT synthesis as described in 5.3.2.4.11. Meanwhile TCX/HQ coding mode operates on the full band down-mix target signal $s_M(n)$ also generated by a STFT synthesis, the high band down-mix target signal $s_{HB}(n)$ is not generated. To keep the SWB TBE target buffer updated in DFT stereo operations during TCX/HQ coding, a process is defined to extract the high-band target signal.

The SWB TBE operates at the bandwidth 6 – 14 kHz for 12.8 kHz core, and at 7.5 – 15.5 kHz for 16 kHz core. To adapt the target high band generation for these two bandwidths, an intermediate length of an intermediate frame, L_{inter}, is determined by rescaling the length of the input frame L in the input sampling frequency, f_{in}. The upper limit of the target band is set to be the Nyquist frequency in the intermediate frame. For the target band with frequency limits of $[f_{lo}, f_{hi}]$, the intermediate length L_{inter} is:

$$L_{inter} = L\frac{f_{inter}}{f_{in}} \tag{5.2-227}$$

The intermediate sampling frequency of the intermediate frame f_{inter} is twice the upper frequency limit of the target band, f_{hi}:

$$f_{inter} = 2f_{hi} \tag{5.2-228}$$

The full band input $s_M(n)$ with length L is then linearly interpolated using the function as defined in [3] 5.4.4.4 to first obtain an intermediate frame $x_{inter}(n), n = 0,1, \ldots, L_{inter} - 1$ with the length L_{inter} following the equations:

$$x_{inter}(n) = \begin{cases} s_M(0) + n_{frac}\big(s_M(1) - s_M(0)\big), & n_{frac} < 0 \\ s_M(L-1) + \big(n_{frac} - n_{floor}\big)\big(s_M(L-1) - s_M(L-2)\big), & n_{frac} > L-1 \\ s_M\big(n_{floor}\big) + \big(n_{frac} - n_{floor}\big)\Big(s_M\big(n_{floor}+1\big) - s_M\big(n_{floor}\big)\Big), & otherwise \end{cases} \tag{5.2-229}$$

The interpolated sampling value at point n is $x_{inter}(n)$, and n_{frac} is a fractional point where the source vector x is to be estimated. The first two cases of $x_{inter}(n)$ handles the samples on the edges of the frame where the new sampling point

is extrapolated from either the last or the first two samples of the frame. An index offset n_{offset} handles the case when $L_{int} > L$ and the first sample point in the stretched frame would be below $n = 0$ and the last sample point exceeding $n = L - 1$. The fractional point, n_{frac} is calculated using the displacement ϵ and index offset n_{offset}:

$$n_{floor} = \lfloor n_{frac} \rfloor \tag{5.2-230}$$

$$n_{frac} = n \frac{L}{L_{inter}} + n_{offset} \tag{5.2-231}$$

$$n_{offset} = 0.5 \frac{L}{L_{inter}} - 0.5 + \epsilon \tag{5.2-232}$$

A displacement denoted as ϵ for n_{offset} is defined according to:

$$\epsilon = \begin{cases} -0.13, & L/L_{inter} > 0.3 \\ 0, & otherwise \end{cases} \tag{5.2-233}$$

The spectrum of the resulting intermediate frame is reversed for $n = 0, 1, \ldots, L_{inter} - 1$ by changing the sign of every second sample:

$$x_{inter,rev}(n) = \begin{cases} -x_{inter}(n), & n\ is\ even \\ x_{inter}(n), & n\ is\ odd \end{cases} \tag{5.2-234}$$

A second linear interpolation is then performed on $x_{inter,rev}(n)$ to adjust the frame length to match the sampling frequency of the following low pass filtering and decimation operation. The cut-off frequency of the low-pass filter is selected to be in the middle of the spectrum. The decimator performs decimation by 2 which gives an output sampling frequency of half the input sampling frequency. When the frequency of decimation is f_{dec}, the decimation frame length L_{dec} is:

$$L_{dec} = L_{inter} \frac{f_{dec}}{f_{inter}} \tag{5.2-235}$$

The decimation process is explained in detail in clause 5.2.6.1.9 in [3]. In the frequency spectrum of the second interpolation output, $x_{dec}(n)$, the target band is aligned in the lower half of the spectrum. The low-pass filter and decimation extracts the target frequency band for the SWB TBE target signal.

5.2.2.3.4.3 Bandwidth switching

A change of the audio bandwidth (BW) may happen as a consequence of a bitrate change or a coded audio bandwidth change. In EVS, when a change from WB to SWB occurs, or from SWB to WB, an audio bandwidth switching post-processing at the decoder is performed in order to improve the perceptual quality. A smoothing is applied for switching from WB to SWB, and a blind audio BWE is employed for switching from SWB to WB. More information can be found in clause 6.3.7 of [3].

In IVAS, a different bandwidth switching (BWS) technique is employed. First, the new BWS algorithm is implemented at the encoder part of the IVAS codec in the core-coder (i.e. there is one BWS module per one transport channel). Then, the BWS algorithm in IVAS is implemented only for switching from a lower BW to a higher BW (for example from WB to SWB). In this direction, the switching is relatively fast (see clause 5.2.2.2.6) and an abrupt high-frequency (HF) content change can be annoying. The IVAS BWS algorithm is thus designed to smooth such switching. On the other hand, no special treatment is implemented for switching from a higher BW to a lower BW because in this direction there is practically no important HF content in the spectrum, so the change of the spectrum content is not unnaturally abrupt and annoying.

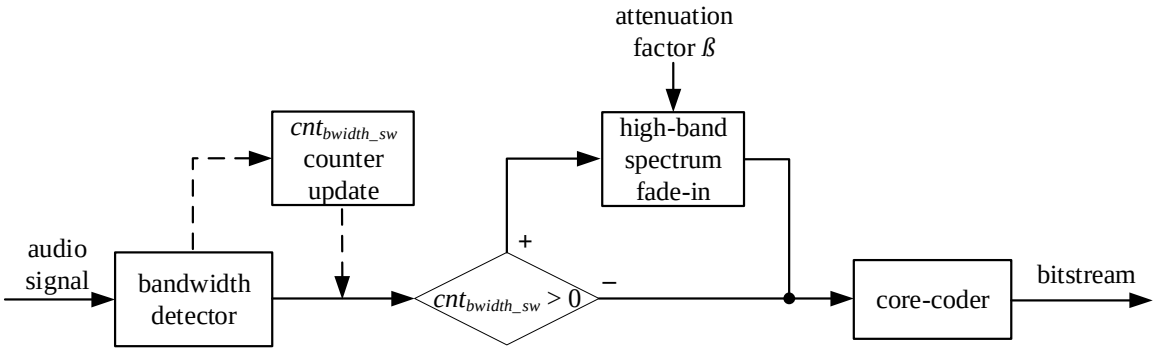


Figure 5.2-23: Schematic block diagram of the bandwidth switching algorithm

A schematic block diagram of the encoder-side BWS algorithm is shown in Figure 5.2-23 and it comprises the final audio bandwidth decision operation as part of the BWD from clause 5.2.2.2.6, a cnt_{bwidth_sw} counter updating operation, a comparison operation, and a high-band spectrum fade-in operation. The idea of the BWS algorithm is to smooth the perceptual impact of an audio BW switching already at the encoder part of the IVAS codec while removing the artifacts in the synthesis. The high-band (HB > 8 kHz) part of the spectrum is attenuated in several consecutive frames after a BWS instance as indicated by the final audio BW decision module from clause 5.2.2.2.6. More specifically, a gain of the HB spectrum is faded-in in an attenuation operation from Figure 5.2-23 and thus smartly controlled in case of a BWS in order to avoid unpleasant artifacts. The attenuation is applied before the HB spectrum is quantized and encoded in the core-coder, so the smoothed BW transitions are already present in the transmitted bitstream and no further treatment is needed at the decoder. For example, in case of audio bandwidth switching from WB to SWB, the HB spectrum corresponding to frequencies above 8 kHz is smoothed before further processing.

The BWS algorithm then works as follows.

Referring to Figure 5.2-23, the calculator first updates a counter of frames cnt_{bwidth_sw} where audio bandwidth switching occurs and attenuation is applied at the end of the pre-processing for each IVAS transport channel based on the final BWD decision as follows. The calculator initially set the value of the counter of frames cnt_{bwidth_sw} to an initialization value of "0". When there is detected – as a response to a final BWD decision from the audio bandwidth decision module (clause 5.2.2.2.6) – a BW change from a lower BW to a higher BW the value of the counter of frames is increased by 1. In the following frames, the counter is increased by 1 in every frame until it reaches its maximum value $B_{tran} = 5$ frames. When the counter reaches its maximum value B_{tran}, the counter cnt_{bwidth_sw} is then reset to 0 and a new detection of a BW switching can occur.

Next, when $cnt_{bwidth\_sw} > 0$, an attenuation is applied to the transport channel signal in frame n with an attenuation factor β_n defined as follows:

$$\beta_n = \frac{cnt_{bwidth\_sw}}{B_{tran}}, \quad n = 0, \dots, B_{tran} - 1. \tag{5.2-236}$$

Finally, the attenuation factor β is applied in the BWS transition period defined by B_{tran} frames depending on the coding mode as follows.

In TCX and HQ core-coder frames the high-band gain of the spectrum $X_M(k)$ of length L as defined in clause 5.3.2 of [3] is controlled and the HB part of the spectrum $X_M(k)$, right after the time-to-frequency domain transformation, is updated (faded-in) in frame n using the following relation:

$$\acute{X}_M(k + L_{WB}) = \beta_n \cdot X_M(k + L_{WB}), \quad k = 0, \dots, K - L_{WB} - 1, \tag{5.2-237}$$

where L_{WB} is the length of spectrum corresponding to the WB audio bandwidth, i.e. $L_{WB} = 320$ samples in the frame length of 20 ms (normal HQ, or TCX20 frame), $L_{WB} = 80$ samples in transient frames, $L_{WB} = 160$ samples in TCX10 frames, and k is the transform domain sample index in the range $[0, K - L_{WB} - 1]$ where K is the length of the whole spectrum in particular transform sub-mode (normal, transient, TCX20, TCX10).

In ACELP core with time-domain BWE (TBE) frames the attenuation is applied with the attenuation factor β_n to the SWB gain shapes parameters of the HB part of the spectrum before these parameters are additionally processed. The temporal gain shapes parameters $g_s(j)$ are defined in clause 5.2.6.1.14.2 of [3] and consist of four values. Thus, in IVAS, there applies in frame n the following attenuation:

$$g_s(j) = \beta_n \cdot g_s(j) \tag{5.2-238}$$

where $j = 0, \ldots, 3$ is the gain shape number.

In ACELP core with frequency-domain BWE (FD-BWE) frames, the high-band gain of the transformed original input signal $X_M(k)$ of length L as defined in clause 5.2.6.2.1 of [3] is controlled and the HB part of the MDCT spectrum is attenuated in frame n using the following relation:

$$X'_M(k + L_{WB}) = \beta_n \cdot X_M(k + L_{WB}), \quad k = 0, \ldots, K - L_{WB} - 1. \tag{5.2-239}$$

Note that NB coding is not considered in IVAS and SWB to FB switching is not treated as its subjective impact is negligible. However, the principles above are used to cover the other BWS scenarios.

The attenuated sound signal is finally encoded in the core-coder (SCE/CPE/MCT). If the counter cnt_{bwidth_sw} is not larger than 0, then the sound signal is encoded in the core-coder without attenuation.

5.2.2.3.5 DTX/CNG operation

5.2.2.3.5.1 General

Processing related to DTX/CNG operation in principle works the same as in EVS, see clause 5.6 of [3]. Coding mode-specific deviations are described in the respective DTX/CNG subclause of the respective format. Differences that apply to multiple or all IVAS coding modes are described in this clause.

5.2.2.3.5.2 VQ Gain scaling and quantization for FD-CNG

In[3], clause 5.6.3.5 equation (1399), the computation of the global gain value for the MSVQ used in FD-CNG is described. A scale value is added as given by table 154 of [3]. In IVAS modes, Table 5.3-21 from this document is used instead.

Table 5.2-27: Energy scaling values in Unified stereo CNG

Bandwidth	**WB**					**SWB & FB**				
Bitrate [kbps]	13.2	16.4	24.4	32	>= 48	13.2	16.4	24.4	32	>= 48
Scaling factor [dB]	-4	-3	-1.6	0.2	0	-2	-3	-0.8	-0.25	0

When calculating the quantized value of the global gain, instead of equation (1400) of [3], Equation (5.2-240) is used:

$$I_{g,FD-CNG} = \min(\max(\lfloor 1.5 g_{FD-CNG} + 45 \rfloor, 0), 127) \tag{5.2-240}$$

5.2.2.3.5.3 Efficient first stage for multistage VQ for FD-CNG

This clause describes the enhanced first stage operation of the FD-CNG MSVQ which is employed in Object-based audio (ISM), Stereo audio, Scene-based Audio (SBA) for quantization of spectral shape for each given FD-CNG base SID-frame type. The subsequent stages of the FD-CNG MSVQ are following the steps as described in clause 5.6.3.5 of [3] up to the given number of total stages.

The FD-CNG SID encoder obtains the MSVQ target vector N^{dB}_{FD-CNG} (corresponding to $N^{dB}_{L,FD-CNG}$, $N^{dB}_{R,FD-CNG}$, $N^{dB}_{M,FD-CNG}$, $N^{dB}_{S,FD-CNG}$ in clause 5.3.5.2.4 for MDCT-based stereo SID encoding, $N^{dB}_{SBA,FD-CNG}$ in clause 5.4.9.5 for SBA SID encoding, $N^{dB}_{ISM,FD-CNG}$ in clause 5.6.6.5 and to $\bar{N}^{dB}_{FD-CNG}$ in the clause of the generic MSVQ description EVS [3] for DFT-based stereo SID encoding and ISM SID encoding). N^{dB}_{FD-CNG} is first subtracted by precalculated mid vector $N^{dB}_{FD-CNG,midQ}$ and then upscaled by factor $FDCNG_{dctInvScale} (= 2.3792724609375)$ into vector $N^{dB}_{FD-CNG,mr\_scaled}$ to optimize the dynamic range for the first stage inner search loop. Subsequently the vector $N^{dB}_{FD-CNG,DCT-II}$, is created in the DCT-II ($M = 24$) domain by applying a normalized DCT- type II transformation to $N^{dB}_{FD-CNG,mr\_scaled}$.

The optimal output of the first MSVQ stage is the list of N_{best} best candidate indexes in the MSE sense and their mean square errors in relation to the FD-CNG domain input signal N^{dB}_{FD-CNG}. The total number of entries in the first stage

codebook is denoted N_{st1} and the N_{best} value for FD-CNG MSVQ-quantization is set to 8. Each first stage candidate points to an $idx_{glob} \in [0 \ldots N_{st1} - 1]$ in the first stage global codebook.

To achieve an improved VQ efficiency in terms of both cycles and ROM-storage, the FD-CNG VQ search for the set of N_{best} candidates is performed in a slightly suboptimal fashion, wherein some candidates are obtained in an initial suboptimal pairwise inner search, and some are finally established in a low cost circular neighbour analysis post-optimization step. The 1st stage codebook is compactly stored in $N_{segm} = 4$ segments with different even truncation lengths in the DCT-II ($M = 24$) domain. The employed truncation lengths allow for different degrees of high frequency content between segments and the truncated transformed FD-CNG vectors with low pass character may be searched using fewer cycles. The employed truncation lengths are $N_{trunc,segm} = \{8, 10, 16, 18\}$ for segments $segm \in [0 \ldots N_{segm} - 1]$. The segment sizes are $N_{cand,segm} = \{16, 17, 17, 78\}$ and the total size of the codebook is $N_{st1} = \sum_{segm=0}^{N_{segm}-1} N_{cand,segm} = 128$. A set of N_{best} initial stage one candidates I_{cand} are established through employing N_{segm} pair-wise inner search loops with $N^{dB}_{FD-CNG,DCT-II}$ as target, where each of the N_{segm} segments provides $N_{best}/N_{segm} = 2$ initial candidates. Further, during the initial pairwise inner loop search, the mean square error of every possible codebook entry $i \in [0 \ldots N_{st1} - 1]$, is stored in an auxiliary vector $st1_{MSE}$ in preparation for the post-optimization step of the candidate selection.

To allow for low ROM storage and a low inner loops cycle count of the VQ-operation, the first stage FD-CNG vectors are stored in the format $m_{i,segm,col} \cdot 2^{exp_{segm,col}}$ where the mantissa $m_{i,segm,col}$ consists of only 8 bits and the exponent (a binary shift factor) $exp_{segm,col}$ is an integer. Two consecutive mantissas for column $col \in \left[0, 2, 4, \ldots, N_{trunc,segm} - 2\right]$ and the following column $col + 1$, may be efficiently fetched from ROM as a single 16-bit word , masked and shifted using the column specific integer shift factor of $exp_{segm,col}$ and $exp_{segm,col+1}$, where the integer shift factors are fixed for each of the four segments, *segm*.

In the post-optimization step, the N_{best} *initial* candidates in the list I_{cand} (with assistance of the auxiliary vector $st1_{MSE}$ containing stored MSE values) are analysed further by evaluating a set of global neighbours in relation to the two best MSE-performing candidates in I_{cand}. To initialize the post-optimization step, the indexes of the initial candidates in I_{cand} are first copied as starting points into the candidate list I_{final}. An off-line created circular neighbour list of closely located first stage VQ neighbours in the MSE sense with respect to each codebook entry is then traversed to evaluate and establish the final set of candidates in I_{final}. Let the a priori created circular neighbour index list be denoted $Lneighbour(j)$, $j \in [0 \ldots N_{st1} - 1]$, where $j'th$ the position contains a global index first stage value. I.e. with "%" signifying the modulo operation, the first stage vector index at index j is a close MSE neighbour to the vector indicated by the index at $Lneighbour((j-1)\ \%\ N_{st1})$ and also a close MSE neighbour to the vector indicated by the index at $Lneighbour\big((j+1)\%\ N_{st1}\big)$.

If an indicated neighbour to one of two best indexes in I_{cand} within a neighbour cardinal distance of two (in both forward and reverse directions of the circular list) in the list $Lneighbour(j)$, is found to be better in the MSE sense than the current worst candidate in I_{final}, then the worst performing candidate in I_{final} list is replaced with the found better neighbour.

The remaining stages in the FD-CNG-MSVQ are operating in the non-DCT transformed FD-CNG frequency band signal domain, thus the N_{best} codebook entries represented by the global stage 1 indexes in the list I_{final} are transformed back to the input FD-CNG domain using the inverse DCT-II ($M = 24$), then scaled down using the factor $FDCNG_{dctScale} (= 0.4202880859375)$, added to the mid vector $N^{dB}_{FD-CNG,midQ}$, and provided as candidates V_0 to the subsequent MSVQ stages (Equation(1398) in clause 5.6.3.5 of [3] with $k = 0$). The list of 1st stage output in the frequency band domain is provided to the 2nd stage as set of N_{best} vectors: $V_0(j, i), j \in [0 \ldots N_{best} - 1]$ and $i \in [0 \ldots M - 1]$. Further V_0 is accompanied by the list of corresponding first stage N_{best} global indexes in I_{final}.

The required N_{best} MSE's for each of the candidate indexes in I_{final} do not need to be recomputed prior to the next stage as the employed DCT-II(M=24) is an orthogonal transform, however as the inner loop is performed in a scaled domain, the MSE's for the second stage are scaled by $\left(\frac{1.0}{FDCNG_{dctInvScale}}\right)^2$ to match the input domain scaling.

5.2.2.3.5.4 Wideband pre-processing in the first stage FD-CNG MSVQ

For wideband input signals, the MSVQ FD-CNG target vector N^{dB}_{FD-CNG} has a reduced dimension of $M_{WB} = 21$. To be able to maintain the efficient search and reuse of the compactly stored stage one coefficients in case of WB input, the M_{WB} length FD-CNG signal is extended to $M = 24$ using a high-quality extrapolation method.

The extrapolation is performed by first subjecting the WB input FD-CNG signal to the shorter DCT-II (M=21) transform and then the resulting DCT-II coefficients are used as scaling factors for extrapolating the tail of the length M_{WB} basis vectors to a length of M via reflection. The top three ($(M - M_{WB})$ elements of the resulting scaled and reflected basis vectors of length M, are then summed up to provide an extended $N^{dB}_{FD-CNG,ext}$ target in the input FD-CNG signal domain. The basis-vector based extrapolation ensures that no analysis noise is added in the extended input to the subsequent DCT-II(M=24) transformation.

The WB first stage search is subsequently performed using the now extended $N^{dB}_{FD-CNG,ext}$ as target signal as described in clause 5.2.2.3.5.3 with the additional final step that the FD-CNG WB MSVQ first stage output vectors are truncated to $M = 21$ and the corresponding MSEs of the N_{best} candidates are updated to only include contribution from the first M_{WB} FD-CNG coefficients, this update is performed by subtracting the MSE(M=24) values by the contribution from the top three elements of the candidate quantized first stage FD-CNG domain vectors.

5.2.2.3.5.5 LP-CNG encoding - high band quantization

In LP-CNG, quantization of the high band energy is slightly modified with respect to EVS. First, the quantization step size used in equation (1355) of [3] is set to $\Delta=0.7$ for IVAS encoding modes instead of *0.9* as used in EVS. Additionally, a dead-zone interval around the high band energy is added to prevent frequent toggling of the quantization index.

5.2.3 Common audio coding tools

5.2.3.1 General

This clause describes common audio coding tools used for encoding the so-called transport channels. The individual tools can be used simultaneously multiple times similarly as a combination of different tools can be used simultaneously to encode different numbers of transport channels. The exact set-up depends on the actual IVAS format, IVAS total bitrate, and its actual mode and it is described in particular IVAS format related clauses.

5.2.3.2 Single Channel Element (SCE)

The Single Channel Element (SCE) is a coding tool that encodes one transport channel. It is based on one core-coder module. Its block diagram is shown in Figure 5.2-24.

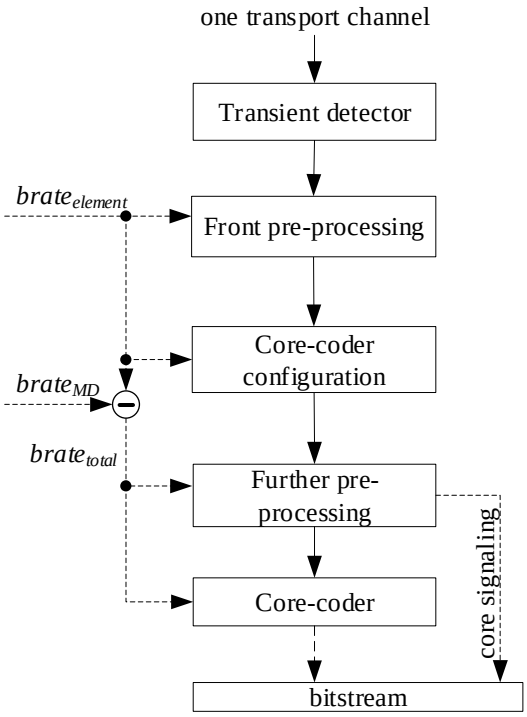


Figure 5.2-24: Block diagram of the Single Channel Element (SCE) encoder

As seen from Figure 5.2-24, the SCE encoder consists of the following modules:

(a) Temporal transient detector (clause 5.2.2.2.7) enables detection and therefore proper processing and encoding of transients in the transform-domain core-coder modules (TCX core, HQ core, FD-BWE).

(b) Front pre-processing (clause 5.2.2.2). Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$.

(c) Core-coder configuration module: high-level parameters like core-coder internal sampling-rate, core-coder nominal bitrate, IGF presence etc. are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-coder set-ups.

(d) Further pre-processing (clause 5.2.2.3.1).

(e) One variable bitrate core-coder (clause 5.2.2.3).

Note that the further pre-processing module (d) and the core-coder module (e) depend on the core-coder total bitrate $brate_{total}$ which is a difference between the element bitrate $brate_{element}$ and bitrate related to encode the metadata $brate_{MD}$ (if present), i.e.

$$brate_{total} = brate_{element} - brate_{MD}. \quad (5.2\text{-}241)$$

5.2.3.3 Channel Pair Element (CPE)

The Channel Pair Element (CPE) is a coding tool that encodes two transport channels. It is based on one or two core-coder module(s) supplemented by stereo coding tools. Its block diagram is shown in Figure 5.2-25.

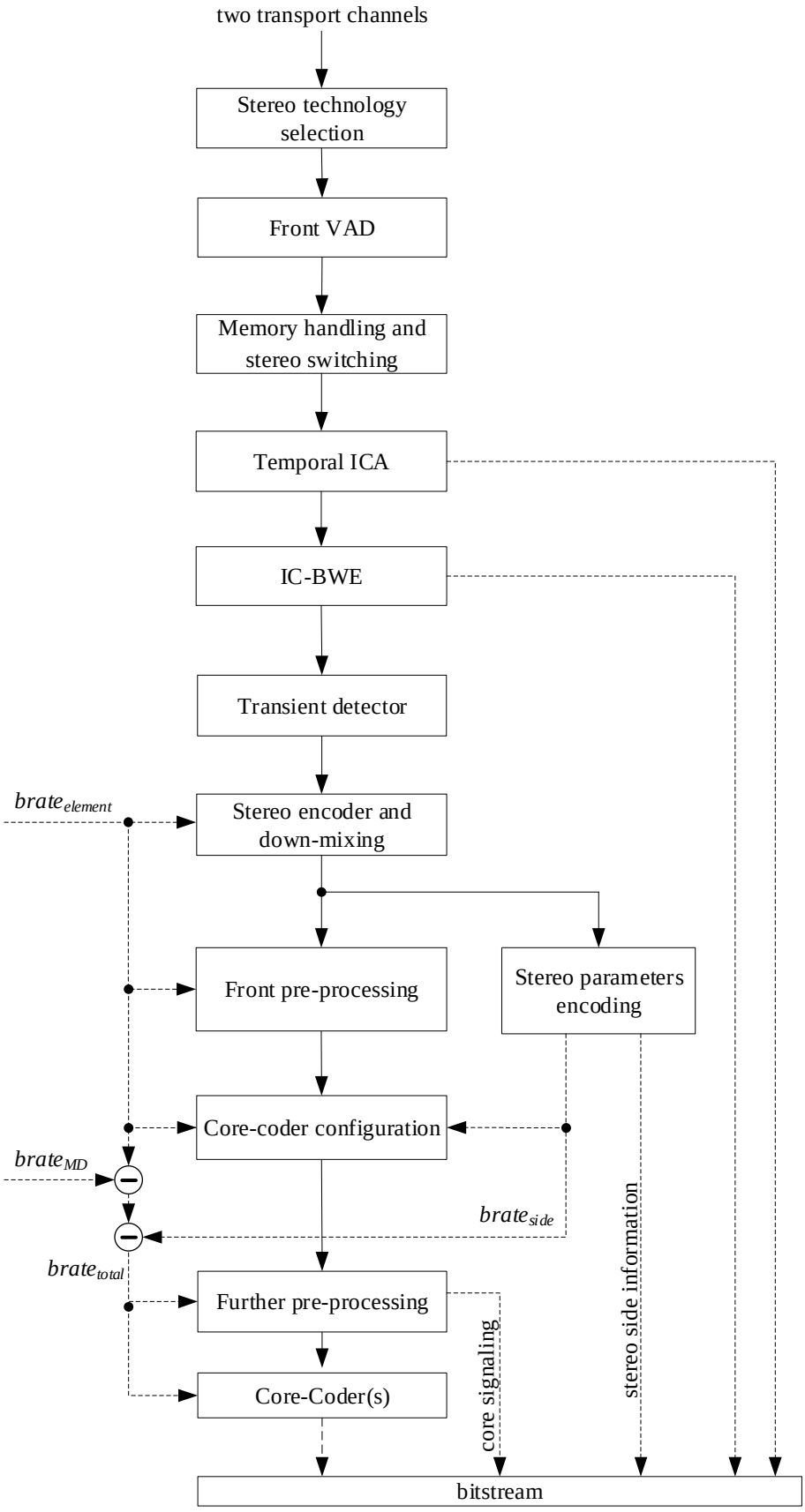


Figure 5.2-25: Block diagram of the Channel Pair Element (CPE) encoder

As seen from Figure 5.2-25, the CPE encoder consists of the following modules:

(a) Stereo technology selection module controls the selection between DFT stereo, TD stereo and MDCT stereo coding modes and it is based on the element bitrate, IVAS format, and information from the stereo classifier (clause 5.3.2.5).

(b) Front VAD module (clause 5.3.2.2.2) runs on both transport channels and its output is used in the DTX operation.

(c) Memory handling and stereo switching (clause 5.3.4) controls the switching between DFT stereo, TD stereo and MDCT stereo coding modes and comprises dynamic allocation/deallocation of static memory of stereo modes data structures depending on the current stereo mode. Also, the TD stereo mode is selected in this module.

(d) Temporal Inter-Channel Alignment (ICA) module to time-align the two transport channels (clause 5.3.2.3.1).

(e) Inter-Channel BWE (IC-BWE) module (clause 5.3.2.2.1) encode high-frequencies in DFT stereo and TD stereo modes.

(f) Temporal transient detector (clause 5.2.2.2.7), one per transport channel, enables detection and therefore proper processing and encoding of transients in the transform-domain core-coder modules (TCX core, HQ core, FD-BWE).

(g) Stereo processing and downmixing module in case of DFT stereo (clause 5.3.2.4) or TD stereo (clause 5.3.2.3) while the stereo parameters are encoded in the stereo parameters encoding module.

(h) Front pre-processing (clause 5.2.2.2) performed on one or two downmixed channels. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$.

(i) Core-coder configuration, one per downmixed channel: high-level parameters like core-coder internal sampling-rate, core-coder nominal bitrate, IGF presence etc. are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-coder set-ups.

(j) Further pre-processing (clause 5.2.2.3.1) performed on one or two downmixed channels.

(k) One or two variable bitrate core-coder(s) (clause 5.2.2.3) performed sequentially on one or two downmixed channels.

(l) Encoding of stereo parameters module depending on the current stereo mode being DFT stereo (clause 5.3.2.4) or TD stereo (clause 5.3.2.3).

Note that the further pre-processing module (j) and the core-coder module (k) depend on the core-coder total bitrate $brate_{total}[n], n = 1, 2$. The sum of total bitrates of both downmixed channels is then a difference between the element bitrate $brate_{element}$ and bitrates related to encode the metadata $brate_{MD}$ (if present) and stereo side information $brate_{side}$ (if present), i.e.

$$brate_{total}[0] + brate_{total}[1] = brate_{element} - brate_{MD} - brate_{side}. \quad (5.2\text{-}242)$$

5.2.3.4 Multichannel Coding Tool (MCT)

5.2.3.4.1 Summary of the system

The Multi-channel Coding Tool (MCT) is the coding tool for encoding more than two transport channels (i.e. 3 and above). MCT processing consists mainly of three processing blocks:

- a preprocessing stage where all individual channels to be coded are pre-processed in a way that all signals are perceptually whitened,
- the joint processing stage, where the whitened signals are adaptively chosen to be encoded jointly, depending on how correlated they are or are adaptively chosen to be encoded individually otherwise
- a post processing stage where all resulting signals are quantized and coded into a bitstream, where additionally the side information of the pre-processing and the joint coding is incorporated.

With MCT, the codec uses the flexibility of signal adaptive joint coding of arbitrary channels and reuses the concepts of MDCT-based stereo joint coding described in 5.3.3. These are:

- Use of perceptually whitened signals for further coding.
- Use of ILD parameters of arbitrary channels to efficiently code panned sources
- Flexible bit distribution among the processed channels based on the energy.

The codec uses Spectral Noise Shaping (SNS) to perceptually whiten the signal. The algorithm further normalizes the SNS-whitened spectrum towards the mean energy level using ILD parameters. Channel pairs for joint coding are selected in an adaptive manner, where the stereo coding consist of a band-wise M/S vs L/R decision. The band-wise M/S decision is based on the estimated bitrate in each band when coded in the L/R and in the M/S mode as described in 5.3.3.4.5. Bitrate distribution among the band-wise M/S processed channels is based on the energy.

5.2.3.4.2 MCT configurations

For the implementation of the MCT algorithm the incoming channels of the multi-channel signals are organized in *N/2* CPEs (described in clause 5.2.3.3), where *N* is the total number of transport channels, that are configured by default to the MDCT-based stereo configuration.

The bitrate assigned per CPE is given by:

$$brate_{CPE} = brate_{total}/N \cdot 2$$

If the number of transport channels are odd then the last CPE is assigned one channel and the second channel is set to channel mode IGNORE, signaling that any processing should be omitted for the particular CPE channel.

5.2.3.4.3 Encoder single channel preprocessing up to whitened spectrum

5.2.3.4.3.1 Overview

Each single channel of the multi-channel signal $s_i(t)$ is analyzed and transformed to a whitened MDCT-domain spectrum $S_i(k)$ following the processing steps as shown in the block diagram of Figure 5.2-26, as is done for the MDCT-based stereo input format described in 5.3.3.3.

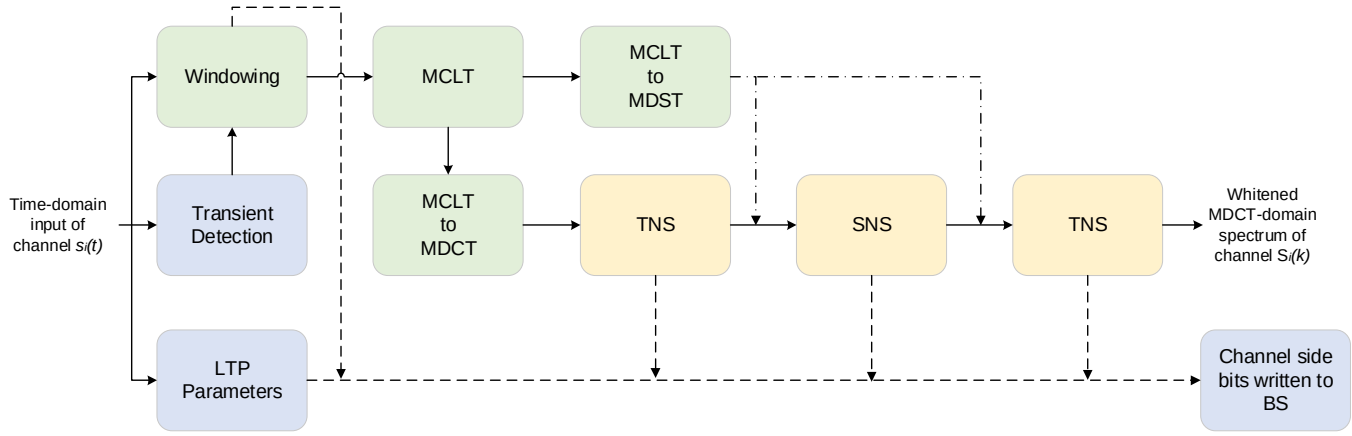


Figure 5.2-26: Block diagram of whitening preprocessing for each channel

The time-to-frequency domain conversion comprising the processing blocks of the time-domain transient detector, windowing, MDCT, MDST and OLA are described in clause 5.3.2 of [3]. MDCT and MDST form modulated complex lapped transform (MCLT); “MCLT to MDCT” represents taking just the MDCT part of the MCLT and discarding MDST and vice versa.

Temporal Noise Shaping (TNS) is done similar as described in clause 5.3.3.2.2 of [3]. Subsequently, Spectral Noise Shaping (SNS) is performed. SNS and the calculation of SNS parameters are described in clause 5.3.3.3.6. The order of the TNS and the spectral noise shaping (SNS) is adaptive. The existence of the 2 TNS boxes in the figures is to be understood as the possibility to change the order of the SNS and the TNS. The decision of the order of the TNS and the SNS is described in the MDCT-based stereo clause 5.3.3.3.5.2.

5.2.3.4.3.2 Detection of silent channels

The power spectrum that is computed to extract the SNS scale factors described in clause 5.3.3.3.6.3 is also used to determine whether a channel is silent and therefore, ignore it for the remaining processing to save on computational complexity but also bitrate. From the computed power spectrum, the total energy per channel as in:

$$E_i = \sum_{k=0}^{L_{frameTCX}} (X_i^{MDCT}[k])^2 + (X_i^{MDST}[k])^2$$

where $L_{frameTCX}$ is the full-band frame length in bins, $X_i^{MDCT}[k]$ and $X_i^{MDST}[k]$ are the MDCT/MDST spectra of the *i*-th channel.

Then depending on whether E_i is smaller than a given threshold SILENT_CHANNEL_THRES=100 the channel mode flag for the *i*-th channel is set to IGNORE or to REGULAR otherwise. For the channels set to the IGNORE flag, any further processing is omitted and are not counted to the active channels that are considered for the MCT joint encoding that is described in clause 5.2.3.4.4.

The channel mode flag is written in the bitstream with 1 bit to be used analogously at the decoder side.

5.2.3.4.4 Joint channel Encoding System Description

5.2.3.4.4.1 Overview

In the described system, after each channel is transformed to the whitened MDCT domain, signal-adaptive exploitation of varying similarities between arbitrary channels for joint coding is applied. From this procedure, the respective *channel-pair blocks* are detected and chosen to be jointly coded using a band-wise M/S transform as is done for stereo signals with the MDCT-based stereo approach as described in detail in clauses 5.3.3.4.5-5.3.3.4.7.

An overview of the encoding system is given in Figure 5.2-28. For simplicity block arrows represent single channel processing (i.e. the processing block is applied to each channel). Block “MDCT-domain analysis” is represented in detail in Figure 5.2-27.

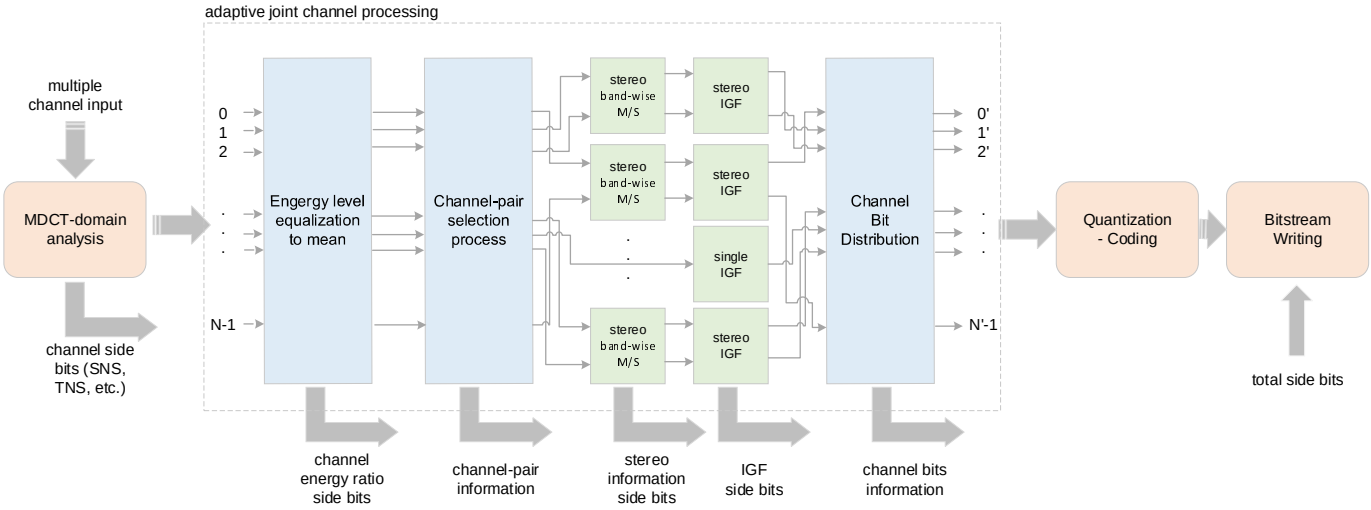


Figure 5.2-27: Block diagram of MCT and the adaptive joint channel processing

In the following clauses, the individual steps of the algorithm applied per frame are described in detail. A data flow graph of the algorithm described is given in Figure 5.2-28. For simplification in the graph, the cross-correlation vector is called “CC vector” and “CP” represents channel-pairs.

It should be noted, in the initial configuration of the system, there is a channel mask indicating for which channels the multi-channel joint coding tool is active. Therefore, for input where LFE channels are present, they are not considered in the processing steps of the tool.

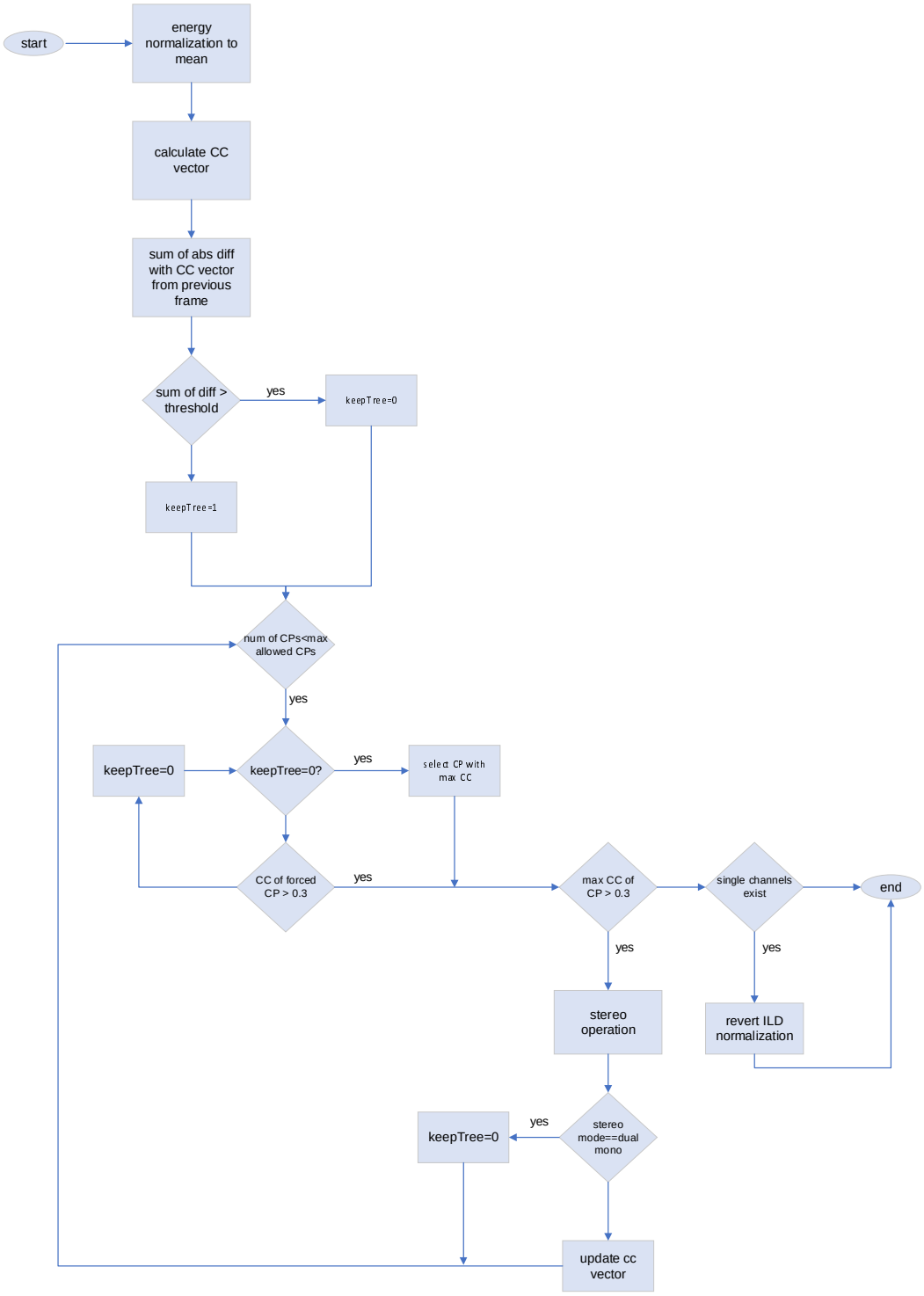


Figure 5.2-28: Data flow graph of MCT algorithm

5.2.3.4.4.2 Energy normalization of all channels towards mean energy.

An M/S transform is not efficient if ILD exists, that is if channels are panned. We avoid this problem by performing a broadband energy normalization, so that the preprocessed whitened spectra of each channel has a normalized energy to a mean energy level $\bar{E}$.

- Calculate energy E_i for each channel $i = 0, \ldots, N-1$, where N is the total number of channels.

$$\mathrm{E_i} = \sqrt{\sum_{\mathrm{k=0}}^{\mathrm{L_{frame}}} \mathrm{S}_i^{\mathrm{MDCT}}(\mathrm{k})^2} \qquad (5.2\text{-}243)$$

where L_{frame} is the total number of spectral coefficients of the current frame and $S_i^{MDCT}(k)$ is the respective preprocessed whitened MDCT spectrum of channel i ;

- calculate mean energy:

$$\bar{E} = \frac{\sum_{i=0}^{N} E_i}{N} \quad (5.2\text{-}244)$$

- normalize the spectrum of each channel towards mean energy $\bar{E}$.

  If $E_i > \bar{E}$ (downscaling)

$$\alpha_i = \frac{\bar{E}}{E_i} \quad (5.2\text{-}245)$$

  where a_i is the broadband normalization ratio.

  The broadband normalization ratio is uniformly quantized and sent to the decoder as side information bits, in the same way as is done for the ILD normalization in MDCT-based stereo and Equation (5.3-275).

$$\widehat{ILD}(i) = \max(1, \min(ILD_{RANGE} - 1, \lfloor (ILD_{RANGE} * a_i + 0.5) \rfloor) \quad (5.2\text{-}246)$$

  where $ILD_{RANGE} = 2^{ILD_{bits}}$ and $ILD_{bits} = 4$;

  then the quantized scaling ratio with which the spectrum is finally scaled is given by

$$\hat{\alpha}_i = \frac{I\bar{L}D(i)}{ILD_{RANGE}} \quad (5.2\text{-}247)$$

  if $E_i < \bar{E}$ (upscaling)

$$\alpha_i = \frac{E_i}{\bar{E}} \quad (5.2\text{-}248)$$

  and

$$\hat{\alpha}_i = \frac{ILD_{RANGE}}{I\bar{L}D(i)} \quad (5.2\text{-}249)$$

  where $\widehat{ILD}(i)$ is calculated as in Equation (5.2-246).

To distinguish whether we have downscaling/upscaling at decoder and in order to revert the normalization, besides the $\widehat{ILD}$ value for each channel, a 1-bit flag (0=downscaling/1=upscaling) is written in the bitstream as side information.

Finally, each channel is scaled using the computed scaling ratio $\hat{a}_i$ to obtain the normalized spectrum as follows:

$$\bar{S}_i^{MDCT} = \hat{a}_i \cdot S_i^{MDCT}$$

where $i = 0, \ldots, N-1$, and N is the total number of channels.

It should be noted that the same scaling is applied to the MDST spectrum of the respective channels S_i^{MDST} which is needed for computing the power spectrum after the joint processing for performing the IGF analysis.

5.2.3.4.4.3 Calculation of normalized inter-channel cross-correlation values for all possible channel-pairs

In this step, in order to decide and select which channel pair has the highest cross-correlation value and therefore is suitable to be selected as a pair for stereo joint coding, the inter-channel normalized cross-correlation value for each possible combination of channel pairs is calculated. The normalized cross-correlation value for each channel pair is given by the cross-spectrum as follows:

$$\tilde{r}(i_1, i_2) = \frac{r_{X_{i_1} X_{i_2}}}{\sqrt{r_{X_{i_1} X_{i_1}} r_{X_{i_2} X_{i_2}}}} \quad (5.2\text{-}250)$$

where

$$r_{X_{i_1} X_{i_2}} = \sum_{k=0}^{L_{frame}} \bar{S}_{i_1}^{MDCT}(k) * \bar{S}_{i_2}^{MDCT}(k) \quad (5.2\text{-}251)$$

and $i_1 = 0, \dots, N-1,\ i_2 = i_1, \dots, N-1$, L_{frame} is the total number of spectral coefficients per frame and $\bar{S}_{i_1}^{MDCT}$ and $\bar{S}_{i_2}^{MDCT}$ are the respective normalized MDCT spectra of the channel-pair i_1, i_2 under consideration.

The normalized cross-correlation values for each channel pair are stored in the cross-correlation vector

$$CC = [\tilde{r}_0, \tilde{r}_1, \dots, \tilde{r}_{P-1}] \tag{5.2-252}$$

where P is the maximum number of possible pairs and can be calculated by:

$$P = \big(N * (N-1)\big)/2 \tag{5.2-253}$$

As seen in Figure 5.2-26, depending on the transient detector we can have different block sizes (TCX10 or TCX20 window block sizes). Therefore, the inter-channel cross-correlation is calculated given that the spectral resolution for both channels is the same. If otherwise, then the value is set to 0, thus ensuring that no such channel pair is selected for joint coding.

An indexing scheme to uniquely represent each channel pair is used. An example of such a scheme for indexing six input channels is shown in Table 5.2-28.

Table 5.2-28: Indexing scheme of channel pairs

Channel Index	0	1	2	3	4	5
0		0	1	2	3	4
1			5	6	7	8
2				9	10	11
3					12	13
4						14
5						

To create the indexing scheme the logic in the following pseudo-code may be used:

```
For a given selected channel pair with channel indices chIdx1, chIdx2 respectively:
pairIdx = 0;
for ( ch2 = 1; ch2 < nChannels; ch2++ )
{
    for ( ch1 = 0; ch1 < ch2; ch1++ )
    {
       if ( ch1 == chIdx1 && ch2 == chIdx2 )
       {
           return pairIdx;
       }
       else
       {
           pairIdx++;
       }
    }
}
```

The same indexing scheme is held throughout the algorithm as is used also to signal channel pairs to the decoder. The number of bits needed for signaling one channel-pair amount to

$$bits_{idx} = \lfloor \log_2(P-1) \rfloor + 1 \tag{5.2-254}$$

5.2.3.4.4.4 Channel-pair selection and jointly coded stereo processing

After calculating the cross-correlation vector, the first channel-pair to be considered for joint-coding is with the highest normalized cross-correlation value and higher than a minimum value threshold of 0.3.

$$\text{idx}_{\max} = arg \max_{i|CC[i]>0.3} CC[i], \quad i = 0,1, \dots, P-1 \tag{5.2-255}$$

The selected pair of channels serve as input to the MDCT stereo encoding procedure, namely a band-wise M/S transform. For each selected pair, the decision whether the channels will be coded using full band joint encoding mode M/S, or full band discrete encoding mode L/R coding depends on the estimated bitrate for each case. For band-wise M/S joint encoding mode, for each spectral band (as in the bands defined in 5.3.3.4.2) the bits are estimated for coding M/S or discrete L/R and the final coding mode for the particular band is determined based on the results. Additionally,

the necessary number of bits to signal the respective M/S mask are added to the estimation. The coding mode that is less demanding in terms of bits is selected. This procedure is described in detail in subclause 5.3.3.4.5.

The output of this process results to an updated spectrum for each of the channels of the selected channel-pair as described in clause 5.3.3.4.7. It should be noted that the same stereo processing is applied to the MDST spectra of the selected channel-pair needed to calculate the respective power spectrum for the IGF analysis. Also, information that need to be shared with the decoder (side information) regarding this channel-pair are created, i.e., whichstereo mode is selected (Full M/S, dual-mono, or band-wise M/S) and if band-wise M/S is the mode selected the respective mask of indicating whether M/S coding is chosen (1) or L/R (0).

After the stereo processing of a selected channel-pair block the cross-correlation vector CC is updated accordingly. For the selection of channel pairs, there is no "cascading" allowed. That means, the output signals of the stereo processing of a channel-pair block cannot be reused as an input of another channel pair block combination. That is ensured, by while updating the cross-correlation vector, the values of the normalized cross-correlation for all channel combinations with the channels of a previous channel-pair block, are set to 0. Therefore, it is not possible to select a new channel-pair for which one of the channels was already part of an existing channel pair.

The process continues iteratively until either there are no more possible channel combinations for which the condition of Equation (5.2-255) applies or the maximum number of possible channel pairs $CP_{max} = \lfloor N/2 \rfloor$ is met.

It should be noted that there may be cases where the stereo operation of a selected channel-pair does not alter the spectra of the channels. That happens when the M/S decision algorithm decides the stereo encoding mode is dual-mono. This can happen especially in cases where the normalized cross-correlation value $\tilde{r}(i_1, i_2)$ is close to the minimum threshold. In this case, the arbitrary channels involved are not considered a channel-pair block anymore as they will be coded separately. Also, updating the cross-correlation vector will have no effect.

To continue with the process, the channel-pair with the next highest value is considered. The steps in this case continue as described above. The whole process is described in the data flow graph of Figure 5.2-28.

5.2.3.4.4.5 Retain channel pair selection (stereo tree) of previous frame

In many cases the normalized cross-correlation values of arbitrary channel-pairs can be close and therefore the selection can switch often between this close values. That may cause frequent channel-pair tree switching, which may result to audible instabilities to the output system. Therefore, it is opted to use a stabilization mechanism, where a new set of channel pairs is selected only when there is a significant change to the signal and the similarities between arbitrary channels change. To detect this, the cross-correlation vector of the current frame is compared with the vector of the previous frame and when the difference is larger than a certain threshold then the selection of new channel pairs is allowed.

The variation in time of the cross-correlation vector is calculated as follows:

$$\mathrm{CC}_{diff} = \sum_{i=0}^{P} \left| CC[i] - CC^{[-1]}[i] \right| \quad (5.2\text{-}256)$$

where P is the maximum number of possible combinations defined in Equation (5.2-253). If

$$CC_{diff} > 0.15 \cdot P \quad (5.2\text{-}257)$$

then the selection of new channel-pairs to be jointly coded, as described in the previous subclause 5.2.3.4.4.4, is allowed. Otherwise, the differences are small, then the same channel-pair tree as in the previous frame is used. For each given channel-pair, the band-wise M/S operation is applied as previously described. If, however, the normalized cross-correlation value of the given channel-pair does not exceed the threshold of 0.3 then the selection of new channel pairs creating a new tree is initiated.

5.2.3.4.4.6 Revert energy of single channels

After the termination of the iteration process for the channel pair selection and the pairwise stereo processing procedure, there may be channels that are not part of any channel-pair block and therefore, are coded individually. For those channels the initial normalization of the energy towards the mean energy is reverted to their original energy. Depending on the flag signaling upscaling or downscaling the energy of these channels are reverted using the inverse of the quantized scaling ratio $\frac{1}{\hat{a}_i}$ defined in clause 5.2.3.4.4.2. Additionally, $\widehat{ILD}(i) = 0$, signalling implicitly to the decoder that the i -th channel was coded separately and not as part of a channel-pair block.

5.2.3.4.5 IGF for multi-channel processing

After the joint stereo coding of the selected channel pairs, Intelligent Gap Filling (IGF) analysis is performed.

Regarding IGF analysis, in case of stereo channel pairs an additional joint stereo processing is applied, as is thoroughly described in 5.3.3.6.2. This is necessary, because for a certain destination range in the IGF spectrum the signal can be a highly correlated panned sound source. In case the source regions chosen for this particular region is not well correlated, although the energies are matched for the destination regions, the spatial image can suffer due to the uncorrelated source regions.

Therefore, for each channel pair stereo IGF is applied if the stereo mode of the core region is different to the stereo mode of the IGF region or if the stereo mode of the core is flagged as band-wise M/S. If these conditions do not apply, then single channel IGF analysis is performed. If there are single channels, not coded jointly in a channel-pair, then they also undergo a single channel IGF analysis.

Finally, for each channel of the multi-channel signal the respective side information from the IGF analysis processing is written to the bitstream.

5.2.3.4.6 Bitrate distribution

After the process of joint channel-pair stereo processing, each channel is quantized and coded separately by an entropy coder. Therefore, for each channel the available number of bits should be given. In this step, the total available bits are distributed to each channel using the energies of the processed channels.

The energy of each channel, the calculation of which is described above in Equation (5.2-243) for the normalization step, is recalculated as the spectrum for each channel may have changed due to the joint stereo processing. The new energies are denoted $\tilde{E}_i, i = 0,1, \dots, N$.

As a first step the energy-based ratio r_i, i.e the ratio of the energy of the channel towards the total energy of all channels, with which the bits will be distributed is calculated for each channel:

$$r_{\mathrm{i}} = \frac{\tilde{E}_{\mathrm{i}}}{\sum_{k=0}^{\mathrm{N}} \tilde{E}_{\mathrm{i}}}, \quad i = 0,1, \dots, N-1 \tag{5.2-258}$$

Here it should be noted, that in the case where the input consists also from an LFE channel, it is not taken into account for the ratio calculations. The LFE channel is coded separately in 5.7.2.2 before of the call to MCT to code the remaining channels and therefore the bits for coding the LFE are already subtracted from the bits available for coding the remaining channels The ratio is uniformly quantized:

$$\hat{r}_i = \max(1, \ \min(r_{RANGE} - 1, \lfloor r_{RANGE} \cdot r_i + 0.5 \rfloor)) \tag{5.2-259}$$

$$r_{RANGE} = 2^{r_{bits}} \tag{5.2-260}$$

The quantized bit distribution ratios $\hat{r}_i$ are encoded in the bitstream to be used from the decoder to assign the same amount of bits to each channel to read the transmitted channel spectra coefficients .

The bit distribution scheme is described below:

For each channel assign the initial minimum amount of bits required by the entropy coder $bits_{min}$.

The remaining bits, i.e., $bits_{remaining} = bits_{total} - \sum_{k=0}^{N} bits_{min,i}$ are distributed using the quantized ratio $\hat{r}_i$:

$$\mathrm{bits}_i = \frac{\hat{r}_i}{r_{RANGE}} \cdot \mathrm{bits}_{remaining} \tag{5.2-261}$$

Because of the quantized ratio the bits are approximately distributed and therefore it may be

$$\mathrm{bits}_{split} = \sum_{k=0}^{\mathrm{N}} bits_i \neq bits_{total}. \tag{5.2-262}$$

So in a second refining step the difference $bits_{diff} = bits_{split} - bits_{total}$ are proportionally subtracted from the channel bits $bits_i$:

$$\mathrm{bits}_i = \mathrm{bits}_i - \frac{\hat{r}_i}{r_{RANGE}} \cdot \mathrm{bits}_{diff} \tag{5.2-263}$$

After the refinement step if there is still a discrepancy of $bits_{split}$ in comparison with $bits_{total}$ the difference (usually very few bits) is "donated" to the channel with the maximum energy.

Finally, in the case where all channels are silent and are flagged to be ignored, as described in clause 5.2.3.4.3.2, all the bits are assigned to the first input channel..

The exact same procedure is followed from the decoder in order to determine the amount of bits to be read to decode the spectrum coefficients of each channel.

5.2.3.4.7 Quantization and coding of each channel

After having calculated the bit rate distribution, for each channel of the mutli-channel signal quantization and encoding follows with the assigned number of bits. Quantization, noise filling and the entropy encoding, including the rate-loop, are as described in 5.3.3.2 of [3] and 5.3.3.8 of this specification. The power spectrum (magnitude of the MCLT) is used for the tonality/noise measures in the quantization and IGF as described 5.3.3.2 of [3] and 5.3.3.8 of this specification. Since whitened and band-wise M/S processed MDCT spectrum is used for the power spectrum, the same SNS and M/S processing must be done on the MDST spectrum. Likewise, the same normalization scaling towards the mean energy must be done for the MDST spectrum as it was done for the MDCT. For the frames where TNS is active, MDST spectrum used for the power spectrum calculation is estimated from the whitened and M/S processed MDCT spectrum.

5.2.4 Common spatial metadata coding tools

5.2.4.1 General

Common spatial metadata encoding tools are methods shared by the SBA, MASA and McMASA coding modes for encoding their spatial metadata. The methods are steered by the configuration parameters received as in input and shown in Table 5.2-29.

Table 5.2-29: Main configuration parameters for spatial metadata coding

Configuration parameter name	Description	Operating point
nbands	Number of parameter subbands	SBA, MASA, McMASA
nblocks	Number of subframes	SBA, MASA, McMASA
start_band	Index of the first parameter band	SBA, MASA, McMASA
Ho_dirac	Flag indicating that Higher-order DirAC is used in SBA format	SBA

The nbands and nblock variables represent the number of frequency subbands that are coded and the number of subframes that are coded, respectively. They may differ from the total number of subbands, *B*, or total number of subframes *M* used in analysis or coming from the input.

5.2.4.2 Spatial metadata composition

The codec operates with spatial metadata as part of the input alongside audio transport channels or as being derived from the input audio data. This section covers the encoding of the spatial metadata parameters related to SBA or MASA input format, as well as some MASA based codec internal models. More precisely we describe in the following coding procedures for audio direction (azimuth, elevation), energy ratio, diffuseness, spread coherence, surround coherence. The number of directions for each method is specified for the cases where these methods are used.

Table 5.2-30: Main metadata configuration parameters for different formats

Operating point	Direction	Number of directions	Energy ratio/ Diffuseness	Spread coherence	Surround coherence
MASA	Yes	1 or 2	Yes	Yes	Yes
McMASA	Yes	1	Yes	Yes	Yes
SBA	Yes	1 or 2	Yes	No	No

In all the methods presented below we will consider the subbands indexed with *b=1:nbands* and the subframes indexed by *m=1:nblocks*. One time-frequency element is denoted as TF tile. The number of subbands and subframes are defined for each mode and operation point.

5.2.4.3 Direction metadata coding tools

5.2.4.3.1 Overview

This subclause defines the coding tools used in the spatial direction encoding methods described in later subclauses. These are low-level functions that will be referred to later in the description. They include the direction parameters (azimuth and elevation) quantization, the raw or fixed rate encoding, as well as methods for variable rate encoding of the direction parameters.

5.2.4.3.2 Direction metadata quantization

5.2.4.3.2.1 Joint azimuth elevation quantization

The spatial audio parameters, azimuth and elevation of each time-frequency (TF) tile are indexed to a point of a spherical grid. The indexing is performed by first joint quantizing the two spatial audio direction parameters, or direction metadata. The direction metadata quantization refers to the quantization of azimuth and elevation values for each time frequency tile of each frame. The direction parameters can be quantized on the spherical grid whose structure is defined for each of the following number of bits: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11. The spherical grid or codebook structure, for each number of bits is formed such that it approximates the covering of the sphere with smaller spheres, wherein the centres of the smaller spheres define the points of the spherical grid.

The number of points in the spherical grid is obtained from the number of horizontal circles, or elevation values on the sphere and the number of points or azimuth values on each circle. For each number of bits that can be used for quantizing the direction parameters, the number of elevation and azimuth values defining the spherical grid are presented in table 5.2-31:

Table 5.2-31: Structure of spherical grids for different bit resolutions

No. bits	No. elevation n_θ	Δ_θ	No. azimuth for each elevation $n_\phi[id_\theta], id_\theta = 1{:}n_\theta$
0	1	0	1
1	1	0	2
2	1	0	4
3	2	45	4, 2
4	2	45	8, 4
5	4	36	12, 7, 2, 1
6	5	25.5	14, 13, 9, 2, 1
7	6	20	22, 21, 17, 11, 3, 1
8	7	15	33, 32, 29, 23, 17, 9, 1
9	10	10.78	48, 47, 45, 41, 35, 28, 20, 12, 2, 1
10	14	6.7	60, 60, 58, 56, 54, 50, 46, 41, 36, 30, 23, 17, 10, 1
11	19	5	89, 89, 88, 86, 84, 81, 77, 73, 68, 63, 57, 51, 44, 38, 30, 23, 15, 8,1

The number of elevation points n_θ corresponds to the number of elevation points in the upper hemisphere including the value zero for elevation. Therefore, for each number of bits, there are $2\,n_\theta - 1$ codewords in the elevation codebook. The parameter Δ_θ corresponds to the distance in degrees between two consecutive elevation codewords. The values $n_\phi[id_\theta], id_\theta = 1{:}\,n_\theta$ specify the number of azimuth values for the quantized elevation with index id_θ at the corresponding number of bits. For instance, for 5 bits there are 4 elevation values in the upper hemisphere. For the first elevation value, corresponding to the horizontal circle on the Equator, there are 12 azimuth values, for the second elevation value there are 7 azimuth values and so on.

The direction quantization is performed by scalar quantizing first the elevation in the elevation codebook. One or two elevation value candidates are selected and for each of them the azimuth is scalar quantized in the corresponding azimuth codebook. The direction quantization is performed for each azimuth-elevation pair (ϕ, θ) using $B, 0 \le B \le 11$ bits. When performing the quantization, the value of B depends on the value of the corresponding quantized energy ratio index. This dependency is presented in the table 5.2-32:

Table 5.2-32: Direction parameters bit allocation as a function of the quantized energy ratio index

Index	0	1	2	3	4	5	6	7
bits_dir[i][j]	11	11	10	9	7	6	5	3

For 0 bits there is only one value for the elevation, the zero value, and only one value for the azimuth, the zero value. For 1 bit there is only one value for the elevation, the zero value, and two quantized values for the azimuth, 0 and -180, corresponding thus to front and back directions. For other values of the number of bits, the absolute value of the elevation is first scalar quantized on $n_\theta[B]$ levels where $n_\theta[B]$ corresponds to the value of n_θ for B bits in table 5.2-31. The $n_\theta[B]$ levels are chosen such that the first $n_\theta[B]-1$ levels are equidistant at the distance given by the corresponding Δ_θ value and the last codeword is at 90 degrees. At most two candidate elevation values, $\hat{\theta}_1, \hat{\theta}_2$ are chosen: the nearest neighbour and the next nearest neighbour in terms of indexes of the elevation codebook such that $\hat{\theta}_1 \leq \theta \leq \hat{\theta}_2$. The decision to select the nearest neighbour only, or both the nearest neighbour and the next nearest neighbour for the elevation is done as follows:

```
if n_θ[B] < 6 ∧ input format is MASA
{
    consider 2 elevation candidates
}
else
{
    consider 1 elevation candidate
}
```

For each of the 1 or 2 quantized elevation candidates the corresponding quantized azimuth values, $\hat{\phi}_1, \hat{\phi}_2$ are obtained by scalar quantizing the azimuth using the function quantize_phi() presented in clause 5.2.4.3.2.2.

If there are two candidates, for each of the two quantized candidate pairs, ($\hat{\theta}_1$, $\hat{\phi}_1$) and ($\hat{\theta}_2$, $\hat{\phi}_2$) the distance on the sphere between the original unquantized azimuth and elevation values and their quantized counterparts is calculated according to the following formula:

$$dist_j = -(\sin(\theta)\sin(\hat{\theta}_k) + \cos(\theta)\cos(\hat{\theta}_k)\cos(\phi - \hat{\phi}_k)),\ k = 1,2. \quad (5.2\text{-}264)$$

The above measure corresponds to the L2 norm of the distance between the original and quantized points on the sphere.

The candidate pair with smallest distance on the sphere is selected and their corresponding indexes in the elevation codebook, idx_θ, and in the azimuth codebook idx_ϕ, are stored.

After quantizing the elevation absolute value as above, the information about the sign is also added to the quantized elevation value and considered in the formation of the quantized elevation index. The elevation index is transformed to $idx_\theta := 2 * idx_\theta - 1$ if the elevation is positive and it is transformed to $idx_\theta := 2 * idx_\theta$ if the elevation is negative.

The spherical index of the resulting point in the spherical grid, idx_{sph} , is formed by enumerating the points on the grid starting from the frontal direction at zero elevation and zero azimuth, enumerating the point on the circle for elevation zero, then the points on the circle corresponding to the first positive elevation value, then the points on the circle corresponding to the first negative elevation value, then second positive, second negative and so on.

For the azimuth elevation pairs of each TF tile, after the quantization, the following indexes are further used in the encoding procedure: the elevation index, the azimuth index and the spherical index.

5.2.4.3.2.2 Azimuth quantization

The azimuth codebook is uniform and the number of points is given by the $n_\phi$corresponding to the quantized elevation value. The azimuth is differently quantized if it corresponds to a direction from MASA input format or to a direction from multichannel input format in McMASA representation. Both variants are presented in this clause.

If the input format is MASA, the azimuth points for two consecutive circles in the spherical grid are shifted by the value $shift$ such that for zero valued elevation the first azimuth codeword has value zero. For the following elevation value, the first azimuth codeword has value equal to half of the corresponding quantization step: $\frac{\Delta_{\phi_{id_\theta}}}{2} = \frac{360}{2\, n_\phi[id_\theta]}$. If the elevation values are numbered from 0, the first azimuth point on circles corresponding to even numbered elevation values are zero, while those corresponding to odd elevation values are $\frac{\Delta_{\phi_{id_\theta}}}{2}$. The shifting is considered only when the number of azimuth points on the horizontal circles on the sphere is larger than 2. The value $shift$ is thus defined by:

$$shift = \begin{cases} 0, if\ id_\theta\ is\ odd \\ \frac{\Delta_{\phi_{id_\theta}}}{2}, if\ id_\theta\ is\ even \end{cases} \quad (5.2\text{-}265)$$

The azimuth quantization for MASA input format is performed as shown in the following equation:

$$id_\phi = \left[\frac{\phi - 180 - shift}{\Delta_{\phi_{id_\theta}}}\right], 0 \le \phi < 360 \quad (5.2\text{-}266)$$

For obtaining the quantized value $\hat{\phi}$ and the final quantized azimuth index, the index id_ϕ goes through the following verifications that take into account that the azimuth values are cyclic:

```
if ( id_φ + (n_φ[id_θ] >> 1) < 0 ) id_φ =  id_φ + 1;
if ( id_φ  - (n_φ[id_θ]  >> 1) >= 0 ) id_φ = id_φ -( n_φ[id_θ]  >> 1 );
```

In addition, the quantized azimuth index is transformed with the following pseudo code, such that it has positive values for directions pointing to the left side and negative values for directions pointing to the right side. The frontal direction has value zero.

```
if ( id_φ  == -((n_φ[id_θ]  >> 1) + ( n_φ[id_θ]  % 2 )))
{
    id_φ  = id_φ + (n_φ[id_θ]  % 2);
}
else
{
    if ( id_φ  == ((n_φ[id_θ]  >> 1) + (n_φ[id_θ]  % 2)))
    {
        if (n_φ[id_θ]  % 2)
        {
            id_φ = id_φ - 1;
        }
        else
        {
            id_φ = id_φ - id_φ;
        }
    }
}
```

The quantized azimuth value is $\hat{\phi} = id_\phi * \Delta_{\phi_{id_\theta}}$. The domain of the quantized azimuth is consequently , $-180 \le \phi < 180$. After the value of the quantized azimuth is calculated, an additional transformation is performed on the quantized azimuth index ordering the quantized values such that the first index is assigned to the zero quantized azimuth value followed by alternating positive and negative azimuth values.

If the input format is multichannel and the encoding is done through the McMASA representation (described in 5.7.3), the azimuth quantization is preceded by a companding function. The compading function definition depends on the multichannel input format type and on absolute elevation value being larger than a threshold or not. The companding function is defined for the input domain [-180, 180) and is specified below:

```
if multichannel format is 5.1 or 5.1+2 or 5.1+4
{
    pointsA = { 0.0, 50.0, 90.0, 150.0, 180.0};
    pointsB = { 0.0, 90.0, 110.0, 170.0, 180.0};
}
else
{
    pointsA = { 0.0, 60.0, 110.0, 150.0, 180.0};
    pointsB = { 0.0, 90.0, 110.0, 170.0, 180.0 };


if ( absolute elevation > 40)  ∧ (multichannel format is 5.1+2 or 5.1+4 or 7.1+4)
{
    pointsA = { 0.0, 30.0, 80.0, 150.0, 180.0 };
    pointsB = { 0.0, 10.0, 100.0, 170.0, 180.0};
}
find k = 0:4 such that  abs(φ) ≤ pA[k + 1]
calculate comp(φ) = sign(φ) (pB[k] + (pB[k+1] − pB[k]) / (pA[k+1] − pA[k]) (abs(φ) − pA[k]))
```

The companded signed value is then scalar quantized and the signed index id_ϕ is transformed such that there are exactly the number of values:

```
if ( id_phi + ( n_φ[id_θ] >> 1 ) < 0 )
{
    id_phi += 1;
}
if ( id_phi - ( n_φ[id_θ] >> 1 ) >= 0 )
{
    id_phi = -( n_φ[id_θ] >> 1 );
}
if ( id_phi == -( ( n_φ[id_θ] >> 1 ) + ( n_φ[id_θ] % 2 ) ) )
{
    id_phi += ( n_φ[id_θ] % 2 );
}
else
{
    if ( id_phi == ( ( n_φ[id_θ] >> 1 ) + ( n_φ[id_θ] % 2 ) ) )
    {
        if ( n_φ[id_θ] % 2 )
        {
            id_phi -= 1;
        }
        else
        {
            id_phi = -id_phi;
        }
    }
}
```

The resulting index is used for obtaining the quantized azimuth value in the companded domain $\hat{\phi}_c = id_\phi * \Delta_{\phi\, id_\theta}$ and then inverse companded to obtain the real quantized azimuth value.

The azimuth quantization and indexing are wrapped together under *quantize_phi*() function that will be referred in later clauses.

5.2.4.3.3 Direction metadata raw encoding

After the direction quantization operation presented in 5.2.4.3.2 for each TF tile, the obtained spherical index can be represented on exactly the number of bits that have been used at the quantization. The raw encoding of the direction is performed by writing these indexes in the bitstream, for each subband and each subframe using the allotted number of bits.

5.2.4.3.4 Direction metadata entropy encoding tools

5.2.4.3.4.1 Quasi-uniform encoding

The *encode_quasi_uniform(value, alphabet_size)* function is used to encode $value$ with quasi-uniform probability using a punctured code. For $value \in \{0, \dots, alphabet\_size - 1\}$, a number of the smallest ones are encoded using $\lfloor log_2(alpbabet\_size) \rfloor$ bits, and the rest using $\lfloor log_2(alpbabet\_size) \rfloor + 1$ bits. If $alphabet\_size$ is a power of two, binary coding results. The function is defined in pseudo-code as:

```
nbits = floor(log2(alphabet_size))
thresh = 2 ^ (nbits + 1) - alphabet_size
if (value < thresh)
    write_bits(value, nbits)
else
    write_bits(value + thresh, nbits + 1)
```

If *alphabet_size* is a power of 2, then *alphabet_size = 2 ^ bits*, and *thresh = 2 ^ bits*, therefore the else branch is never used, and binary coding results. Otherwise, the first *thresh* smallest values are encoded using a binary code having *nbits* bits, and the rest, starting with *value = thresh*, are encoded using a binary code having *nbits + 1* bits. The first binary code encoded using *nbits + 1* bits has the value *value + thresh = 2.thresh*, therefore the decoder can figure out, by reading only the first *nbits* bits and comparing its value with *thresh*, if it needs to read one more additional bit.

5.2.4.3.4.2 Extended Golomb-Rice coding tool

The extended Golomb-Rice entropy coding tool is based on Golomb-rice coding with an integer parameter $p \geq 0$, to code an unsigned integer u. In traditional Golomb-rice coding, first, u is split into the least significant part with p bits, $u\_lsp =$

$u \bmod 2^p$, and the most significant part $u\_msp = \lfloor u \div 2^p \rfloor$. The most significant part is coded in unary, using u_msp 1-bits and a terminating zero bit. The least significant part is coded in binary.

Because arbitrarily large integers can be coded, some coding efficiency may be lost when the actual values to be coded have a known and relatively small alphabet size. The Extended Golomb-Rice method combines three improvements over the traditional Golomb-Rice coding, for coding a vector of values, each with a known and potentially different alphabet size u_alph. First, the alphabet size of the most significant part can be computed as $u\_msp\_alph = \lceil u\_alph \div 2^p \rceil$. If the maximum possible value of the most significant part is coded, $u\_msp\_alph - 1$, the terminating zero bit can be eliminated, because this condition can be implicitly detected at the decoder side, knowing the alphabet size as well. This is called the limited Golomb-rice coding. Additionally, for the same case when $u\_msp = u\_msp\_alph - 1$, the alphabet size of the least significant part u_lsp, which can be computed as $u\_alph - (u\_msb\_alph - 1) \cdot 2^p$, may be smaller than 2^p, allowing to use advantageously the *encode_quasi_uniform()* function instead of binary coding with p bits. The function *encode_quasi_uniform()* is also advantageously used when a particular value u has an alphabet u_alph smaller than 2^p. Finally, when $u\_msp\_alph \leq 3$ the Limited Golomb-Rice method produces codes having only two possible lengths, $1 + p$ and $2 + p$ bits. Once again, the function *encode_quasi_uniform* function is optimal for up to two lengths, and therefore it is used instead in this case.

The final pseudo-code of the function *encode_extended_gr()*, looks like:

```
Encode_extended_gr():
     u_msb_alph = ( alphabet_size + ( 1 << p ) - 1 ) >> p
    if ( u_msb_alph <= 3 )
    {
    encode_quasi_uniform( u, alphabet_size );
    }
    else
    {
        u_msb = u >> p;
        u_lsb = u & ( ( 1 << p ) - 1 )
        write_bit(1) x u_msb  //Leading MSB 1-bits
        if ( u_msb < u_msb_alph_size - 1 )
        {
             write_bit(0) x 1 // Terminating 0-bit for MSB
            write_binary(u_lsb) // LSB binary code
        }
        else
        {
            encode_quasi_uniform( u_lsb, alphabet_size - ( ( u_msb_alph - 1 ) << p ) );
        }
    }
```

5.2.4.4 Diffuseness and energy ratio coding methods

5.2.4.4.1 Diffuseness and energy ratio definitions

Directional component energy ratios and diffuseness parameter are quantized and coded using the same set of coding tools. The diffuseness can be interpreted as the complement of the ratio of the sum of directional energies to the total energy, and it is used to deduce the energy ratios of the directional components. If several directional components are transmitted, the ratios of their respective energies to the total energy are additionally transmitted, but only for the *N-1* directional components, where *N* is the total number of directional components. The remaining directional energy ratio can be deduced from the diffuseness and the transmitted energy ratios.

It is important to note that the diffuseness and energy ratio parameters are quantized and coded with different time resolution than the direction parameters. The diffuseness and energy ratio parameters are transmitted once per frame with a time of resolution of 20 ms, while the direction parameters are transmitted *nblocks* per frame.

5.2.4.4.2 Diffuseness parameter quantization

Each diffuseness parameter bounded between 0 and 1 is quantized to one of the discrete levels, using a non-uniform quantizer producing the diffuseness index. The quantizer thresholds and levels were derived from the well-known properties of the quantization sensitivity of the Inter-Channel Coherence (ICC), which is more sensible to quantization error towards a coherence of 1 than a coherence of 0. The diffuseness, which can be seen as of a complementary nature

of the coherence, shows then an inverse property: the diffuseness is more sensitive to quantization error toward 0. Moreover, a diffuseness value of 1 is avoided to be coded and transmitted because of sound rendering consideration.

Two resolutions are possible for quantizing the diffuseness parameters, using 8 and 16 levels. The two quantizers are defined by the number of their thresholds and reconstruction levels given in table 5.2-33:

Table 5.2-33: Diffuseness quantization thresholds and levels

Number of levels (*diffuseness_levels*)	Number of bits in raw coding	Thresholds (*diffuseness_thresholds*)	Reconstruction levels
8	3	0.0 0.01904296875 0.06298828125 0.119384765625 0.22119140625 0.399169921875 0.547607421875 0.734619140625 2.0	0.0, 0.03955078125, 0.089599609375, 0.158935546875, 0.308349609375, 0.473876953125, 0.63232421875, 0.85009765625
16	4	0.0, 0.009521484375 0.01904296875 0.0410156250 0.06298828125 0.0911865234375 0.119384765625 0.1702880859375 0.22119140625 0.3101806640625 0.399169921875 0.473388671875 0.547607421875 0.641113281250 0.734619140625 0.8673095703125 2.0	0.00 0.0142822265625 0.030029296875 0.052001953125 0.07708740234375 0.10528564453125 0.14483642578125 0.19573974609375 0.26568603515625 0.35467529296875 0.436279296875 0.510498046875 0.5943603515625 0.6878662109375 0.80096435546875 0.93365478515625

A placeholder out-of-range large threshold value (2.0) is added at the end of thresholds to make searching it simpler. The quantization of the diffuseness values is then achieved using the following simple procedure:

```
for ( diffuseness_index = 0; diffuseness_index < diffuseness_levels; diffuseness_index ++ )
      if ( diffuseness_value < diffuseness_thresholds [diffuseness_index + 1] )
            return diffuseness_index;
```

5.2.4.4.3 Diffuseness and energy ratio indices coding

The quantized diffuseness indices are coded using one of the three available methods: raw coding, one value only, and two consecutive values only.

By default, raw coding is used, and the method is signaled by setting to 1 the *diff_use_raw_coding* bit flag, which indicates whether the raw coding method is used. For raw coding, each diffuseness index value is encoded using the *encode_quasi_uniform()* function.

If all index values are equal, the one value only method is used, and *diff_use_raw_coding* is set to 0. A second bit (*diff_have_unique_value*) is used to indicate this method, then the unique value is encoded using the *encode_quasi_uniform()* function.

Else if all index values consist only of two consecutive values, the two consecutive values only method is used, indicated by setting *diff_use_raw_coding* and *diff_have_unique_value* to zero. The smaller of the two consecutive values is encoded using the *encode_quasi_uniform* function, taking into account that its alphabet size is reduced to $diff\_alph - 1$. Then, for each value, the difference between it and the minimum value is encoded on one bit.

Table 5.2-34 summarizes the different coding schemes, as well the resulting used coding tools, where *N* represents the *nbands-start_band* diffuseness parameters to code:

Table 5.2-34: Diffuseness coding methods, associated with their signaling and the subsequent coding. N represents the number of diffuseness indices to code.

Method	*diff_use_raw_coding* bit	*diff_have_unique_value* bit	Coding of diffuseness indixes
Raw coding	1	--	*N* times *encode_quasi_uniform()*
One value only	0	1	*encode_quasi_uniform(value, alphabet_size)*
Two consecutive values	0	0	*encode_quasi_uniform(value_min, alphabet_size-1)* Nx (value – value_min) on 1 bit

The diffuseness indices transmitted are important for the subsequent quantization and coding of the direction metadata. It will dictate the accuracy of direction quantification, following the principle that more the diffuseness is high, less important is the direction accuracy.

5.2.4.4.4 Diffuseness and energy ratio coding with two concurrent directions

5.2.4.4.4.1 Coding method overview

In the case of two sets of directional parameters, the diffuseness (or the equivalent diffuse-to-total energy ratio) and the direct-to-total energy ratio parameters are quantized and transmitted together in a combined scheme to save bits and improved quantization accuracy.

The general process is that, first, a diffuseness ratio is quantized into an index value. Then, the index value of the quantized diffuseness ratio is used to select the number of bits for quantizing a second ratio parameter. This second ratio parameter and its quantization depends on whether the metadata codec is used in Ho-DirAC mode or in MASA format metadata mode.

In case of Higher-order DirAC, a diffuseness and the direct-to-total ratio is given. The direct-to-total energy is quantized on a number of bits which depends on the quantized diffuseness index as shown in table 5.2-35.

Table 5.2-35: Number of bits to code direction energy ratio in Ho-DirAC

diff_index	≥ 6	≥ 4 && < 6	< 4
dfRatio_bits	1	2	3

The direct-to-total ratio is then quantized uniformly between 0 and 1, with step of *1/(2^dfRatio_bits-1)*.

In the case of MASA format metadata mode, a following specific method is applied for each time-frequency tile containing two directional parameter sets. As energy ratio parameters are averaged into a same value for each subframe within a frequency band in preliminary processing, further processing is done only on frequency bands when it is possible to save complexity.

As a preliminary step, the metadata of the two directional parameter are organized such that the direct-to-total energy ratio of the first direction r_{dir1} is always equal or larger than the direct-to-total energy ratio of the second direction r_{dir2}, that is, $r_{dir1} \geq r_{dir2}$. The process of reordering is described in clause 5.5.3.2.9.

First, diffuseness ratio r_{diff} is obtained with the relation:

$$r_{diff} = 1 - (r_{dir1} + r_{dir2})$$

Then a distribution factor of the two direct-to-total energy ratios (dfRatio in tables) is calculated as

$$r_{df} = \frac{r_{dir1}}{r_{dir1} + r_{dir2}}$$

In the special case of $r_{dir1} + r_{dir2} \leq \epsilon$, then the distribution factor $r_{df} = 0.5$. In this special case it is assumed that the direct-to-total energy ratios are equal.

The r_{diff} is then quantized using the method described in clause 5.2.4.4.2. The quantized diffuseness is represented by the diff_index parameter. Using the diff_index, the number of bits for quantizing r_{df} is selected with a function using

thresholds that fulfill the table 5.2-36. In practice, higher r_{diff} results in less bits used to quantize r_{df}. The selected dfRatio_bits value is then used to select a corresponding uniform scalar quantizer for quantizing the r_{df}. The r_{df} is then quantized using the selected uniform scalar quantizer to obtain dfRatio_index parameter which represents the quantized r_{df}. The diff_index and dfRatio_index values are then entropy encoded to the bitstream as described in clauses 5.2.4.4.3 and 5.2.4.4.4.2.

Table 5.2-36: Number of bits to code direction energy ratio in MASA

diff_index	≥ 6	≥ 3 && < 6	< 3
dfRatio_bits	1	2	3

5.2.4.4.4.2 Distribution factor ratio coding for two concurrent directions

Encoding the dfRatio_index values are done similarly as encoding diff_index values but with adaptation to what methods are available and what alphabet size is used depending on the realized number of bits used to quantize each r_{df}. As the number of raw bits known, it is possible to allow only those methods to be used that can offer benefit in compression.

First, total number of bits for raw coding raw $bits_{df-raw}$ is calculated as a sum of dfRatio_bits over all coding bands. Next, maximum value is selected from dfRatio_bits values in the coding and assigned to $bits_{df-max}$ to represent the largest used number of bits to code any r_{df} for this metadata frame.

The dfRatio_index coding algorithm is then configured to use the methods in table 5.2-34 as follows:

- If $bits_{df-raw} \geq bits_{df-max} + 2 + B_{coding}$, then all methods are used.
- Else if $bits_{df-raw} \geq bits_{df-max} + 1$, then "Two consecutive values"-method is not used
- Else, only raw coding mode is used

Here, B_{coding} represents the total number of used coding frequency bands that is configured for the metadata codec.

The alphabet_size for the coding methods is set to $2^{bits_{df-max}}$ Signalling different coding methods is also done, when necessary, that is, the coding method is in use. Otherwise, the dfRatio_index is encoded similarly as the diff_index.

5.2.4.4.4.3 Direction quantization bits adjustment with two concurrent directions

In the case of two sets of directional parameters and MASA format metadata for each time-frequency tile, the quantization resolution for quantizing the direction parameters need to be determined. This is due to the combined effect of the energy ratio model of MASA format metadata in encoding and the initial quantization resolution assignment based on the quantized direct-to-total energy ratio index. The energy ratio model of MASA format metadata states that

$$r_{dir1} + r_{dir2} + r_{diff} = 1$$

and the initial quantization resolution assignment based on the quantized direct-to-total energy ratio index assigns bits for direction parameter quantization based on the value given by the index of the quantized direct-to-total energy ratio for the direction. For two directions which are essentially equal, that is, $r_{dir1} \approx r_{dir2}$, neither of the directions can individually have a high direct-to-total energy ratio, and due to the non-linear quantization of the direct-to-total energy ratios, the quantization accuracy of the direction is not high enough.

This combined effect is compensated with the following approach. The following process is done separately for each coding band which has two directional spatial audio parameter sets, and the process affects the quantization spatial resolution of the direction parameters. First, the diff_index and dfRatio_index values are decoded to quantized diffuse-to-total ratio $\hat{r}_{diff}$ and quantized distribution factor $\hat{r}_{df}$ respectively. The decoding of the index values is described in clause 6.2.4.2 where the index values are decoded to give the decoded quantized values $\hat{r}_{diff}$ and $\hat{r}_{df}$.

The first quantized direct-to-total energy ratio is obtained as

$$\hat{r}_{dir1} = \hat{r}_{df}\left(1 - \hat{r}_{diff}\right)$$

The second quantized direct-to-total energy ratio is obtained as

$$\hat{r}_{dir2} = 1 - \hat{r}_{diff} - \hat{r}_{dir1}$$

Then, an inverted index is determined for both of the quantized direct-to-total energy ratios by re-quantizing values $1 - \hat{r}_{dir1}$ and $1 - \hat{r}_{dir2}$ using the diffuseness quantizer described in clause 5.2.4.4.2. This provides the indices index_dirRatio1Inv and index_dirRatio2Inv respectively.

Next, a combined ratio is determined by taking a sum of the quantized ratios:

$$\hat{r}_{comb} = \hat{r}_{dir1} + \hat{r}_{dir2}$$

Then, the larger of the two quantized direct-to-total energy ratios is selected. When $\hat{r}_{dir1}$ is larger than (or equal to) $\hat{r}_{dir2}$, $\hat{r}_{dir1}$ is selected and a set of modified direct-to-total energy ratios is obtained as follows:

$$r_{dir1-mod} = \hat{r}_{comb}$$
$$r_{dir2-mod} = \frac{\hat{r}_{dir2}}{\hat{r}_{dir1}} \hat{r}_{comb}$$

When $\hat{r}_{dir2}$ is larger and consequently selected, the indices in the two above equations are turned the other way round resulting in

$$r_{dir2-mod} = \hat{r}_{comb}$$
$$r_{dir1-mod} = \frac{\hat{r}_{dir1}}{\hat{r}_{dir2}} \hat{r}_{comb}$$

The modified direct-to-total energy ratios are then used to determine the modified inverted ratio indices index_dirRatio1 Inv_mod and index_dirRatio2Inv_mod by using the diffuseness quantizer to quantize values $1 - r_{dir1-mod}$ and $1 - r_{dir2-mod}$ respectively.

Next a check is performed where both modified inverted ratio indices are compared to their respective original indices as a difference in order to limit the maximum difference caused by the modification. In practice, this is expressed as

- If, index_dirRatio1Inv - index_dirRatio1Inv_mod > MASA_DIR_RATIO_COMP_MAX_IDX_STEPS, then index_dirRatio1Inv = ratio_index_1 - MASA_DIR_RATIO_COMP_MAX_IDX_STEPS
- Else, index_dirRatio1Inv is unmodified

Same is done for the second direction. Thus, the difference of the energy ratios is limited as well based on the difference between the modified and non-modified values. These modified energy ratio indices are then used in the algorithm presented in clause 5.2.4.3.2 to select bits used for quantizing the two spatial direction parameters associated with the time-frequency tile, and the bits in turn define the quantization spatial resolution for the two direction parameters of the time-frequency tile in the encoding process.

It should be noted that although Ho-DirAC mode encodes two directional parameter sets per frequency band, the modified ratio scheme is not used for it as the energy ratio model is different.

5.2.4.5 Direction metadata coding methods

5.2.4.5.1 Entropy coding 1 (EC1)

The Entropy Coding 1 (EC1) is the first method used to attempt to encode the direction metadata. It has two coding modes, a full raw coding mode and a frame-wise entropy coding mode, and then selects the most efficient mode producing the fewest bits. Unlike other entropy coding methods, EC1 does not re-quantize direction parameters. It losslessly codes the previously quantized direction indices, derived from the spherical quantization solely selected by the quantized diffuseness indices.

5.2.4.5.1.1 Full raw coding mode

The full raw coding mode sequentially code, without any modelling, each spherical index or azimuth index of the time-frequency grid. The spherical indices are coded from the lowest frequency band to the highest, and for each frequency band from the earliest time block to the latest.

In case of 3D audio scene, the spherical indices are coded within an integer number of bits, using a resolution depending on the diffuseness quantization index as described in table 5.2-37. The spherical indices are then written sequentially in the order as described above. The bit consumption is then easily estimated as:

$$direction\_bits\_raw = \sum_{b=start\_band}^{nbands} nblocks * spherical\_idx\_bits \quad (5.2\text{-}267)$$

In case of 2D audio scene, i.e. if the current configuration takes into account only the horizontal equatorial plane, the elevation is always set to 0, and only the azimuth indices have to be transmitted. It is achieved by using the *encode_quasi_uniform()* coding tool. Considering the azimuth alphabet size used for each frequency band, the bit consumption of the raw coding is then given by:

$$direction_{bits_{raw}} = \sum_{b=start\_band}^{nbands-1} \sum_{n=0}^{nblocks-1} encode\_quasi\_uniform\_length(az\_idx(b,n), az\_alph(b)) \quad (5.2\text{-}268)$$

5.2.4.5.1.2 Frame-wise entropy coding mode

The frame-wise entropy coding mode uses some modelling to code the quantized directions, which are grouped into two categories. The first group contains the quantized directions for diffuseness indices $diff\_idx(b) \leq ec\_max$ which are entropy coded, and the second contains the quantized directions for diffuseness indices $diff\_idx(b) > ec\_max$ which are raw coded, where ec_max is a threshold optimally chosen depending on $diff\_alph$. This approach implicitly excludes from entropy coding the frequency bands with high diffuseness, when frequency bands with low to medium diffuseness are also present in a frame, to avoid mixing statistics of the residuals. For a mixed diffuseness frame, raw coding is always used for the frequency bands with high diffuseness. However, if all frequency bands have high diffuseness, $diff\_idx(b) > ec\_max$, the threshold is set in advance to $ec\_max = diff\_alph$ in order to enable entropy coding for all frequency bands.

For the entropy-coded group, an average direction vector is first derived, by converting each quantized direction which is entropy coded back to a direction vector in cartesian coordinate and computing their sum. From the average direction an average elevation el_avg and azimuth az_avg are obtained. These two values are then quantized using the best angular precision deg_req used by the quantized directions which are entropy coded, denoted by deg_req_avg, which is the required angular precision corresponding to the smallest diffuseness index $diff$_idx_min $= min(diff\_idx(b))$, for $b \in \{0, \dots, nbands - 1\}$ and $diff\_idx(b) \leq ec\_max$. In case of High-Resolution mode, used on certain MASA modes, $diff$_min $_idx$ is set to zero, for getting the maximum quantization accuracy possible to quantize the average direction. The average direction is then used as a predictor for entropy-coded group directions.

Average direction quantization and subsequent predictive coding are performed separately for elevation and azimuth.

5.2.4.5.1.3 Elevation entropy-coding

The average elevation alphabet and codebook are derived from the angular resolution and number of angles theta (no_theta), which are defined in Table 5.2-37 as a function of the diffuseness index:

Table 5.2-37: Definition of the elevation angular resolution and no_theta in function of the diffuseness index and number of bits allocated to the spherical indexing

Diff_idx	0	1	2	3	4	5	6	7
Spherical_bits	11	11	10	9	7	6	5	3
No_theta	19	19	14	19	6	5	4	2
Delta_theta (degree)	5	5	6.7	10.78	20	25.5	36	45

The elevation codebook is then built delta_theta[diff_idx].ele_idx, where ele_idx is defined as the integers between -no_theta[diff_idx] +1 and no_theta[diff_idx]-1. The alphabet size of the codebook is then avq_elevation_alphabet = 2no_theta[diff_idx]-1.

The absolute value of the average elevation is quantized with nearest neighbor from the elevation codebook, and the sign if finally coded using the following rule:

```
Avg_elevation_idx = ( avq_elevation_alphabet >> 1) – avg_abs_elevation_index if avg_elevation <0
Avg_elevation_idx = ( avq_elevation_alphabet >> 1) + avg_abs_elevation_index if avg_elevation >=0
```

For each direction to be entropy coded, the dequantized average elevation q_el_avg is projected using the precision of that elevation to code, to obtain predicted elevation indices. For an elevation index el_idx, its precision, which can be derived from el_alph, is used to compute the projected average elevation index $el\_avg\_idx\_p$. For the corresponding azimuth index az_idx, its precision on the horizontal circle situated at the q_el elevation, which can be derived from az_alph, is used to compute the projected average azimuth index $az\_avg\_idx\_p$.

The projection to obtain predicted elevation is computed as

$$el\_avg\_idx\_p = round\left(\frac{el\_avg\_idx}{el\_avg\_alph-1} \cdot (el\_alph - 1)\right).$$

To facilitate bit-exact operation, the previous formula can be rewritten using integer only math, including division, as

$$el\_avg\_idx\_p = (2 \cdot el\_avg\_idx \cdot (el\_alph - 1) + (el\_avg\_alph - 1))\ div\ (2 \cdot (el\_avg\_alph - 1)).$$

The signed distance el_idx_dist is computed as the difference between each elevation index l_idx and its corresponding $el\_avg\_idx\_p$. Additionally, because the difference produces values in the interval $\{-el\_alph + 1, \dots, el\_alph - 1\}$, they are reduced to the interval $\{-\lfloor el\_alph \div 2 \rfloor, \dots, \lfloor el\_alph \div 2 \rfloor\}$ by adding el_alph for values which are too small and subtracting el_alph for values which are too large, like in modular arithmetic. If this reduced distance relative to $el\_avg\_idx\_p$ is interpreted using wrap-around, it can produce all values from to the unsigned alphabet containing el_alph values.

The elevation part consists of three components: the average elevation index, a Golomb-Rice parameter, and the reduced signed elevation distances. The average elevation index el_avg_idx is converted to signed, so that the zero value is in the middle of the signed interval of possible values, the *ReorderGeneric* function is applied, and the result is coded using the *EncodeQuasiUniform* function. The Golomb-Rice parameter, having an alphabet size depending on the maximum of the alphabet sizes of the elevation indices, is coded using the *EncodeQuasiUniform* function. Finally, for each reduced signed elevation distance el_idx_dist, the *ReorderGeneric* function is applied to produce $el\_idx\_dist\_r$, and the result is coded using the Extended Golomb-Rice method with the parameter indicated above.

The operation is repeated by adding an offset to the average elevation index, offset selected in the following order {+1,-1,+2,-1, +3,-3…}, where the number of tested offsets is defined by search effort requested. By default, the search effort is set to 3, meaning that the best offset among 0, +1 and -1 is retained as the one engendering the lowest bit consumption for the elevation.

In case of McMASA, or when the directions are only defined in the north hemisphere for positions of positive or null elevation, the average elevation codebook is limited to an alphabet of *no_theta[diff_idx]* elements, discarding the negative elevation. The sign is then not needed to be transmitted since always positive, and the average elevation is simply coded by the *encode_quasi_uniform*() function. Moreover, during the search effort, only positive offsets are then tested.

5.2.4.5.1.4 Azimuth entropy-coding

Unlike the elevation, which has a single resolution possible for a given diffuseness index, the azimuth can be quantized on different angular resolution for the same spherical quantization, depending on the elevation plane on which it is quantized. For the entropy coding, the average azimuth is quantized using *quantize_phi()* on the equatorial plane using the maximal resolution of the given spherical quantization.

For each direction to be entropy coded, the dequantized average azimuth q_az_avg is projected using the precision of that the direction to be coded, to obtain predicted azimuth indices. For the corresponding azimuth index az_idx, its precision on the horizontal place situated at the associated q_el elevation, which can be derived from az_alph, is used to compute the projected average azimuth index $az\_avg\_idx\_p$.

The projection to obtain predicted azimuth indices is computed as

$$az\_avg\_idx\_p = round\left(\frac{q\_az\_avg}{360} \cdot az\_alph\right)\ mod\ az\_alph,$$

which can be easily simplified to

$$az\_avg\_idx\_p = round\left(\frac{az\_avg\_idx}{az\_avg\_alph} \cdot az\_alph\right)\ mod\ az\_alph.$$

To facilitate bit-exact operation, the previous formula can be rewritten using integer only math, including division, as

$$az\_avg\_idx\_p = \left((2 \cdot az\_avg\_idx \cdot az\_alph + az\_avg\_alph)\ div\ (2 \cdot az\_avg\_alph)\right)\ mod\ az\_alph.$$

At the poles, where $az\_alph = 1$, we always have $az\_idx = 0$ and set $az\_avg\_idx\_p = 0$ directly.

The signed distance az_idx_dist is computed as the difference between each azimuth index az_idx and its corresponding $az\_avg\_idx\_p$. The difference operation produces values in the interval $\{-az\_alph + 1, \dots, az\_alph - 1\}$, which are reduced to the interval $\{-az\_alph \div 2, \dots, az\_alph \div 2 - 1\}$ by adding az_alph for values which are too small and subtracting az_alph for values which are too large. When $az\_alph = 1$, the azimuth index is always $az\_idx = 0$ and nothing needs to be coded.

As for the elevation, the azimuth part also consists of three components: the average azimuth index, a Golomb-Rice parameter, and the reduced signed azimuth distances. The average azimuth index az_avg_idx is converted to signed, so that the zero value is in the middle of the signed interval of the possible values, the *ReorderGeneric* function is applied, and the result is coded using the *encode_quasi_uniform()* function. The Golomb-Rice parameter, having an alphabet size depending on the maximum of the alphabet sizes of the azimuth indices, is coded using the *encode_quasi_uniform()* function. Finally, for each reduced signed azimuth distance az_idx_dist, the *ReorderGeneric* function is applied to produce $az\_idx\_dist\_r$, and the result is coded using the Extended Golomb-Rice method with the parameter indicated above.

For example, if the best angular precision deg_req_min used is 5 degrees, then the maximum of the azimuth alphabet sizes az_alph will be $az\_alph\_max = 4 \cdot \lceil 90 \div deg\_req\_min \rceil = 72$. In this case, the Golomb-Rice parameter values (denoted as p in the description of the Golomb-Rice method below) are limited to the interval $\{0,1,2,3,4,5\}$. The optimal value of the Golomb-Rice parameter az_gr_param is chosen by efficiently computing, for each value in the interval above, the total size in bits for all the $az\_idx\_dist\_r$ values to be coded using the Extended Golomb-Rice method, and choosing the one which provides the smallest bit size.

As for the elevation, the operation is repeated by adding an offset to the average azimuth index, offset selected in the following order {+1,-1,+2,-1, +3,-3…}, where the number of tested offsets is defined by the search effort requested. By default, the search effort is set to 3, meaning that the best offset among 0, +1 and -1 is retained as the one engendering the lowest bit consumption for the azimuth.

Azimuth encoding for the McMASA metadata input format adds additional flexibility by considering the use of adaptive averaging when calculating the quantized average azimuth index. The encoding uses the originally quantized azimuth values, which were quantized based on the quantization indexes of weighted total-to-direct energy ratios per sub band as described in clause 5.2.4.4.2. The weighted average of total-to-direct energy ratio is obtained as described in equation (5.5-3). The encoding of the azimuth for the McMASA mode is performed in the same manner as presented in the first part of this section, but with the difference of how the average is calculated. In the McMASA mode an average azimuth value is calculated for each sub band, and in addition the variance of the sub band average azimuth values is calculated for the frame. Note that the variance of the sub band average azimuth values is calculated by summing (over the number of the sub bands) the square of the difference between each sub band average azimuth value and the average of all the sub band average azimuth values.

If the resulting variance value is higher than a threshold (=25) an adaptive azimuth average will be used for the azimuth encoding of the sub frames of each sub band of the frame. If the resulting variance is less than or equal to the threshold the calculated azimuth average will be used for the azimuth encoding of the sub frames of each sub band of the frame, similar to what has been presented in the first part of this section. The decision of using the adaptive average is signalled with 1 bit to the decoder.

The encoding of the azimuth values using the adaptive azimuth average is presented below. On a frame basis, the input to the encoding routine comprises; one energy ratio value per sub band for all sub bands of the frame, *nblocks* number of azimuth, and azimuth index values per sub band, that is one azimuth value and one azimuth index value for each TF tile in the frame bands

1. The quantised average azimuth index is calculated for the first sub band, avg_idx0
2. For each quantised average azimuth index value avg_idx in {avg_idx0-1, avg_idx0, avg_idx0+1}
    a. For each subframe of the first sub band
        i. Calculate a difference azimuth index, with respect to avg_idx (i.e. the difference between the quantised azimuth index for the respective subframe and sub band and the avg_idx)
        ii. Reorder the difference azimuth index using the function *ReorderGeneric* to make it positive

b. End for

c. For sub band = 2:*nbands*

i. Calculate for first subframe the difference azimuth index with respect to avg_idx

ii. Set average azimuth, avg_azi to the azimuth value of first sub frame

iii. Update the average azimuth, avq_azi, using method 1. Then quantize the updated avg_azi to give the updated quantized average azimuth index avg_idx (which becomes the current avg_idx with respect to the next subframe).

iv. For subframe i= 2:*nblocks*

1. Calculate difference azimuth index with respect to current avg_idx

2. Reorder difference azimuth index using the function *ReorderGeneric* to make it positive

3. Update avg_azi using method 2

4. Then quantize avg_azi to obtain the updated quantised average azimuth index (which becomes the current avg_idx with respect to the next subframe)

v. End for

d. End for

e. Estimate number of bits used to encode with Golomb Rice the azimuth difference indexes for all TF tiles

3. End for

4. Use the quantised average azimuth index value giving the smallest number of bits and encode it with Golomb Rice code

5. Encode corresponding difference azimuth indexes.

The method 1 of average azimuth update from step 2.c.iii is a weighted average update using a weight, *w*, ($w = 0.5$) for the average azimuth value of the corresponding sub band from the previous frame and a weight 1-*w* for the azimuth value of the first sub frame. The method 2 for average azimuth update from step 2.c.iv.3 is without any weights and only for the values within a sub band.

Azimuth encoding for the MASA metadata input format also incorporates additional flexibility into the scheme for the case when the number of bits used by EC1 at the original quantization resolution is larger than the number of bits allowed. This additional flexibility allows for an additional small reduction in bitrate in order to limit the loss in azimuth quantization resolution. This has the effect of avoiding a too severe reduction in the quantization resolution of the azimuth angle.

In order to avoid a severe reduction in quantization resolution, a first quantization resolution can be used for the azimuth values, with the result that some of the azimuth values [of the TF tiles] are quantized at a suboptimal resolution. However, suboptimally quantized azimuth values can be tolerated under the following conditions: there is more than one sub band in the frame; the difference between the number of bits obtained with the original EC1 encoding scheme and the number of allowed bits is less than the number of TF tiles, where the number of TF tiles is given as *nbands*nblocks*. Also on an individual TF tile basis a further condition is also considered where it is checked whether a number of bits can be gained when the size of the azimuth alphabet for the individual TF tile is larger than 40. If all these conditions are met, the azimuth value of each TF tile for which the additional condition is met is compared against the average azimuth value and the difference is assigned a suboptimal quantized value by reducing Golomb Rice index. The number of bits that can be saved compared to the original quantization resolution are calculated and the operation is repeated for all individual TF tiles that meet the above further condition providing there remains individual TF tiles that meet the above further condition or as long as there is a need to reduce the number of bits used for encoding. The need to reduce the number of bits used for encoding is indicated by comparing the resulting necessary bits for encoding the directional metadata to the total number of bits allowed for encoding the directional metadata.

Under these conditions, the total number of bits that would be saved by using suboptimal azimuth values for the eligible TF tiles are calculated. Smaller differences with respect to the average index will be encoded corresponding to encoding smaller indexes differences. The bit savings are made possible by the use of Golomb Rice encoding which generally uses smaller codewords for the encoding of smaller values. If the number of bits saved is enough to reach the number of bits that are available, then this EC1 method is used, and the corresponding indexes are encoded. However, if the savings in terms of the number of bits is not enough then the EC1 method is considered to be too expensive and will not be used.

5.2.4.5.1.5 Decision for fully raw coding or entropy-coding

Once the optimal code is found for the entropy coded elevation and azimuth, its bit consumption summed the bits used for coded the raw coding are compared to the full raw coding bit consumption. The method using the less bits is retained and signal by the bit flags in table 5.2-38:

Table 5.2-38: Signalling bits for EC1

EC1 signaling 2-bit field	EC1 disable flag	Full raw coding disable flag
Raw coding	0	0
Frame-wise entropy coding	0	1

5.2.4.5.2 Entropy coding 2 (EC2)

The entropy coding method 2 (EC2) of the direction parameters azimuth and elevation is performed subband-wise. It is tried out when the number of bits estimated for the use of the EC1 method on the default quantization resolution is too high with respect to the available number of bits for the directional parameters encoding. The available number of bits is calculated as the difference between the total available number of bits for the encoding of the metadata at the considered operation point for the current direction and the number of bits already used for signalling and for the already encoded metadata parameters. In addition the use of the EC2 method is considered only when the number of bits determined to be available for the encoding of one direction parameters are less than a threshold set at 900 bits. There are *nbands* subbands and *nblocks* subframes. Similarly to the EC1 method, the direction parameters of each time frequency tile are allocated a number of bits dependent on the value of the energy ratios, more precisely on their quantization index. There are 8 possible indexes and the number of bits allocated for the direction parameters corresponding to the subband i and subframe j, $bits\_dir[b][m]$, *b=1:nbands*, *m = 1:nblocks*, are shown in table 5.2-32. The central idea of EC2 is to slightly modify the quantization resolution to a value less than the initial bit allocation for the TF tiles of some of the subbands.

The encoding procedure is illustrated by the following sequence:

1. For each subband b=1:nbands
   a. Estimate the number of bits corresponding to raw encoding method from clause 5.2.4.3.3 for the sub band directional parameters of sub band b, $bits\_band[b] = bits\_raw[b]$
   b. Estimate $bits\_band[b] = bits\_ec[b]$ the number of bits for band-wise entropy encoding of the azimuth and elevation values for the time-frequency tiles of only the sub band *b* according to the method described in clause 5.2.4.5.1.
   c. If $bits\_raw[b] < bits\_ec[b]$
   i. Select raw encoding $bits\_band[b] = bits\_raw[b]$;
   d. Else
   i. Select entropy encoding $bits\_band[b] = bits\_ec[b]$;
   e. End if
2. End for
3. Calculate $bits\_EC2 = \sum_{b=1}^{nbands} bits\_band[b]$
4. If $bits\_EC2 > bits\_allowed \wedge nblocks > 1 \wedge audio\ directions\ are\ in\ 3D$

a. Gradually change the quantization resolution to a second quantization resolution by reducing 1 bit at a time from the bit allocation for each TF-tile for the sub bands where raw encoding was more efficient. The reduction is performed until the resulting number of bits equal the bits allowed ($bits\_EC2 == bits\_allowed$) or it is no longer allowed to reduce because the minimum number of bits per TF tile is reached

b. If $bits\_EC2 == bits\_allowed$ and minimum number of bits per TF tile was not passed

i. Re-quantize the azimuth and elevation values for the TF tiles where the bit allocation has been modified

ii. Method EC2 is used

c. Else

i. Other encoding method is used (see clause 5.2.4.5.3)

d. End if

5. End if

Note that the gradual bit reduction in 4.a cannot go beyond a upper threshold obtained as the sum of $\sum_{b_{raw}} \text{bits\_band}[\text{b}_{\text{raw}}] - \text{M} * \text{min\_bits})$, where b_{raw} is the index of a raw encoded sub band and min_bits is the minimum number of bits allowed for one TF tile.

If the quantized audio direction of at least one TF tile in a frame is in the 3-dimensional space, then the audio directions are in 3D. The quantization for assessing 3D status is performed with the original azimuth and elevation values.

The variable $bits\_allowed$ that the resulting number of bits of EC2 has been compared against represents the number of bits remaining for the encoding of the directional parameters for the considered frame and direction after general signalling and non-direction dependent parameters encoding.

The EC2 method is signalled by two bits: first one to indicate that EC1 is not used (first bit is 1) and second bit (second bit is 0) to indicate that EC2 is used and not EC3. In the EC2 method, one dedicated bit will signal, for each sub band, if raw encoding (bit is 0) is used or not (bit is 1). After the raw encoding/entropy encoding signalling bits the entropy encoded indices associated with the directional metadata parameters of the entropy encoded subbands that used the original quantization resolution are written. They are followed by the fixed rate indices associated with the directional metadata parameters of the raw encoded subbands that used the slightly reduced, second quantization resolution for spatial audio signal directional metadata parameters.

5.2.4.5.3 Entropy coding 3 (EC3)

5.2.4.5.3.1 Encoding flow of EC3

The EC3 method is intended to be used for the cases when the quantization resolution for the direction parameters needs to be reduced for all, or most of the TF tiles, because it is too high for the number of bits allowed for the direction parameters encoding at the considered operation point. It is a switched fixed rate/variable rate coding. The number of bits that need to be saved depends on whether the IVAS codec input format is SBA or MASA and these values, as well as the bit reduction method are explained in the following.

The original bit allocation for the direction parameters encoding is done based on the weighted average of the energy ratios across each sub band. This average value is assigned to represent the energy ratio value of the TF tiles in the considered sub band and it is non-uniformly scalar quantized on 3 bits as presented in subclause 5.2.4.4.2. The initial estimate of bit allocation for the direction parameters on a per sub band basis is given by table 5.2-32, where the bit allocation is distributed on a sub band by sub band basis according to the quantization index of the weighted average of the energy ratios for the specific sub band.

The reduction of bits is performed as follows: for each sub band, and for each TF tile of the sub band, the number of bits for the quantization of the direction parameters (elevation and azimuth) are reduced one at a time for as long as there is a need to reduce, while making sure that the remaining bits per tile is not lower than the minimum allowed value.

After the bit reduction which ensures that the sum of the allocated bits $\sum_{b=1}^{nbands} \sum_{j=1}^{Mnblocks} bits\_dir\_red[b][j]$ equals the number of available bits left after the encoding of at least the energy ratios, the encoding of directional parameters is performed as follows:

1. For each sub band *b = 1:nbands-1*
   a. Select the current sub band index according to a predefined order $o(b)$
   b. Quantize the directional parameters and obtain the quantized elevation and azimuth values as well as the quantized elevation and azimuth indexes and the corresponding spherical index for each TF tile of the current sub band
   c. Calculate the allowed bits for current sub band $bits\_allowed(o(b)) = \sum_{j=1}^{M} bits\_dir\_red[o(b)][j]$
   d. Encode the direction parameter indexes by using fixed rate encoding with the reduced allocated number of bits, $bits\_fixed(o(b)) = bits\_allowed(o(b))$ and using a variable rate coding, $bits\_ec(o(b))$; select the one using less bits and use one bit to indicate the method: $nb = \min(bits\_fixed(o(b)), bits\_ec(o(b))) + 1$
   e. If there are bits available with respect to the allowed bits: (if bits_allowed > nb)
   i. Redistribute the difference, $dif = allowed\_bits - nb$, to the following subbands, by increasing gradually the values of $bits\_dir\_red[o(k)][j], k = b + 1: B, j = 1: M$, until all dif bits are used
   f. Else
   i. Subtract one bit from the first TF tile of the next sub band $bits\_dir\_red[o(b+1)][0] := bits\_dir\_red[o(b+1)][0] - 1$
   g. End if
2. End for
3. Encode the direction parameter indexes for the last sub band with the fixed rate approach using $\sum_{j=1}^{M} bits\_dir\_red[o(B)][j]$ bits

The number of bits that should be reduced, *reduce_bits*, are given by the following pseudo-algorithm and depending on the codec format.

```
if ( hQMetaData->is_masa_ivas_format == 0 ) /* SBA format */
{
    reduce_bits = bits_dir_raw - ( total_bits_1dir - bits_diff[d] - bits_coherence[d] -
    bits_signaling[d] );
    ind_order[0] = -1; /* signals default subband order */
}
else /* MASA or MASA related format */
{
    ind_order[0] = 0;
    reduce_bits = min( nbands * nblocks + MASA_BIT_REDUCT_PARAM, bits_dir_raw - ( total_bits_1dir -
    bits_diff[d] - bits_coherence[d] - bits_signaling[d] ) );

    if ( reduce_bits > bits_dir_raw - nbands * nblocks )
    {
         reduce_bits = bits_dir_raw - nbands * nblocks;
    }
}
```

When there is a MASA related input format, not more than 1 bit per TF tile plus a buffer of MASA_BIT_REDUCT_PARAM = 10 additional bits are to be reduced when obtaining the reduced bit allocation for the directional parameters. The *bits_dir_raw* is the total number of bits allocated in the original bit allocation for the directional parameters encoding.

For the MASA input format, a predefined order of sub bands $o(b)$ is determined using the initial bit allocation, $bits\_dir$ for the directional parameters for each TF tile of each subband and the bit allocation obtained after the bit reduction (for each TF tile of each sub band) described in the previous algorithmic flow, $bits\_dir\_red$. For each TF tile, a penalty value associated with the bit allocation (for the TF tile) is defined as the relative bit reduction with respect to the original bit allocation i.e. $(bits\_dir[b][m] - bits\_dir\_red[b][m])/bits\_dir[b][m]$, b = 1: nbands, m =

1: nblocks . A penalty value is then obtained for each subband in turn as the average of the above penalty values over the subframes (TF tiles) of the subband. The predefined order $o(b)$ of subbands for the encoding of the sub bands is then determined as starting from the unencoded sub band with the lowest penalty with subsequent unencoded sub bands ordered in monotonically increasing order of penalty values.

For the SBA input format, the order of sub bands is the default order of increasing sub band indexes.

The fixed rate encoding methods are using sending the spherical index covering both the quantized elevation and azimuth values. The variable rate coding methods used for the encoding of the elevation index and of the azimuth index are presented next.

5.2.4.5.3.2 Variable rate encoding of direction parameters within EC3

For sake of simplicity, we will use in the following the index b to denote the subband that is currently being encoded, instead of $o(b)$. If the number of subframes is larger than 1 the quantization is performed as a switched method deciding whether to use joint entropy encoding of the azimuth index and the elevation index or whether to consider sending an average common direction per subband as reference, followed by the corresponding azimuth differences, the elevation values being set to the value in the average common direction. Within the switched method, the decision to use the average common direction is enabled by the calculation, for each sub band b, of 2 distortion measures d_1 and d_2. The first distortion measure represents the estimation of the L2 norm distance on a surface of a sphere between the point on the sphere given by the elevation and azimuth and the point on the sphere given by the quantized elevation and quantized azimuth according to the joint quantization scheme.

$$d_1(b) = \sum_{j=1}^{nblocks} 1 - \cos\hat{\theta}_{bj}\cos\theta_{bj}\cos\left(\Delta\phi_{bj}(\hat{\theta}_{bj}, n_{\phi_{\hat{\theta}}})\right) - \sin\theta_{bj}\sin\hat{\theta}_{bj}$$

where $nblocks$ are the number of TF tiles in a sub band, and $\hat{\theta}_{bj}$ is the quantised elevation.

The quantized value of $\hat{\theta}_{bj}$ (elevation) is obtained, keeping in mind that for at most 4 bits for directional information, there are three possible values for the elevation 0, and +/-45 degrees. The actual number of quantized values for the elevation are obtained from the number of bits available for the considered TF tile. The parameter $n_{\phi_{\hat{\theta}}}$ holds the number of azimuth values on the horizontal circle corresponding to $\hat{\theta}_{bj}$ (see table 5.2-31). The value for $\cos\left(\Delta\phi_{bj}(\hat{\theta}_{bj}, n_{\phi_{\hat{\theta}}})\right)$ is approximated as the maximum distortion in the azimuth domain, given the fact that the quantized value of the elevation, $\hat{\theta}_{bj}$, is known as well as the number of bits for the current TF block. If there are $n_{\phi_{\hat{\theta}}}$ azimuth values for the current $\hat{\theta}_{bj}$, the maximum or worst value of the distortion in azimuth domain, $\Delta\phi_{bj}(\hat{\theta}_{bj}, n_{\phi_{\hat{\theta}}})$, is $180/n_{\phi_{\hat{\theta}}}$ (half of the distance between two consecutive points).

The distortion corresponding to the method using the common direction is given by:

$$d_2(b) = \sum_{j=1}^{M}\left(1 - \cos\theta_{av}\cos\theta_{bj}\cos(\Delta\phi_{CB}(bits\_allowed(b) - N_{av} - 1)) - \sin\theta_{bj}\sin\theta_{av}\right)$$

θ_{av} is the average value of the elevation values for the subbframes of the current subband, N_{av} is the number of bits that would be used to quantize the average direction (average elevation and average azimuth, jointly). $\Delta\phi_{CB}(bits\_allowed(b) - N_{av})$ are the values from table 5.2-39, practically obtained for the trained VQ codebooks for the difference azimuth values, for the corresponding number of bits, $bits\_allowed(b) - N_{av} - 1$ (total number of bits for the current subband minus bits for average direction, minus 1 bit to signal between joint and vector quantization.

Table 5.2-39: Azimuth codebook performance estimations

$bits\_allowed(b) - N_{av} - 1$	0	1	2	3	4	5	6
$\cos(\Delta\phi_{CB})$	0.848	0.8988	0.9563	0.9744	0.9816	0.9877	0.9925

After calculating the two distortions, d_2 is penalized if the variance of the elevation of the TF tiles of the considered subband is higher than a threshold.

```
if variance_{j=1:M}(θ_bj) > 130
 d_2 = d_1 + 1
endif
```

The decision between the switched encoding method and the joint encoding is summarized by the following pseudo-code:

```
if M > 1
    if ((1 < max(bits_dir_red[b][j]) ∧ max(bits_dir_red[b][j])<= 3 ∧  d2 <= d1  ) ∨
  (max(bits_dir_red[b][j]) <= 1))
        use_common_direction(θ, idθ, ϕ, idϕ);
   else
       joint_encoding(θ, idθ, ϕ, idϕ);
    endif
else
    joint_encoding(θ, idθ, ϕ, idϕ);
endif
```

As mentioned in clause 5.2.4.5.3.1, this decision is made for all sub bands but the last one, within the corresponding order. The initial calculation of the distortion measures is only an estimation and not the complete calculation of the actual distortion of the two considered methods.

The main functionality of the `use_common_direction()` function, corresponding to the distortion measure d_2, is described in the following:

```
if bits_allowed(b) == 0
    quantize all azimuth and elevation values of the subband to value 0
else if bits_allowed(b) <= M+1
    quantize elevation values for the subband to value 0
    quantize and encode all azimuth values of the subband with truncGR() function
else
    select the quantized average elevation value as the value closest to all elevation values by
   trying the 5 available values for the average elevation (0, 36, -36, 90, -90) corresponding to
   the indexes 0,1,2,3,4
    if idθav == 0
        signal the index idθav with the code “0”
        quantize and encode all azimuth values of the subband with truncGR() function
    else if idθav == 3 ∨ idθav == 4
        signal the index idθav with the code “1110” or “1111” respectively
    else if idθav == 1 ∨ idθav == 2
        signal the index idθav with the code “10” or “110” respectively
        quantize and encode all azimuth values of the subband with truncGR() function
    endif
endif
```

When the common direction encoding approach is used, the average elevation over the subframes of a sub band is quantized on a variable number of bits. The number of bits depends on the total number of allowed bits for the direction quantization of the sub band. The average azimuth is quantized to zero. Within the variable bit allocation for the quantization of the average elevation, for the degenerate cases when there are very few bits, the average elevation for all TF tiles of the sub band is quantized to zero, and all TF tiles of the sub band are assigned the elevation average value, which is zero.

The function *truncGR()* performs the encoding of the difference azimuth values with respect to the average value and has different functionality depending on the input audio format. Its functionality is described in the following sections for both considered input formats.

5.2.4.5.3.3 Low bitrate azimuth encoding for MASA format

When the *truncGR()* function is called there are very few bits left available for encoding. The first considered case is when the number of bits is less or equal to the number of azimuth values to encode plus 1 (i.e. M+1). If this is the case, the available bits are indicating whether the azimuth is pointing to the front or to the back using 1 bit per azimuth value, as long as there are bits.

If more than M+1 bits are available, several different quantization resolutions are considered. They are arranged such that the first higher quantization resolution comprises a set of quantization values for the 3 bit resolution, and each preceding quantization resolution comprises a set of progressively fewer quantization values.

The set of azimuth values for the 3-bit codebook is: {-180, -135, -90, -45, 0, 45, 90, 135}. The values of this 3-bit codebook have the indexes { 1, 7, 3, 5, 0, 4, 2, 6} respectively. The 2-bit azimuth codebook is a subset of the 3 bit codebook and is formed of the values corresponding to the indexes: {0, 1, 2, 3}, i.e. {-180, -90, 0, 90}. The 1-bit codebook includes only the front and back orientation corresponding to the index values {0, 1}, i.e. {-180, 0}.

The quantization of the azimuth values of sub band b, ϕ_{bj}, j=1:M is first using the maximum resolution codebook, i.e. the 3 bit codebook. If the number of bits that would be used to encode the resulting indexes with a Golomb Rice encoder of order zero is above the allowed number of bits, the lower quantization resolution codebook will be used for the quantization of the azimuth values. For this reason, some or all azimuth values should be requantized in the lower resolution codebook. The order used for making the requantization is obtained from the increasing order of the angular quantization distortion:

$$d = cos\ \hat{\theta}_{av}\ cos\ \theta_{bj}\ cos\left(\phi_{bj} - \hat{\phi}_{bj}\right)\ - sin\ \theta_{bj}\ sin\ \hat{\theta}_{av}$$

where $\hat{\theta}_{av}$ is the average elevation obtained in clause 5.2.4.5.3.2. The actual re-quantization is performed only if the original codeword belongs to the higher resolution codebook but not to the lower resolution codebook. The number of bits for encoding the indexes is iteratively updated after each re-quantization, in the above determined order. If, after considering the azimuth values of all sub bands, the resulting number of bits for encoding is still higher than the available number of bits, the codebook resolution is reduced a second time and the re-quantization procedure is redone. No further iterations will be needed because it has been originally ensured that the number of allowed bits las larger than M+1.

In a pseudo-code form, the algorithm goes as follows:

```
1. NB = 3
2. Quantize to φ̂_bj, j = 1:M the azimuth values φ_bj, j = 1:M in the NB-bits codebook
3. Calculate the number of bits, nbits, needed to encode the resulting indexes for all subbands,
with a Golomb Rice encoder of order zero
4. If nbits > allowed_bits
    Calculate the angular quantization distortion, d, for each of the quantized values taking into
account the quantized elevation value
    Sort the TF tiles in increasing order of their angular quantization distortion
    For each TF tile in increasing order of the distortion
        If the quantized azimuth belongs only to the NB-bit codebook, but not to the (NB-1)-bits
        codebook, re-quantize it in the (NB-1)-bit codebook
        Recalculate the number of bits for Golomb Rice encoding for the TF tiles
        If nbits <= allowed_bits
            Break
        End if
    End for
    If nbits > allowed bits
        NB = NB-1;
        Go to 4.a
    End if
End if
```

The Golomb Rice encoding has been described in clause 5.2.4.3.4.2.

5.2.4.5.3.4 Low bitrate azimuth encoding for McMASA format

If the input audio format is multi-channel, for the low and mid-range bitrates, as defined in clauses 5.7.1 and 5.7.3, the internal representation is McMASA and the corresponding metadata is encoded following the main procedures of MASA metadata encoding, with several adjustments that consider the original input format being multi-channel audio format. For this case, the highest resolution codebook for the azimuth is formed of the following codewords {-135, -110, -30, 0, 30, 110, 135}. The lower resolution codebooks are {-110, -30, 0, 30, 110}, {-30, 0, 30}, and {0}.

The encoding is performed first in the highest resolution codebook, the number of bits needed for encoding are estimated, as well as the quantization distortion for each input azimuth value. If the number of bits needed is larger than the number of available bits, each azimuth value is tested to be quantized in a lower resolution codebook, which in practice, for the selected codebook structure, corresponds to quantizing to a value closer to the value zero. The actual azimuth input selected to use the lower resolution codebook is selected as the one that affects the least the overall quantization distortion. After selection, the number of bits is again estimated, and they are further similarly reduced by re-quantization until the bit budget is attained.

5.2.4.6 Coherence coding

5.2.4.6.1 Spread coherence coding

In addition to the direction parameters, azimuth and elevation, the spatial metadata also includes the coherence parameters on a TF tile basis. When present, there are two types of coherence parameters: a spread coherence for each direction in a TF tile, when there is more than one direction present, otherwise there will be a single spread coherence

value associated with it the TF tile; and a single surround coherence specified for the TF tile which is irrespective of the number of directions.

This section presents the encoding of the spread coherence, and the encoding of the surround coherence is presented in the following section. Both spread coherence and surround coherence values are represented as 8-bit floating point numbers between 0 and 1. Two main cases are considered: when the number of subframes is 4 and when the number of subframes is 1. We present first the case when the number of subframes is 4.

In summary, the encoding of the spread coherence values is performed on a sub band by sub band basis for the spread coherence values associated with the TF tiles of the sub band. For each sub band, the spread coherence values of the sub band are formed into a vector (with each element of the vector being a spread coherence value associated with the sub frame of the sub band in question.) The vector is then transformed using a low complexity 4-dimension DCT transform, where only the first two components of the transformed vector are retained for further processing, the remaining components of the DCT transformed vector are set to zero. The retained DCT components are then quantized using determined codebooks, where each codebook is determined specific to the component of the retained component, and each determined codebook is selected based on an energy ratio value and the variance of the azimuth values of the considered sub band.

The low complexity 4-dimensional DCT transform and inverse transform are implemented as follows.

For a spread coherence vector $x = [x_1, x_2, x_3, x_4]'$ the matrix multiplication with the DCT matrix of order 4 is equivalent to:

$$y := DCT(x) = \begin{bmatrix} 0.5\,(a+b) \\ 0.653281482438188\,c + 0.270598050073099\,d \\ 0.5(a-b) \\ 0.270598050073099\,c - 0.653281482438188\,d \end{bmatrix}$$

where $a = \frac{(x_1+x_2)}{256}$, $b = \frac{(x_2+x_3)}{256}$, $c = \frac{(x_1-x_4)}{256}$, $d = \frac{(x_2-x_3)}{256}$.

For the inverse DCT transform, the multiplication with the inverse DCT matrix of order 4 is realized by:

$$DCT^{-1}(y) = 128 * \begin{bmatrix} a' + c' \\ b' + d' \\ b' - d' \\ a' - c' \end{bmatrix}$$

Where $a' = y_1 + y_3$, $b' = y_1 - y_3$, $c' = 1.306562964876376\,y_2 + 0.541196100146198\,y_4$, $d' = 0.541196100146198\,y_2 - 1.306562964876376 y_4$.

As mentioned above, after the DCT transform, only the first and the second DCT components of the transformed vector of the spread coherence values vector (for the sub band) are quantized. The first and second DCT components are each quantized with a separate codebook.

The codebook determined for the zero order DCT coefficient (first component of the vector) depends on the quantization index of the weighted average of the energy ratio over the subframes of the sub band and on the azimuth variance of the sub band. The codebook is defined according to the following table 5.2-40.

Table 5.2-40: Codebook for DCT coefficient of order zero of the spread coherence

Azimuth variance	index	Codebook	Codebook length (no_cv)
< Thr	0	0.0478, 0.2547, 0.5515, 0.9865, 1.3573, 1.6838, 1.9674	7
	1	0.1211, 0.4123, 0.6997, 1.0154, 1.4197, 1.7971	6
	2	0.1614, 0.5674, 0.9708, 1.3699, 1.7357	5
	3	0.2349, 0.7306, 1.2051, 1.6547,	4
	4	0.2381, 0.7185, 1.1758, 1.6335	5
	5	0.2400, 0.7027, 1.1407, 1.6243	4
	6	0.3518, 0.9398, 1.6454	3
	7	1.0	1
>= Thr	0	0.1445, 0.3432, 0.5351, 0.7524, 1.0095, 1.3210, 1.7399	7
	1	0.2002, 0.4447, 0.6869, 0.9593, 1.3090, 1.7029	6
	2	0.2617, 0.5556, 0.8645, 1.2185, 1.6051	5
	3	0.3318, 0.6810, 1.0445, 1.4710	4
	4	0.3324, 0.6588, 0.9980, 1.4164	4
	5	0.3378, 0.6658, 1.0009, 1.4269	4
	6	0.3986, 0.8244, 1.3526	3
	7	1.0	1

As mentioned above, the choice of the codebook (codeword values and number of codewords) for encoding the first component is determined by the variance of the azimuth values for the sub band and the quantization index of the weighted average value of the energy ratios values for the sub band. The weighted average of the energy ratio is obtained as described for instance in equation (5.5-3). The codebooks from the upper half of the table should be selected based on the quantized weighted average energy ratio index for the sub band if the variance of the azimuth is lower than a determined threshold selected to be 30. The codebooks from the lower half of the table are considered for selection when the azimuth variance is higher than or equal to the determined threshold.

The resulting indexes considering their corresponding maximum values according to table 5.2-40 are encoded using a product code for all sub bands. If the sum of all number of codewords for all sub bands is larger than a threshold value (96), two indexes are formed, one for the first half of the sub bands and a second one for the second half.

The DCT component of order 1 (the second component) is quantized using a codebook depending on the number of sub bands in the frame and the sub band index. The dependency on the sub band number and index is illustrated in the following table 5.2-41:

Table 5.2-41: Spread coherence codebook allocation based on sub band index

Number of sub bands	Codebook index for each sub band
5	0, 1, 2, 3, 4
8	0, 1, 1, 2, 3, 3, 4, 4
12	0, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4
18	0, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4
24	0, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4

The above table should be read such that if, for instance there are 8 sub bands, the first sub band will use the codebook of index 0, the second and third sub bands will use the codebook of index 1, the fourth sub band will use the codebook of index 2 and so on. The 5 codebooks indexed by the numbers 0 to 4 are presented in the following table 5.2-42.

Table 5.2-42: Codebooks for the first DCT component of the transformed spread coherence vector

Index	Codebook
0	-0.0019, 0.2597, -0.2696, 0.6269, -0.6295,
1	-0.0055, 0.2582, -0.2784, 0.6473, -0.6538,
2	-0.0029, 0.2506, -0.2630, 0.6498, -0.6540,
3	-0.0004, 0.2546, -0.2579, 0.6851, -0.6743,
4	0.0017, 0.2521, -0.2479, 0.7217, -0.7157

The DCT coefficients of order 1 are encoded by first calculating the average value of the quantization indexes, rounding it to an integer, encoding it with a Huffman code, and then encoding the index differences with the Golomb Rice

encoder of order 0. Before encoding, the index differences, id_{dif}, they are first transformed to positive values using the function:

$$f(id_{dif}) := \begin{cases} -2id_{dif}, if\ id_{dif} \leq 0 \\ 2id_{dif} - 1, if\ id_{dif} > 0 \end{cases} \quad (5.2\text{-}269)$$

For the case when there is only one subframe for each sub band, the above DCT transform based method no longer applies. Instead, the spread coherence value of each sub band is uniformly scalar quantized, without the DCT transform. The number of quantization levels for the scalar quantizer, *no_levels*, depends on the energy ratio of the sub band, *energy_ratio(b)* and the total number of sub bands, *nbands*.

The number of quantization levels is given by $no\_levels = 7 - energy\_ratio(b) + \frac{nbands}{6}$.

The obtained indexes, one per sub band, for the spread coherences are further encoded as follows. If the number of sub bands in a frame is larger than 8, the maximum index over the sub bands is obtained and the number of codewords, *no_cv*(*b*), for the sub bands which are larger than the maximum index are set to the maximum index plus one. The maximum index value is then encoded with the Golomb Rice code of order zero. Two coding strategies are tried out by estimating their bit consumption and then the one with lower bit consumption is selected and encoded.

The first method is a fixed rate approach using $\lceil \log_2 \prod_{b=1}^{nbands} no\_cv(b) \rceil$ bits. If $\sum_{b=1}^{nbands} no\_cv(b) > 96$, two indexes are obtained, one for the first half of the subbands and the second one for the second half. This is done to make sure the number of bits per index can be represented as an integer.

The second method determines and removes the minimum of the quantized spread coherence index across the sub bands of the frame and then encodes with a Golomb Rice code both the minimum index value and the minimum -removed index values for all the sub bands of the frame The Golomb Rice order for encoding the minimum value is 0, while for the encoding of the quantized spread coherence indexes for the frame, an optimal value between 0 and 1 is selected.

5.2.4.6.2 Surround coherence coding

For each sub band there is one weighted average surround coherence value based on the signal energy. This is obtained as described in clause 5.5.3.2.4. Also, for each sub band a significance measure of the surround coherence of the sub band is calculated, which is applicable across the TF tiles of the sub band. The measure is based on the weighted average energy ratio for the sub band *b*, of direction *i*, $energy\_ratio_i(b)$, and is calculated in accordance with clause 5.5.3.2.4. The significance measure of the surround coherence corresponds to the proportion of coherent non-directional energy per sub band, $\epsilon(b)$. It is calculated as

$$\epsilon(b) = 1 - \sum_{i=1}^{nD} energy\_ratio_i(b) \quad (5.2\text{-}270)$$

using the energy ratio values for both directions if the number of directions *nD* is 2. The significance measure value is quantized using the codebook from table 5.2-33 and the resulting quantization index is *idx_er* is used as a second significance measure that will be available also at the decoder. Based on the second significance measure the weighted average surround coherence for the current sub band is quantized with the codebook from table 5.2-43 which corresponds to the quantization index *idx_er*.

Table 5.2-43: Codebook surround coherence

idx_er	Codebook	Codebook length (no_cv)
0	16, 99	2
1	12, 64, 167	3
2	10, 45, 100, 218	4
3	8, 34, 70, 124, 235	5
4	7, 27, 55, 90, 141, 242	6
5	7, 23, 45, 71, 105, 153, 247	7
6	6, 20, 38, 60, 86, 119, 165, 249	8

Therefore, the above process results in a quantization index of a weighted average surround coherence value for each sub band. If the number of sub bands, *nbands*, is larger than 8, the maximum index over the sub bands is first obtained and the number of codewords, *no_cv*(*b*), for the sub bands which are larger than the maximum index, is set to the maximum index plus one. The maximum index value is then encoded with the Golomb Rice code of order zero. Two coding strategies are tried out by estimating their bit consumption. The one with lower bit consumption is selected and encoded.

The first method is a fixed rate approach using $\lceil \log_2 \prod_{b=1}^{nbands} no\_cv(b) \rceil$ bits. If $\sum_{b=1}^{nbands} no\_cv(b) > 96$, two indexes are obtained, one for the first half of the subbands and the second one for the second half. This is done in order to make sure the number of bits per index can be represented as an integer.

The second method determines and removes the minimum weighted average surround coherence index across the sub bands of the frame and then encodes with a Golomb Rice code both the minimum index value, and the minimum-removed index values for all the sub bands of the frame. The Golomb Rice order for encoding the minimum value is 0, while for the encoding of the weighted average surround coherence indexes, the optimal value between 0 and 1 is selected.

5.2.4.7 DTX coding

DTX coding is common for SBA-DirAC and MASA. In the case of non-active frames and for the SID frames, it transmits a second sound field parameter representation, which is more compact than the parametric sound field representation used for active frames. The total payload allocated to the SID sound field parameters is a maximum of 2.8 kbps, corresponding to the total SID bit rate of 5.2 kbps minus the 2.4 kbps SID of core-coding.

The number of parameter bands is limited to 5 for MASA and 2 for the DirAC part of SBA, representing the average of parameter bands expect the last one and the highest parameter band. The number of direction (DoA) is also limited to only one direction per parameter band. In case of MASA, the parameters are combined to one direction, 5 bands, and 1 time block.

First the diffuseness quantized on 8 levels is mapped to 4 levels on coded in raw coding on 2 bits:

$$diff\_index[b] \leftarrow \max(4, diff\_index[b])$$

The associated spherical or azimuth coding resolution and bit demand is then computed for 3D or 2D audio scene, resolution. If the bit demand is greater than the bit allocated for the spatial parameter of the SID frame, then two different mechanisms are used, one for MASA and another for SBA.

In MASA and if the bit demand is too low compared to the bit-budget for the SID spatial parameters, the spherical or azimuthal resolution is modified greedily, from the lowest to the highest parameter band, in increments of +1, in order to meet the bit budget. If the bit demand is too high, the resolution is also modified greedily, but this time from the highest to the lowest band, and by a decrement of -1. The same logic has then to be applied at the decoder side.

In DirAC, the quantization resolutions are only changed if the bit demand is too high. It is achieved by increasing the diffuseness index by 1, from the highest to the lowest parameter bands. This is repeated until the bit budget is fulfilled. Since the diffuseness indices are transmitted, no extra logic must be applied at the decoder side.

The diffuseness indices for each parameter bands are then coded using the function *encode_quasi_uniform(),* for an alphabet of 4. The directions are averaged by averaging their cartesian coordinates through the time blocks for each parameter bands. The average directions are then transformed back to polar coordinates by quantizing on the azimuth in case of 2D scenes or the azimuth and elevation jointly by using the spherical quantization and indexing in case of 3D scenes. The spherical indices are coded in raw coding while the azimuth indices are coded using the *encoder_quasi_uniform(),* with the appropriate alphabet.

5.2.5 Modified Discrete Fourier Transform (MDFT) Analysis Filter Bank

5.2.5.1 General

The MDFT analysis filter bank is a 12-band convolution-type filter bank with a 1 ms signal delay. It is implemented by making use of 12 pre-computed bandpass filters, each filter associated with a centre frequency, with the overall filter impulse response $h[n]$ formed from the weighted sum of the band-pass filter impulse responses $h_b[n]$ such that

$$h[n] = \sum_{b=1}^{B} h_b[n] w_b \quad (5.2\text{-}271)$$

where the weights w_b are dynamic across blocks and are used to alter the gain of the audio signal at each frequency band. The filters are applied in the MDFT domain (subclause 5.2.5.2) at a 20 ms block size with 1 ms crossfade between blocks. The filter bank is in the audio signal path for SBA coding (subclause 5.4) and is used to generate an active downmix which is signal-dependent and time-varying. The MDFT-domain filter bank output $Y[k]$ for input $X[k]$ is given by

$$Y[k] = \sum_{b=1}^{B} X[k]H_b[k]w_b. \quad (5.2\text{-}272)$$

5.2.5.2 Modified Discrete Fourier Transform (MDFT)

The N-point Modified Discrete Fourier Transform ($MDFT_N$) is the DFT of input signal $x[n]$ modulated by a complex exponential $e^{\frac{-j\pi n}{N}}$.

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} e^{\frac{-j2\pi n(k+\frac{1}{2})}{N}} x[n]. \quad (5.2\text{-}273)$$

The modulation corresponds to a half-bin shift in the DFT spectrum, which results in no bin centered at DC.

5.2.5.3 Filter bank responses

5.2.5.3.1 General

The pre-computed filter bank magnitude responses $|H_b|$ for each of the 12 bands b for sampling rate $Fs = 48000$ Hz are shown in Figure 5.2-29 and the corresponding impulse responses $h_b[n] = MDFT_{960}^{-1}(H_b[k])$ are shown in Figure 5.2-30. It can be observed from Figure 5.2-29 that the filters are narrower at lower frequencies than at higher frequencies. Figure 5.2-30 shows that the filters are aligned to have peak energy at 1 ms. These filters sum very closely to a unit impulse with a 1 ms delay:

$$\sum_{b=1}^{12} \mathrm{h_b}[n] \cong \delta[\mathrm{n} - \mathrm{L}] \quad (5.2\text{-}274)$$

where $L = 0.001Fs$. This delay is equal to the audio latency of the filter bank. Note that $h_b[n] = 0, \forall n < 0$.

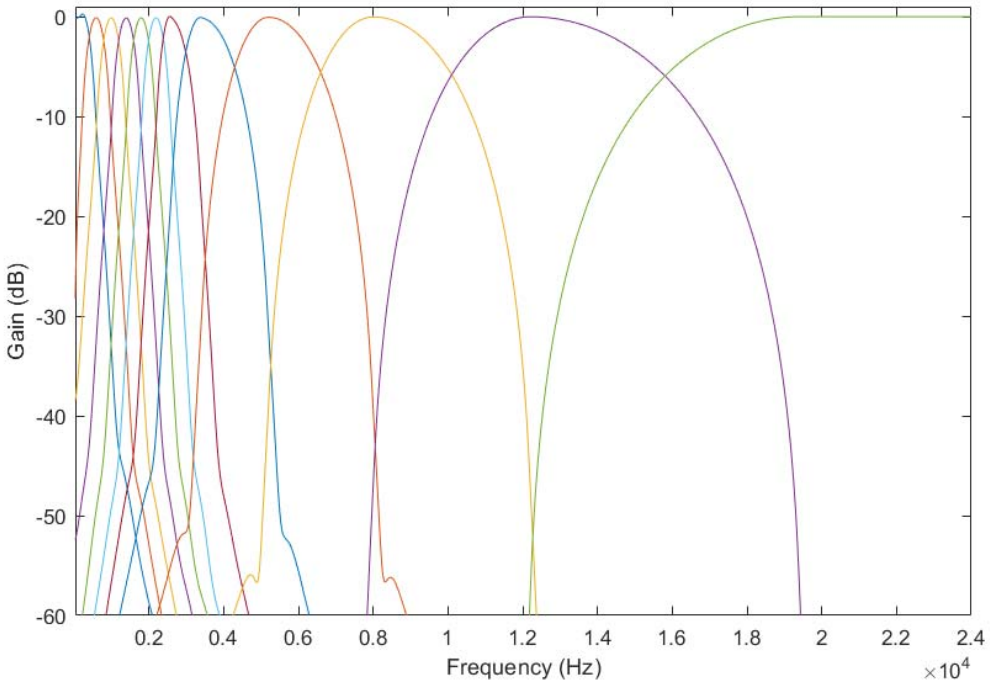


Figure 5.2-29: MDFT filterbank magnitude responses

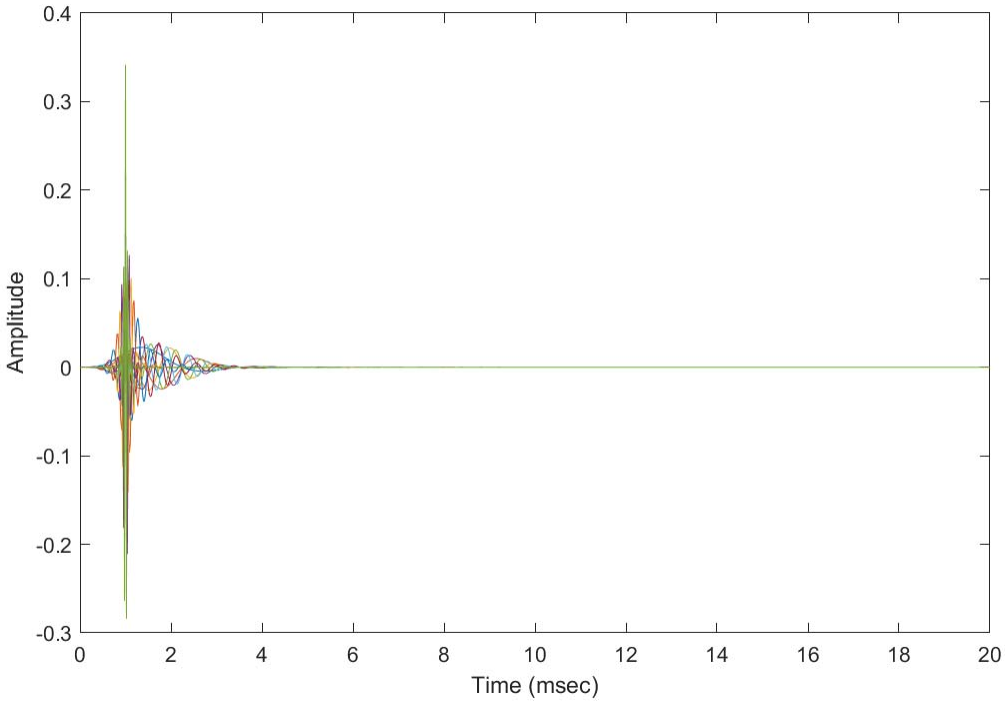


Figure 5.2-30: 48 kHz MDFT filterbank impulse responses

5.2.5.3.2 Magnitude responses for 5 ms stride operation

Note: $H_{abs,5ms}^{mdft}(b,k)$ is the short stride filterbank magnitude response referred in Eqn (5.4-20).

For SBA parameter estimation, SPAR banded signal energies and covariances are computed based on Modified Discrete Fourier Transform (MDFT) analysis running at a stride of 5 ms and a window length of 10 ms resulting in 240 complex coefficients per stride. The MDFT analysis coefficients are then weighted to emulate SPAR filtering and compute parameters associated with certain SPAR bands. The weighting is derived from the stored SPAR MDFT frequency responses where the MDFT window covers the entire SPAR filter impulse response.

First, short stride SPAR filter power responses $H'^{mdft}_{abs,5ms}(b,k)$ for SPAR band b and frequency index k are derived from the original SPAR frequency responses $H_b[k]$ as

$$H'^{mdft}_{abs,5ms}(b,k) = \sum_{i=4k}^{4k+3} |H_b(i)|^2 \, , for \text{ b } = 0,1,\dots,11 \text{ and k } = 0,1,\dots,239 \tag{5.2-275}$$

Second, short stride SPAR filter weights $H_{abs,5ms}^{mdft}(b,k)$ are derived from the short stride power responses as

$$H_{abs,5ms}^{mdft}(b,k) = max\left\{\frac{H'^{mdft}_{abs,5ms}(b,k)}{H'^{mdft}_{abs,5ms}(b,k_{max}^{b})} - 0.3, 0.0\right\}/0.7 \tag{5.2-276}$$

where $H'^{mdft}_{abs,5ms}(b,k_{max}^{b})$ represents the maximum power response for band b:

$$H'^{mdft}_{abs,5ms}(b,k_{max}^{b}) \geq H'^{mdft}_{abs,5ms}(b,k) \; for \; all \; k. \tag{5.2-277}$$

Since some of the computed weights are zero, energies and covariances with respect to a certain SPAR band can be computed efficiently by considering only those short stride MDFT coefficients where the respective SPAR weighting is greater than zero. To better facilitate this processing, helper frequency indices are used as below:

k_0^b is the smallest frequency index for SPAR band b with a weighting greater than zero, so the following is true:

$$H_{abs,5ms}^{mdft}(b,k_0^b) > 0$$

$$H_{abs,5ms}^{mdft}(b,k) = 0 \; for \; k < k_0^b \tag{5.2-278}$$

k_1^b is the largest frequency index for SPAR band b with a weighting greater than zero, so the following is true:

$$H_{abs,5ms}^{mdft}(b, k_1^b) > 0$$

$$H_{abs,5ms}^{mdft}(b, k) = 0 \; for \; k > k_1^b \tag{5.2-279}$$

5.2.5.3.3 Responses for sampling rates below 48 kHz

For operation at 32 kHz and 16 kHz sampling rates, it is desirable to maintain the properties of the original 48 kHz design whilst also maintaining the same 1 ms processing delay. To achieve this, one seeks to obtain a new set of filters that sum approximately to a unit impulse at 1 ms, maintain the low-latency property that $h_b[n] = 0, \forall n < 0$ and closely approximate the banded frequency responses of the original design below the new (lower) Nyquist frequency. The following method is used to construct appropriate filters for these sampling rates from the original 48 kHz filters.

The starting point is to truncate the original frequency response at the new Nyquist frequency, then convert to the time domain to obtain an impulse response h_b':

$$h_b'[n] = MDFT_M^{-1}(H_b'[k]) \tag{5.2-280}$$

where

$$H_b'[k] = \begin{cases} H_b[k], & k < M \\ 0, & otherwise \end{cases} \tag{5.2-281}$$

and the block size $M = 0.02Fs$ corresponds to the index of the Nyquist frequency at the new sampling rate Fs. This impulse response represents a filter that matches the original frequency response perfectly but will in general contain pre-ripple energy at $n < L$ that extends below n = 0 and would therefore require additional delay to implement. To address this, a fade function $s(n)$ is defined as

$$s[n] = \begin{cases} 1 & n < -L \\ 0 & n > L \\ f_{cheby}[n] & -L \le n \le L \end{cases} \tag{5.2-282}$$

where f_{cheby} is a pre-computed ramp function. The pre-ripple repsonse r_b^{pre} is obtained from the original impulse response as follows:

$$r_b^{pre}[n] = h'_b[n] s[n] \, h_{win}[n]. \tag{5.2-283}$$

where h_{win} is an M-point Hanning window.

The post-ripple response r_b^{post} is then obtained by time-reversing the pre-ripple response:

$$r_b^{post}[n] = - r_b^{pre}[-n], \tag{5.2-284}$$

A final window is defined as

$$h_{fwin}[n] = \begin{cases} \sin(\pi \frac{n+1}{2L}), & n < L \\ 1, & L \le x < 2L \\ \sin(\pi \frac{M-n}{2(M+1-2L)}), & 2L \le x < M \\ 0, & otherwise \end{cases} \tag{5.2-285}$$

The mofidied impulse response h_b^{ld} is then obtained by subtracting the pre-ripple response and adding the post-ripple response, as follows:

$$h_b''[n] = (h'_b[n] - r_b^{pre}[n] + r_b^{post}[n]), \tag{5.2-286}$$

$$h_b'''[n] = \begin{cases} h_b''[2M - L + n], & -M \le n < -M + L \\ h_b''[n - L], & -M + L \le n < 0 \\ -h_b''[n - L], & 0 \le n < L \\ h_b''[n - L], & L \le n < M \end{cases}, \tag{5.2-287}$$

$$h_b^{ld}[n] = (h'''_b[n]) \, h_{fwin}[n]. \tag{5.2-288}$$

As an optimisation, filters in the original 48 kHz design corresponding to bands where the energy above the Nyquist frequency is essentially zero are used as in equation (5.2-280) whilst filters for bands having nearly all their energy above the Nyquist frequency are not used for processing. For bands 8 – 10 at 16 kHz sampling rate, and bands 10 – 12 at 32 kHz sampling rate, which contain energy above and below the Nyquist frequency, the filters as given by equation (5.2-286) are used.

5.3 Stereo audio operation

5.3.1 Stereo format overview

The IVAS codec supports native stereo coding at bitrates from 13.2 kbps to 256 kbps. The stereo format encoder consists of stereo modules operating in time domain or frequency domain representing the following stereo modes:

- Time-domain (TD) stereo mode
- Discrete Fourier Transform (DFT) domain parametric stereo mode
- Modified Discrete Cosine Transform (MDCT) domain stereo mode

The use of stereo modes at different bitrates is summarized in Table 5.3-1.

Table 5.3-1: Overview of bitrates, bandwidths, and coding modes in Stereo format

Bitrate [kbps]	Maximum bandwidth	Stereo mode
13.2 – 24.4	SWB	DFT / TD stereo
32	FB	DFT / TD stereo
48 - 256	FB	MDCT stereo

At low bitrates from 13.2 kbps up to 32 kbps the IVAS stereo codec switches between TD stereo mode and DFT stereo mode in a so-called Unified stereo module (clause 5.3.2). The selection between DFT stereo and TD stereo is done in each frame based on the stereo classifier decision (clause 5.3.2.5). At higher bitrates from 48 kbps up to 256 kbps the IVAS stereo codec then operates in MDCT stereo mode (clause 5.3.3).

In addition, the IVAS codec supports a stereo downmix processing to generate a mono signal for EVS-interoperable encoding (clause 5.11).

5.3.2 Unified stereo

5.3.2.1 Overview

The unified stereo is switched framework for stereo format coding at bitrates from 13.2 kbps to 32 kbps. It consists of the TD stereo (clause 5.3.2.3) and the DFT stereo (clause 5.3.2.4) modes while the selection between them is driven by the stereo classifier (clause 5.3.2.5).

5.3.2.2 Common unified stereo tools

5.3.2.2.1 Inter-Channel Bandwidth Extension (IC-BWE)

5.3.2.2.1.1 IC-BWE System overview

After the ICA block aligns the left and right channels of the stereo signal by adjusting the target and reference signals, they are ready to be encoded at the input sampling rate, *Fs*.. The Mid-Side Converter downmixes the incoming stereo signal mid (M) and side signals (S) at the input sampling rate (F_S) as shown below:

$$M = reference(n) + target(n + N_1), \text{ and} \quad (5.3\text{-}1)$$

$$S = ref(n) - g_{MD} target(n + N_1), \quad (5.3\text{-}2)$$

where *N1* is the currentShiftValue, and g_{MD} is a downmix factor.

The (*M*) mid_{Fs} and (*S*) $side_{Fs}$ are further split into two bands, the low-band(LB) and the high-band (HB). The low band spanning from 0-8 kHz and the high band spanning frequencies above 8 kHz to support WB, SWB, and FB. For coding the mid channel, a split band BandWidthExtension (BWE) is used. The low band mid signal (mid_{LB}) in the low-band mid channel F_{s_core} is encoded using a Low-Band Encoder, and the high band mid signal (mid_{HB}) is encoded using a BandWidthExtension (BWE) technique, Time-domain Bandwidth Extension (TBE). The low band side signal, $mid_{LB,}$ in the low-band mid channel $_{Fs\_core}$ is encoded using any signal coding techniques.

Explicit waveform coding of the high band side signal is unnecessary because signal phase perception in the high band is substantially lower than for low band. For coding stereo signals, a valid assumption is that the mid_{HB} is similar to the dominant channel's HB signal (L_{HB} or R_{HB}). Thus, the Inter-Channel BWE (IC-BWE) Encoder determines a high band channel which fits the assumption that the mid_{HB} is approximately similar in energy level to that high-band reference channel signal, Ref_{HB}. Once the RefHB is identified, it is used to select the high band non-reference channel, $NonRef_{HB.}$ The Inter-Channel BWE determines (i) a gain mapping for the energy level of the $NonRef_{HB}$, based on Ref_{HB} and (ii) spectral shape mapping for the spectral shape of the $NonRef_{HB}$ based on Ref_{HB}.

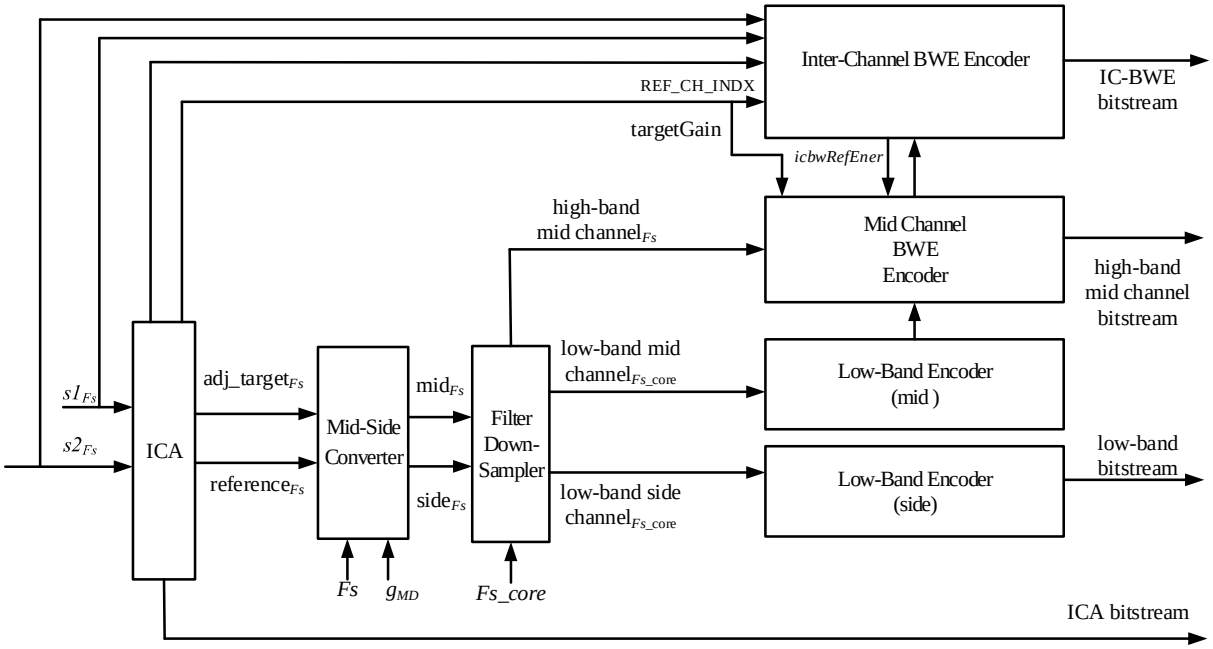


Figure 5.3-1: Block diagram of IC-BWE in relation to other blocks

5.3.2.2.1.2 Mid-Channel BWE Encoder

The Mid-Channel BWE Encoder includes the following sub-blocks: (a) Non-Harmonic Detector; (b) High Band Mixing Gains Estimator; (c) Bandwidth Extension (BWE) block; (d) Linear Predictive Coefficient (LPC) block, and (e) a High-Band Gain Estimator. The Mid-Channel BWE Encoder receives as input the high-band mid-channel signal output from the Filter-Down-sampler shown in Figure 5.3-1. The outputs of the Mid-Channel BWE Encoder are a high-band LPC bitstream that include quantized high-band LPC's, the high-band mid-channel bitstream which include the quantized gain shape and quantized gain frame of the High-Band Gain estimator, and the synthesized high-band mid-channel signal and the Mid-Channel-BWE parameters required by the IC-BWE Encoder.

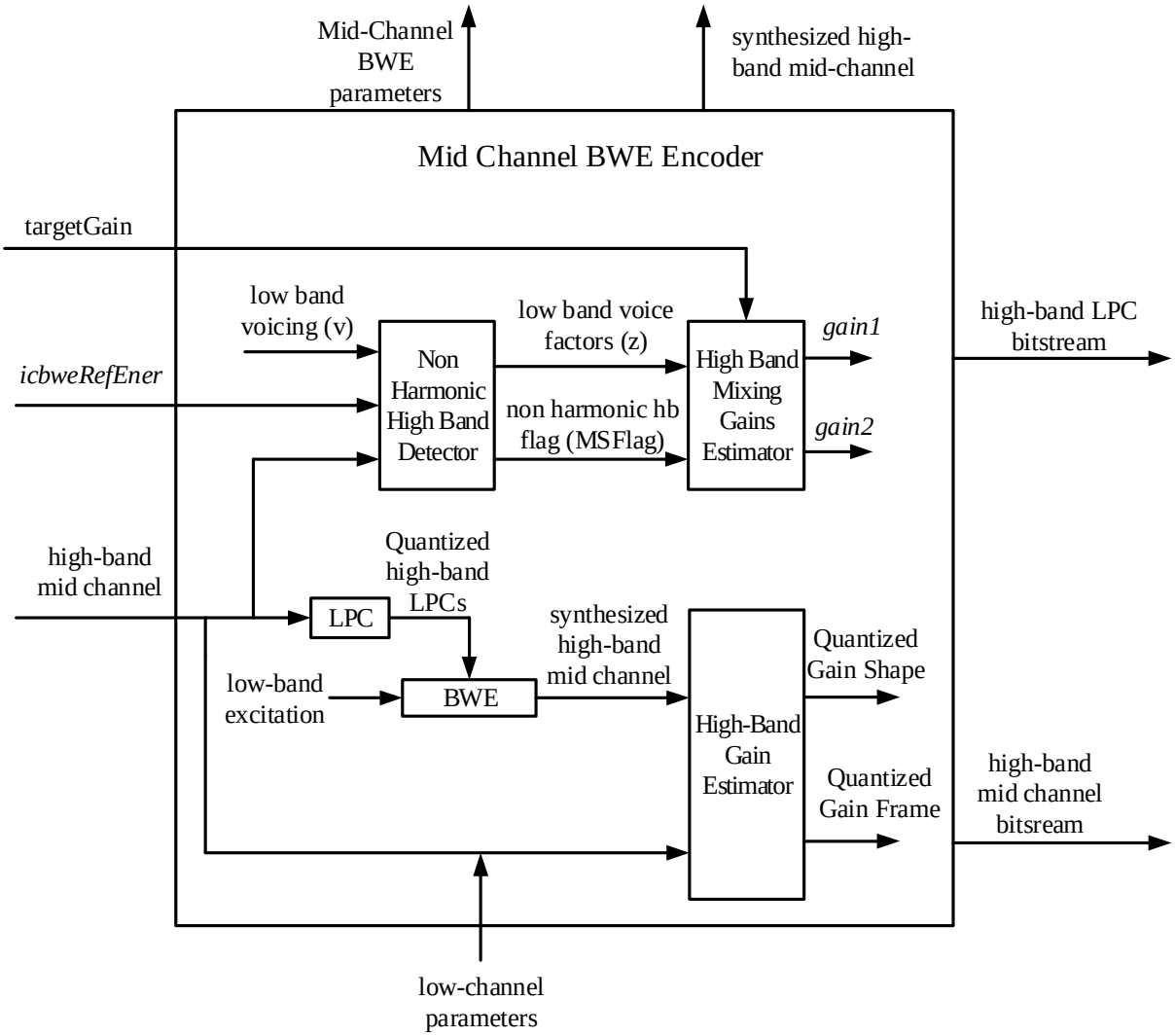


Figure 5.3-2: Block diagram of the Mid-Channel BWE Encoder

(a) Non-Harmonic High Band Detector

The Non-Harmonic High Band Detector determines a value of a non-harmonic high band flag, MSFlag, associated with the high-band mid signal, where the non-harmonic high band flag corresponds to whether the high-band mid signal is harmonic or non-harmonic. The MSFlag is based on a (i) low-band voice value (v); and (ii**)** (i) gain value. The low-band voice value corresponds to voicing factors from the low-band mid signal. The gain value corresponds to using the energy value of the high-band mid channel signal, which is predominantly attributed to the energy of the high band reference channel signal, Ref_{HB}. The energy, icbweRefEner of RefHB is generated by computing the magnitude of the spectrum (REF) of Ref_{HB} over the last (20) narrow bands of the spectrum of REF.

$$icbweRefEner = \sum_{B=20}^{39} \|REF\|, \quad (5.3\text{-}3)$$

where B are the narrow bands of the spectrum of REF. The low-band voice value, v, is the average low band voicing. The energy, icbweRefEner, and the low-band voice value, v, are used to compute the value of the non-harmonic band flag,. MSFlag, by using a three-level decision tree to calculate a regression value. The three-level decision tree is an indicator of the likelihood of non-harmonic HB content. The variable, t, based on the ICBWERefEner is used in the three-level decision tree, where t is as shown below:

$$t = log10 \frac{cbweRefEner + epsilon}{lowbandEner + epsilon}, \quad (5.3\text{-}4)$$

where epsilon $=10^{(-6)}$.

Pre-determined regression values based on the conditions satisfied are present in the regV[] array below. The pseudo code that implements the logic of the three level decision tree is shown below:

```
/* Three Level Decision Tree to calculate a regression value first */
/*regv is anarray of regression values */
if ( t < thr[0] ) /* level 1 */
{
    if ( t < thr[1] ) /* level 2 */
    {
    regression = (v < thr[3] ) ? regV[0]:regV[1] /* level 3 */
    }
    else{regression = ( v < thr[4] ) ? regV[2] : regV[3] /* level 3 */
    }
}else
        {
        if ( t < thr[2] ) /* level 2 */
        {
        regression = ( v < thr[5] ) ? regV[4] : regV[5] /* level 3 */
        }
        else
        {
        regression = ( v < thr[6] ) ? regV[6] : regV[7] /* level 3 */
        }
}
```

The regression is converted into a hard decision, i.e., classification. If [the regression < .79] and (i) [the bandwidth is SWB or higher] and (ii) the voice activity detection flag is not zero, then the MSFlag is set to 1. When the regression is high and in any operating mode of SWB, FB, and when the current frame is an active frame, the MSFlag is one, indicatig non-harmonic content.

(b) High Band Mixing Gains Estimator

The High Band Mixing Gains Estimator function is to generate two high band mixing gains (gain1 and gain2), based on the value of the MSFlag and low band voicing factors (z).

The High Band Mixing Gains Estimator performs a check on the target gain, targetGain, from the ICA block. If the targetGain is >= .5 or the targetGain is > 2, then the low band voicing factors (z) are scaled by .5, i.e.,

$$z = \frac{z}{2}. \tag{5.3-5}$$

When the non-harmonic band flag, MSFlag, is 1,

$$gain1 = 0, \tag{5.3-6}$$

and

$$\text{gain}2 = 1, \tag{5.3-7}$$

otherwise

$$gain1 = \sqrt{z} \tag{5.3-8}$$

and,

$$gain2 = \sqrt{1 - z}. \tag{5.3-9}$$

where z represents the low-band voicing factors.

(c) Bandwidth Extension (BWE) and LPC

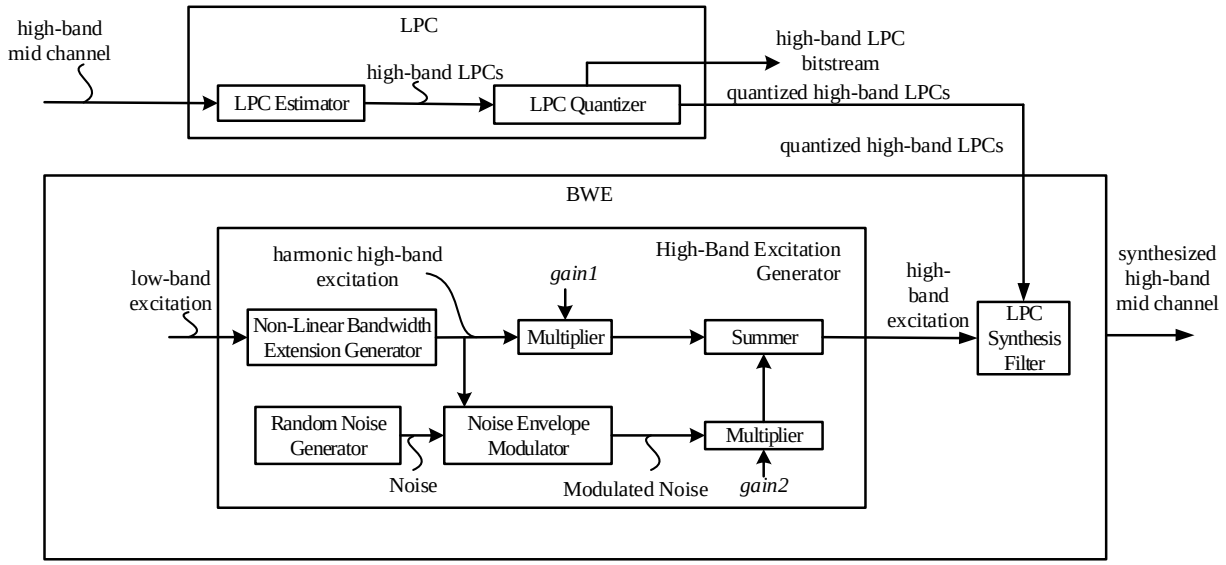


Figure 5.3-3: Block diagram of the BWE and LPC modules

The BWE module generates the synthesized high-band-mid channel signal based on a high-band excitation signal. The High Band Excitation generator accepts as input a low-band excitation signal and extends it using a Non-Linear Bandwidth Generator to generate a harmonic high band excitation signal. The harmonic high and excitation signal is adjusted by a multiplication with gain1. Modulated noise is produced by mixing the harmonic-high band excitation signal with random noise. The modulated noise is adjusted by a multiplication with gain2. The adjusted modulated noise and adjusted harmonic high band excitation signal are combined to produce a high band excitation signal. The high band excitation signal is filtered with an LPC Synthesis filter to generate the synthesized high-band mid channel signal. The LPC Synthesis filter uses the quantized high band LPC coefficients from the LPC module.

The LPC module shown in Figure 5.3-13 includes an LPC Estimator that generates high band LPC's. The LPC Quantizer quantizes the high band LPC coefficients (or a representation of them) which are included in the high-band LPC bitstream.

(d) The High Band Gain Estimator

The High Band Gain Estimators compares the high-band mid signal to the synthesized version of the high-band mid signal, over a frame, to generate the gain frame parameter and gain shape parameter.

5.3.2.2.1.3 IC-BWE Encoder Diagram

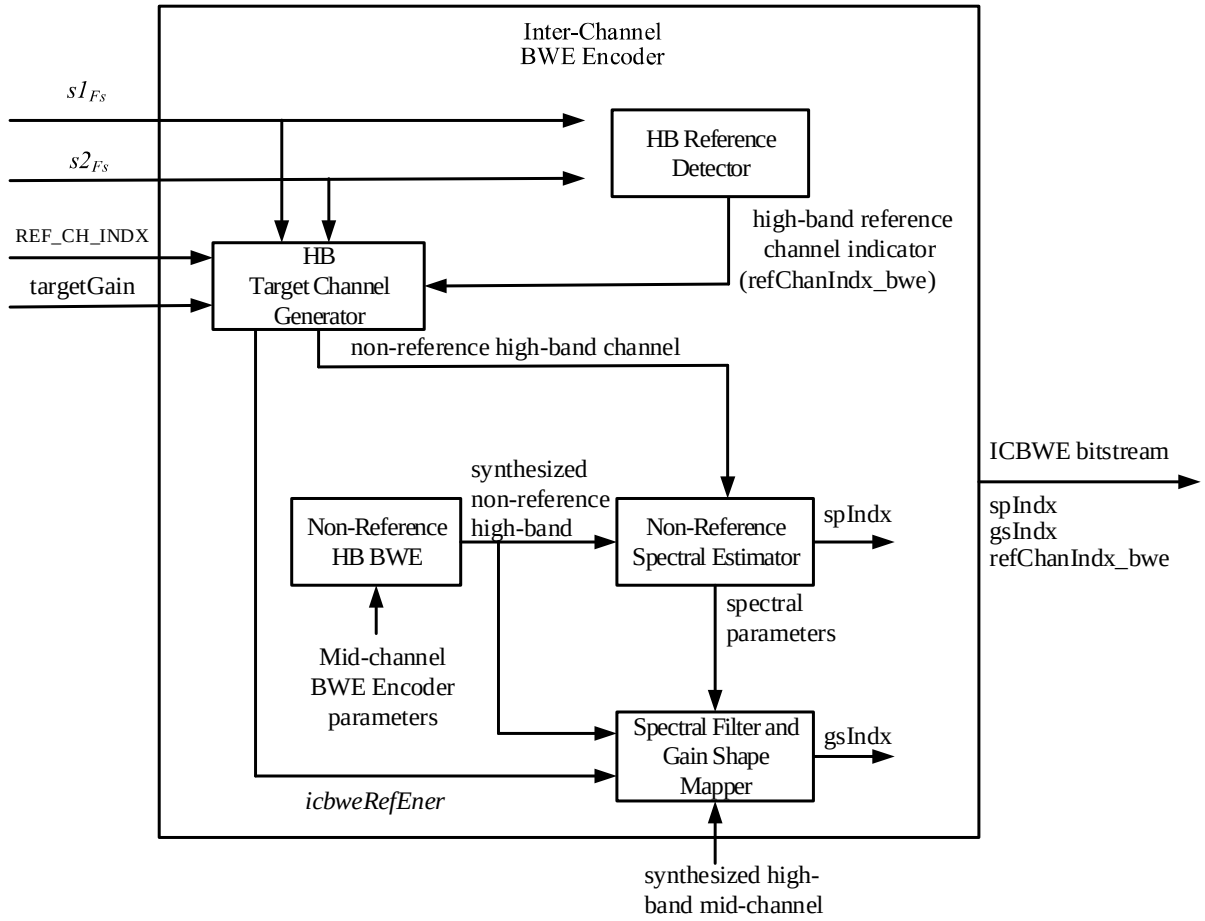


Figure 5.3-4: Block diagram of the IC-BWE Encoder

The IC-BWE Encoder selects either $s1_{Fs}$ or $s2_{Fs}$ as a non-reference target channel based on a high-band reference channel indicator. The reference channel indicator in the HB Target Channel Generator produces a non-reference high band channel signal which is passed to the Non-Reference Spectral Estimator. The Non-Reference HB BWE generates a synthesized non-reference high-band channel based on a non-reference high-band excitation corresponding to the non-reference target channel. The Non-Reference Spectral Estimator estimates the spectral mapping parameters based on the synthesized non-reference high-band channel. The Spectrum and Gain Shape Mapper applies the spectral mapping parameters by filtering the synthesized non-reference high-band signal and generates a spectrally shaped synthesized non-reference high-band channel signal. The IC-BWE produces an IC-BWE bitstream that includes different fields, sub-bitstreams, that include the quantized high band reference channel indicator (refChanIndx_bwe), the quantized spectral index (spIndx), and quantized gain shape index (gsIndx).

5.3.2.2.1.4 HB Reference Detector

The high band's reference channel could be different from the low band's reference channel because the reference channel low band is based on inter channel shift and the reference channel. The selection in the high band is decided based on the inter channel energy differences. The HB reference Detector uses different stages to find the reference channel identification, RCI_bwe_{HB} (refChanIndx_bwe), that designates the incoming stereo (L/R) signal, $s1_{Fs}$ or $s2_{Fs}$, as the high-band portion of the high-band reference signal. The reference channel indicator (RCI [RefChanIndx]) and the targetGain determined by the ICA are used as an initial estimate of determining. In the second stage, the energy of the two channels is computed and compared. The combinaton of both stages selectively update the RCI_bwe_{HB} of the high-band reference signal.

(a) Stage-1

Initialize the refChanIndx_bwe to the refChanIndx (REF_CH_INDX).

When (the targetGain > 1) and the RCI = Right ($s2_{FS}$) switch the RCI to the Left Channel.

When (the targetGain <= 1) and the RCI = Left ($s1_{FS}$) switch the RCI to the Right Channel.

The following pseudocode implements the logic of Stage-1:

If ((targetGain > 1) && refChanIndx == R_CH_INDX) || targetGAin < 1 && refChanIndx = L_CH_INDX

{ refChanIndx_bwe =!refChanIndx}

(b) Stage-2

The two signals, $s1_{Fs}$ and $s2_{Fs}$ are passed through the Tilt Balancer, described in the pre-processor of the ICA Clause 5.3.3.3.2.1.2.1, which is a high pass filter. After filtering, the energy of each signal is calculated by taking the square of the value of the n-sample, and summing over all samples in the frame. The equations below show how *E1* and *E2* are caulcualted.

$$E1 = \sum_{n=0}^{N-1}(s1 * FS[n])^2, \quad (5.3\text{-}10)$$

$$E2 = \sum_{n=0}^{N-1}(s2 * FS[n])^2, \quad (5.3\text{-}11)$$

A comparison of the energies to different threshold is performed. If the energy, *E2*, of $s2_{Fs}$ 1 is less than 64% of the energy, *E1*, of $s1_{Fs}$. The left channel ($s1_{Fs}$) is selected as the reference, Ref_{HB} = Left ($s1_{Fs}$). If the energy, *E1*, is less than 64% of the energy, *E2*, the right channel ($s2_{Fs}$) is selected as the reference, Ref_{HB} = Right ($s2_{Fs}$). If neither condition is met, leave the reference channel unchanged.

The following pseudocode implements the logic of Stage-2:

```
E1 = sum_2(s1Fs);
E2 = sum_2(s2Fs);
if ( E2 < 0.64*E1 )
    refChanIndx_bwe = L_CH_INDX;
 else if ( E1 < 0.64*E2 )
   refChanIndx_bwe = R_CH_INDX;
```

5.3.2.2.1.5 HB Target Generator

The high-band target channel generator receives the (opposite) channel from the reference channel, i.e., the non-reference channel. The non-reference channel filters the low-band signal components of the non-reference channel to generate the non-reference high-band channel, $NonRef_{HB,}$ which is spectrally flipped and provided to (i) the Non-Reference Spectral Estimator and the (ii) Spectral Filter and Gain Shape Mapper.

5.3.2.2.1.6 Non-Reference Spectral Estimator

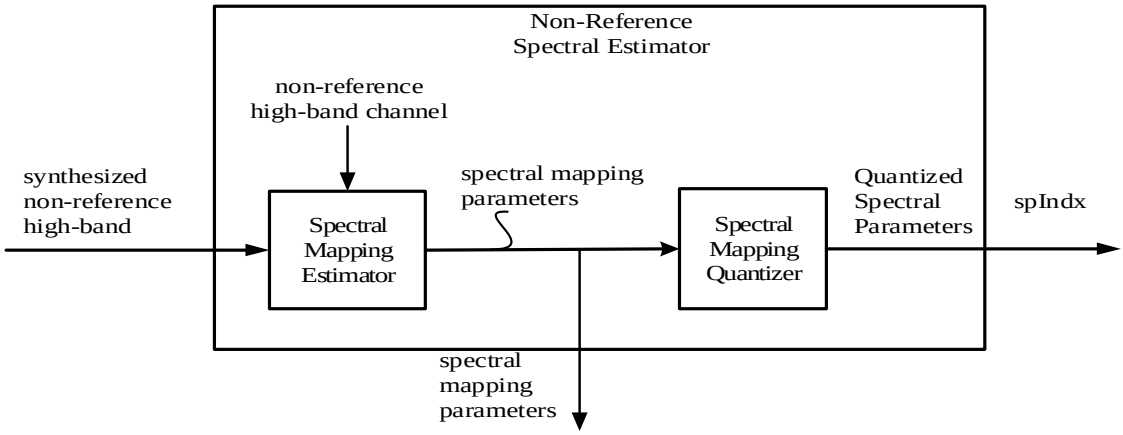


Figure 5.3-5: Block diagram of Non-Reference Spectral Estimator

The Non-Reference Spectral Estimator, in TD mode, estimates the spectral mapping parameters based on distortion reduction measure such that a spectral envelope of a spectrally shaped synthesized non-reference high-band channel is substantially similar to a spectral envelope of a non-reference target channel. The Spectral Mapping Estimator estimates spectral mapping parameters (*u1,u2*) based on the spectral envelope of the synthesized non-reference high-band signal to be approximately close to the spectral envelope of the non-reference high-band channel signal, $NonRef_{HB,}$ signal. The

spectral mapping parameters represent the spectral characteristics (specMapping) of $NonRef_{HB.}$ The spectral mapping parameters (*u1*, *u2*) are quantized by the Spectral Mapping Quantizer. The distortion reduction measure is to use a filter, H(*z*) to substantially match the spectral tilt of the spectral envelope of the non-reference high-band channel signal, $NonRef_{HB}$ (the target), to the spectral envelope of the spectrally shaped synthesized non-reference high-band channel signal (the input). When the filter is a first order filter, the spectral tilt of a signal, *s*(*n*), is expressed as:

$$\frac{r_{ss}[1]}{r_{ss}[0]}, \tag{5.3-12}$$

where

$$r_{ss}[n] = \sum_{i=-(N-1)}^{N-1} s[i] * s[i+n] \tag{5.3-13}$$

is the autocorrelation of signal *s* at lag *n*. H(*z*) is a first order filter used to estimate the tilt, where H(*z*) is expressed as

$$\mathrm{H}(z) = \frac{1}{1-u*z^{-1}}, \tag{5.3-14}$$

where the spectral mapping parameters *u* (u_i *i*=1, 2) are part of the first order filter. The output of the filter in the domain is:

$$y[n] = h[n] \circledast x(n), \tag{5.3-15}$$

where ⊛is a convolution. The autocorrelation of the output is also a convolution of the autocorrelation of the filter with the autocorrelation of the input:

$$r_{yy}(n) = r_{hh}(n) \circledast r_{xx}(n) \tag{5.3-16}$$

Where the autocorrelation of the filter, H(z), is expressed as:

$$r_{hh}(n) = \frac{u^{|n|}}{1-u^2}, \tag{5.3-17}$$

when the filter, H(z), is as shown in (5.3-14).

In computing the reduction measure, the input *x*[n], to the filter is the synthesized $NonRef_{HB}$, synthNonRefHB, where *x*[*n*] represents the n^{th} sample of $synthNonRef_{HB}$. The spectrally shaped synthesized non-reference high band is the output of the filter, *y*[*n*], where *y*[*n*] represents the n^{th} sample of the spectrally shaped synthesized non-reference high-band. The reduction measure includes computing the spectral tilt of $synthNonRef_{HB}$ to be close to the spectral tilt of a different input signal, the target signal, *t*[n], $NonRef_{HB}$, where *t*[*n*], represents sample n^{th} sample of the non-reference high-band channel. The spectral tilt, *T*, of the target signal is expressed as:

$$T = \frac{r_{tt}(1)}{r_{tt}(0)} \tag{5.3-18}$$

To calculate the non-reference target channel signal's tilt, *T*, in terms of the output, *y* [n], T should be solved for as:

$$T = \frac{r_{yy}(1)}{r_{yy}(0)} \tag{5.3-19}$$

The spectral mapping parameters are estimated based on the ratio of the first autocorrelation value of the non-reference target channel signal at lag index one, $r_{yy}(1)$, and the second autocorrelation value of the non-reference target channel signal at lag index zero, $r_{yy}(0)$. The spectral mapping parameters (*u1*, and *u2*) are estimated by expanding equation 5.3-61 in terms of *u*'s exponent to 2. The expansion results in a quadratic equation in terms of *u*, *u* represents the specMapping. The spectral mapping parameters are estimated by using the quadratic equation to match the spectral shape of the of the non-reference target channel and a spectral shape of the spectrally shaped synthesized non-reference high-band channel. The quadratic equation in terms of us is shown below:

$$a * u^2 + b * u + c = 0 \tag{5.3-20}$$

where, *a*, *b*, and *c* are:

$$a = 2 * T * \frac{r_{xx}(2)}{r_{xx}(0)} - \frac{r_{xx}(3)}{r_{xx}(0)} - \frac{r_{xx}(1)}{r_{xx}(0)}, \tag{5.3-21}$$

$$b = 2 * T * \frac{r_{xx}(1)}{r_{xx}(0)} - \frac{r_{xx}(2)}{r_{xx}(0)} - 1, \tag{5.3-22}$$

$$c = T - \frac{r_{xx}(1)}{r_{xx}(0)}, \quad (5.3\text{-}23)$$

where the autocorrelations in (5.3-21)-(5.2-184) are smoothed as follows:

$$\text{rxx(k)} = alpha * \text{rxx(k)} + (1 - alpha) * \text{rxx(k)}, \quad (5.3\text{-}24)$$

where *alpha* =.5, and k = 0..3.

To estimate the spectral mapping parameters, *u1* and *u2*, equation (5.3-20) is solved and *u1* and *u2* are constrained by different criteria (criterion1, criterion2, criterion3). For the current frame, CF, *u1* and *u2*, serve as two spectral mapping parameters candidates to replace the prior spectral mapping parameter a as shown below. Initially,

$$u = specMapping[CF - 1] \quad (5.3\text{-}25)$$

where CF-1 denotes the previous frame specMapping.

Criterion1: If both of the two spectral mapping parameter candidates have an absolute value less than one, the spectral mapping parameter candidate having the smallest value will update the specMapping for the current frame, CF. Criterion1 is expressed as:

$$\text{Criterion1: (a) if } (|u1| < 1) \text{ and (b) if } (|u2| < 1) \text{ then } u = \min(u2, u2), \quad (5.3\text{-}26)$$

Criterion2: If only one spectral mapping parameter candidate has an absolute value less than one, the spectral mapping parameter candidate that has the absolute value less than one will update the specMapping for the current frame, CF.

Criterion2 is expressed as:

$$\text{Criterion2: (a) if } (|u1| < 1)\ u = u1, \text{ or (b)if } (|u2| < 1), u = u2 \quad (5.3\text{-}27)$$

Criterion2 is checked if criterion failed.

Critertion3: If the spectral mapping parameter is negative and greater than -.6 the spectral mapping parameter is unchanged, i.e., *u* is still the previous frame specMapping.

Critertion4: If the spectral mapping parameter is negative and less than -.6 the spectral mapping parameter is changed to -.6.

Criterion5: If the spectral mapping parameter is positive the spectral mapping parameters is changed to zero.

Criterion3, criterion4, and criterion are expressed as:

$$u = \max(\min(u, 0), -0.6), \quad (5.3\text{-}28)$$

where min(arg1, arg2) corresponds to the minimum value between the arguments(*arg1*, *arg2*),

where

$$arg1 = u, \quad (5.3\text{-}29)$$

and

$$arg2 = 0, \quad (5.3\text{-}30)$$

and max corresponds to the maximum value between the arguments(*arg3*, *arg4*), where

$$arg3 = \min(u, 0), \quad (5.3\text{-}31)$$

and

$$arg4 = -.6. \quad (5.3\text{-}32)$$

The logic that implements the equations to solve for the spectral mapping parameters, *u1* and *u2*, are in the following pseudo-code:

```
Smooth rxx[k], k=0..3
smoothed rxx[k] = alpha*rxx[k] + (1-alpha)*rxx[k], where alpha =.5.
```

```
Txx1 = rxx(1)/rxx(0)
Txx2 = rxx(2)/rxx(0)
Txx3 = rxx(3)/rxx(0)
T_desired = T_nonref_target
a = 2 * Txx2 * T_desired - Txx3 - Txx1
b = ( 2 * T_desired * Txx1 - Txx2 - 1.0 )
c = ( T_desired - Txx1)
Set u to previous frame value for gain shape
u = *specMapping;
temp = ( b * b - 4 * a * c )
if ( temp >= 0 && a != 0 )
{
    temp = sqrtf( temp );
    u1 = ( -b + temp ) / ( 2 * a )
    u2 = ( -b - temp ) / ( 2 * a )

  if ( abs( u1 ) < 1.0 && abs( u2 ) < 1.0 )
      u = min( u1, u2 )
  else if ( abs( u1 ) < 1.0 )
      u = u1
  else if ( abs( u2 ) < 1.0 )
     u = u2
 }
}
u = max( min( u, 0 ), -0.6 )
```

The Spectral Mapping Quantizer quantizes the spectral mapping parameters by producing a spectral index, spIndx. The current frame's spectral shape mapping is represented by specMapping. Thus, the continuous range of possible spectral shape values (specMapping) represent the spectral shape parameters. The spectral shape parameters are quantized into discrete set of levels using a quantization table, specMapping_table as expressed below:

$$spIndx = quantize(specMapping, specMapping\_table). \qquad (5.3\text{-}33)$$

Each quantization level in the quantization table represents a specific quantized spectral shape value. The spectral shape index, gsIndx, represents quantized spectral shape mapping parameters, and spIndx is included in the IC-BWE bitstream. Though spIndx varies for different frames, it is used to store the current spectral shape value for the current frame.

5.3.2.2.1.7 Spectral Filter and Gain Shape Mapper

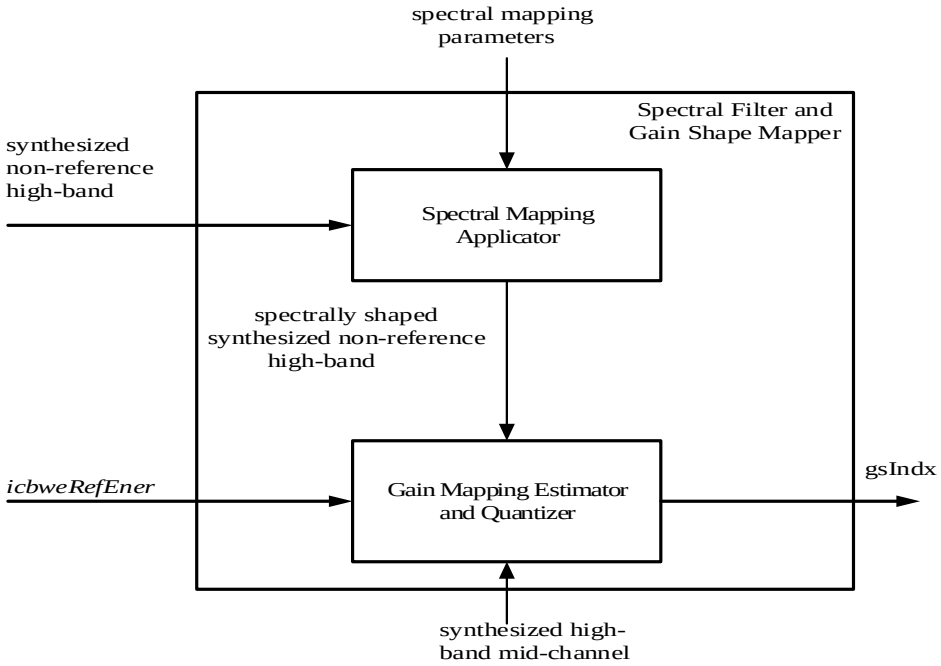


Figure 5.3-6: Block diagram of Spectral Filter and Gain Shape Maper

The Spectral Mapping Applicator, in TD Mode, applies the spectral mapping parameters to the synthesized non-reference high-band signal to generate the spectrally shaped synthesized non-reference high-band signal. The spectral mapping estimator filters the synthesized non-reference high-band signal using the above-described filter $H(z)$, and generates the spectrally shaped synthesized non-reference high-band signal which is provided to the Gain Mapping Estimator and Quantizer. The Gain Mapping Estimator and Quantizer uses a combination of the synthesized high-band mid-channel signal, (synthHB), and spectrally shaped synthesized non-reference high band signal (synthSNonRefHB) to adjust the gain the non-reference high-band channel signal high-band channel signal. The high band gain shape corresponds to a mapping of gain shapes computed by the Gain Mapping Estimator and Quantizer.

The current value of the high band gain shape in a given frame is represented by the parameter, gsMapping.
The obtain the gain shape, the relative gain, relG_targ, between the synthesized signals, synth_{HB} and $\text{spectralsynthNonRef}_{\text{HB}}$ needs to be computed. There are several steps to compute the relative gain, relG_targ.

Step 1: Determine the energy of the $\text{spectralsynthNonRef}_{\text{HB}}$ signal is computed as the square root of the sum of a squares of the $\text{spectralsynthNonRef}_{\text{HB}}$ as expressed below:

$$EnergyspectralsynthNonRefHB = \sqrt{\sum_{i=0}^{N-1} |spectralsynthNonRefHB(i)|\,2} \qquad (5.3\text{-}34)$$

Step 2: Determine the desired gain, gDes, as the ratio of the energy of the $\text{spectralsynthNonRef}_{\text{HB}}$, EnergyspectralsynthNonRefHB signal to the energy, icbweRefEner, of RefHB, as expressed below:

$$gDes = EnergyspectralsynthNonRefHB / icbweRefEner \qquad (5.3\text{-}35)$$

If gDes is 0, icbweRefEner is set to 1 to avoid dividing by zero.

Step 3: Calculate the geometric mean between gDes, and the past Frame's gDes (gDespastFrame) as expressed below:

$$relG\_targ = (gDes^{-.5}) * gDespastFrame^{-.5} \qquad (5.3\text{-}36)$$

The next step to determine the gain shape mapping, gsMapping, is to compute the gain of the synthesized signals, synth_{HB} and $\text{spectralsynthNonRef}_{\text{HB}}$.

$$gainsHB = \sum_{i=0}^{N-1} |synthHB(i)| \qquad (5.3\text{-}37)$$

$$gainsSNonRefHB = \sum_{i=0}^{N-1} |synthSNonRefHB(i)| \qquad (5.3\text{-}38)$$

If gainSNonRefHB is zero, the prior gain shape, gsMapping is used for the value of gsMapping, otherwise

$$gsMapping = \frac{relG\_targ * gainsHB}{gainSNonRefHB} \qquad (5.3\text{-}39)$$

The Gain Mapping Estimator and Quantizer quantizes the high band gain shape, gsMapping, of *synthSNonRefHB*. The quantized high band gain shape is represented with a table, icbwe_gsMapping_table and quantization gain shape index value, gsIndx, of the table. Thus, the continuous range of possible gain shape values (gsMapping) represent gain shape parameters. The gain shape parameters are quantized into discrete set of levels using a quantization table, gsMapping_table as shown below:

$$gsIndx = quantize(gsMapping, gsMapping\_table). \qquad (5.3\text{-}40)$$

Each quantization level in the quantization table represents a specific quantized gain shape value. The gain shape index, gsIndx. represents quantized gain shape mapping parameters, and gsIndx is included in the IC-BWE bitstream. Though the gain shape index varies for different frames, it is used to store the current quantized gain shape value for the current frame.

5.3.2.2.1.8 Non-Reference HB BWE

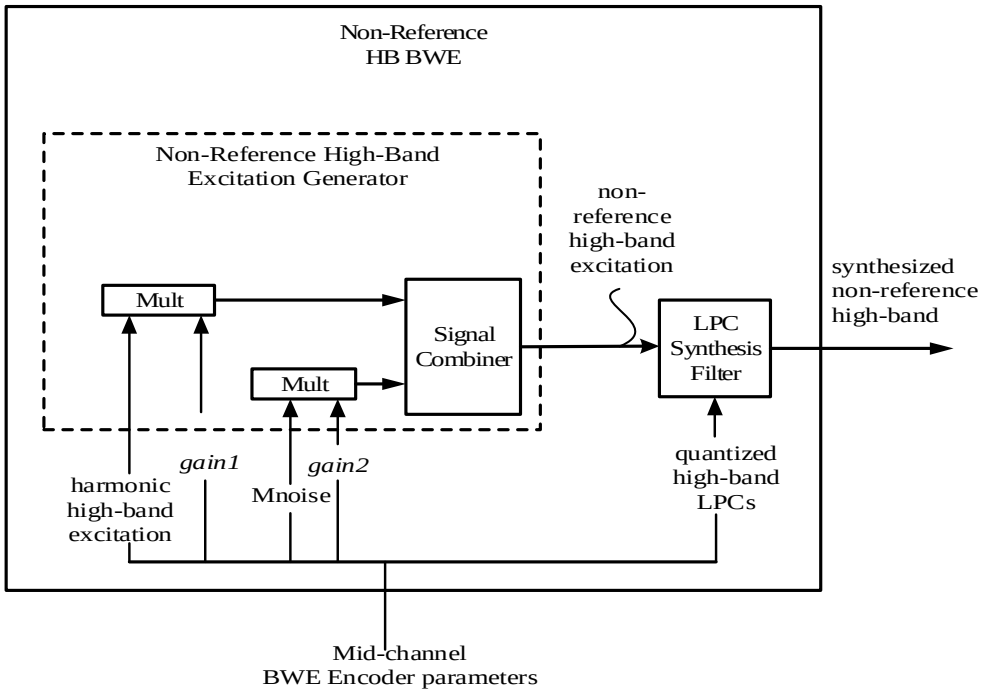


Figure 5.3-7: Block diagram of the Non-Reference HB BWE

The Non-Reference HB BWE uses the Mid-Channel BWE Encoder parameters provided by the Mid-Channel BWE Encoder as described above in Clause 5.3.2.2.1.2 which include the non-linear harmonic high-band excitation and modulated noise, and gains, gain1 and gain2. A gain-adjusted harmonic high-band excitation signal is produced by multiplying the non-linear harmonic high-band excitation by gain1. A gain-adjusted modulated noise signal is produced by multiplying the modulated noise by gain2. The non-reference high-band excitation signal is produced by combining the gain-adjusted harmonic high-band excitation signal and the adjusted modulated noise signal. An LPC synthesis filter generates a synthesized non-reference high band signal based on the non-reference high band excitation signal and the quantized high-band LPC's also received from the Mid-Channel BWE Encoder. The synthesized non-reference high-band signal is provided to the Non-Reference Spectral Estimator.

5.3.2.2.2 Front VAD

The front VAD is applied on the input stereo signals for signal activity detection to aid stereo classification and selection between the TD-based stereo and DFT-based stereo, as well as coordinating the selection of CNG encoding mode for the input stereo channels.

The signal activity detection for each input stereo signal is done as for the IVAS core, see clause 5.2.2.2.5, including DTX hangover addition as described in clause 5.1.12.8 of [3]. The front VAD has independent states for the left and right channel and is independent from the IVAS core VAD(s). The speech/music classifier input for the front VAD is however shared with the core VAD (channel 0), obtained from the previous frame, to reduce complexity.

A joint VAD decision VAD_{01} for the input stereo signals is obtained as

$$VAD_{01} = VAD_0 \bigvee VAD_1 \quad (5.3\text{-}41)$$

where VAD_0 and VAD_1 are signal activity flags indicating activity for channel 0 and 1 respectively.

5.3.2.3 TD-based stereo

5.3.2.3.1 Inter-channel Alignment (ICA)

5.3.2.3.1.1 Overview

The Inter-Channel Alignment module is used in based on different scenarios. Audio capture devices in teleconference rooms, or telepresence rooms, include multiple microphones that acquire spatial audio. The spatial audio may include speech as well as background audio that is encoded in a near-end device and transmitted to a far-end device. The speech/audio from a given source (e.g., a talker) arrives at the multiple microphones at different times depending on how the microphones are arranged as well as where the source (e.g., the talker) is located with respect to the microphones and room dimensions. For example, when the sound source is closer to one microphone, mic1, on the near-end device than another microphone, mic2, on the near-end device the sound emitted from the sound source reaches the first microphone earlier in time than the second microphone. The microphones can also receive/detect sounds from multiple sound sources. The multiple sound sources often include a dominant sound source (e.g., a talker) and one or more secondary sound sources (e.g., a passing car, traffic, background music, street noise). The sound emitted from the dominant sound source could reach the first microphone earlier in time than the second microphone.

To illustrate the concept of ICA, consider a teleconference room setting as shown below in Figure 5.3-8.

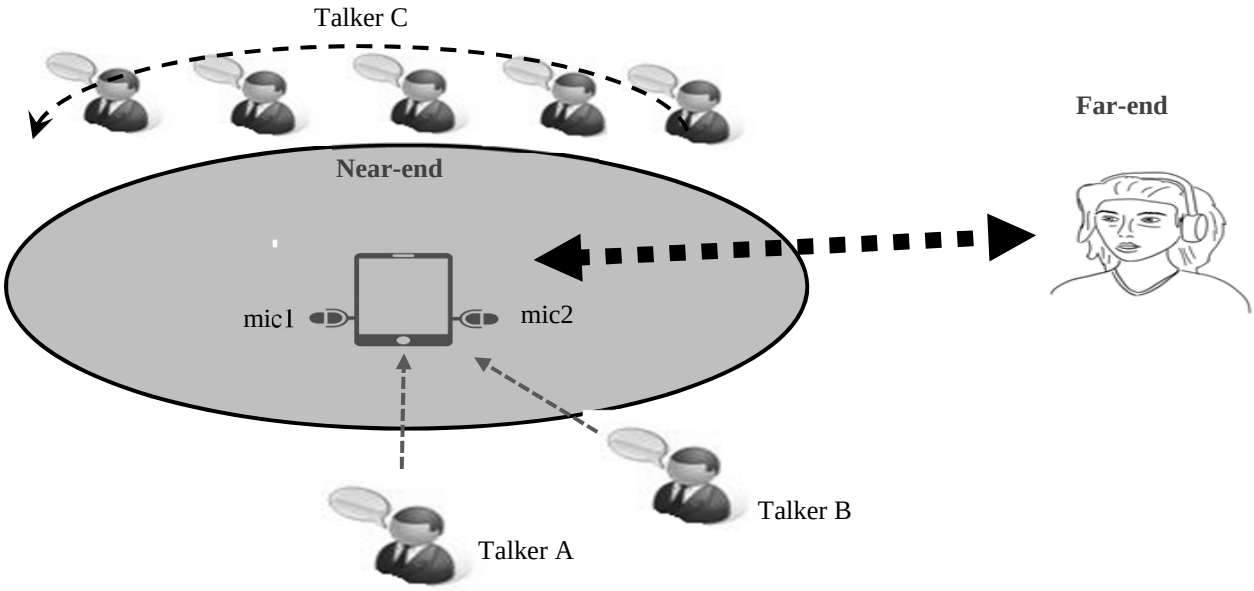


Figure 5.3-8: Teleconference Scenarios

Case 1: Temporally Aligned Channels

If Talker A is the person speaking, the sound emitted by Talker A arrives at both near end device microphones, mic1 and mic2, at the same time. In such a case, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally aligned, as shown in Figure 5.3-9 below.

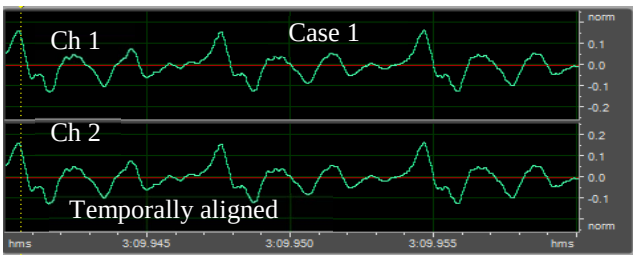


Figure 5.3-9: Temporally Aligned Audio Signals in Ch1 and Ch2

Case 2: Channels temporally NOT aligned

Case 2a: Source to Microphones Distance Difference

If Talker B is the person speaking, the sound emitted by Talker B arrives at both near end device microphones, mic1 and mic2, at different times. In such a case, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally not aligned.

Case 2b: Position of Source is Moving relative to Microphones

If Talker C, whose position is moving relative to the microphones, mic1 and mic2, is the person speaking, the sound emitted by Talker C arrives at both near end device microphones, mic1 and mic2 at different times. In such a case, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally not aligned.

Case 2c: Overlapping Sources speaking simultaneously

If Talkers B and C are speaking at the same time, there speech is overlapping, and the sound emitted by both of them arrives at both near end device microphones, mic1 and mic2, at different times, which results in varying temporal shift values between the speech of Talker B and C depending on who is the loudest talker, closest to the microphone, etc.

In such scenarios, the channels, Ch1 and Ch2, associated with microphones, m1 and m2, are temporally not aligned, i.e., there is a temporal mismatch between the channels.

In each of these scenarios, the audio signals in the channels are shifted and temporally misaligned as shown in Figure 5.3-10 below.

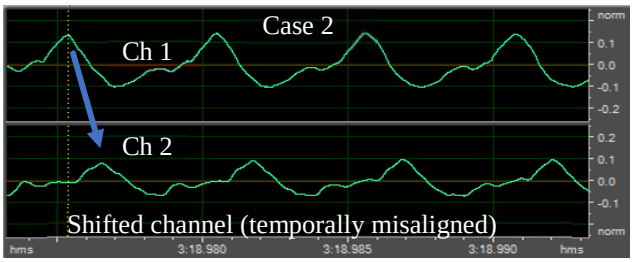


Figure 5.3-10: **Temporally Misaligned Audio Signals in Ch1 and Ch2**

The near-end device encodes the audio signal captured by the microphones, mic1 and mic2, in different frames that contain varying PCM samples depending on the sampling rate, e.g., 960 samples @ 48kHz. Mid-side (MS) coding and parametric stereo (PS) coding are stereo coding techniques that provide improved efficiency over the dual-mono coding techniques.

In dual-mono coding, the Left (L) channel (or signal) and the Right (R) channel (or signal) are independently coded without making use of inter-channel correlation. MS coding reduces the redundancy between a correlated L/R channel-pair by transforming the Left channel and the Right channel to a sum-channel and a difference-channel (e.g., a side channel) prior to coding. The sum signal and the difference signal are waveform coded in MS coding. More bits are spent on the sum signal than on the side signal.

Depending on a recording configuration, i.e., the placement of the microphones relative to one or more sound sources, there is a temporal shift between a Left channel and a Right channel, as well as other spatial effects such as echo and room reverberation. When the temporal shift and phase mismatch between the channels are not compensated, the sum channel and the difference channel contain comparable energies reducing the coding-gains associated with MS or PS techniques. The reduction in the coding-gains is based on the amount of temporal (or phase) shift. The comparable energies of the sum signal and the difference signal limits the usage of MS coding in certain frames where the channels are temporally shifted but are highly correlated.

When the sound source is closer to the first microphone, mic1, than to the second microphone, mic2, frames of the second audio signal are delayed relative to frames of the first audio signal. In this case, the first audio signal is referred to as the “reference audio signal” or “reference channel” and the delayed second audio signal is referred to as the “target audio signal” or “target channel.” Alternatively, when the sound source is closer to the second microphone than to the first microphone, frames of the first audio signal is delayed relative to frames of the second audio signal. In this case, the second audio signal is referred to as the reference audio signal or reference channel and the delayed first audio signal is referred to as the target audio signal or target channel.

The encoder determines the target signal, or the reference signal based on a mismatch value (e.g., an estimated shift value or the final shift value) corresponding to a current frame to be encoded and mismatch (e.g., shift) values corresponding to previous frames.

5.3.2.3.1.2 Inter-Channel Aligner

5.3.2.3.1.2.1 Overview

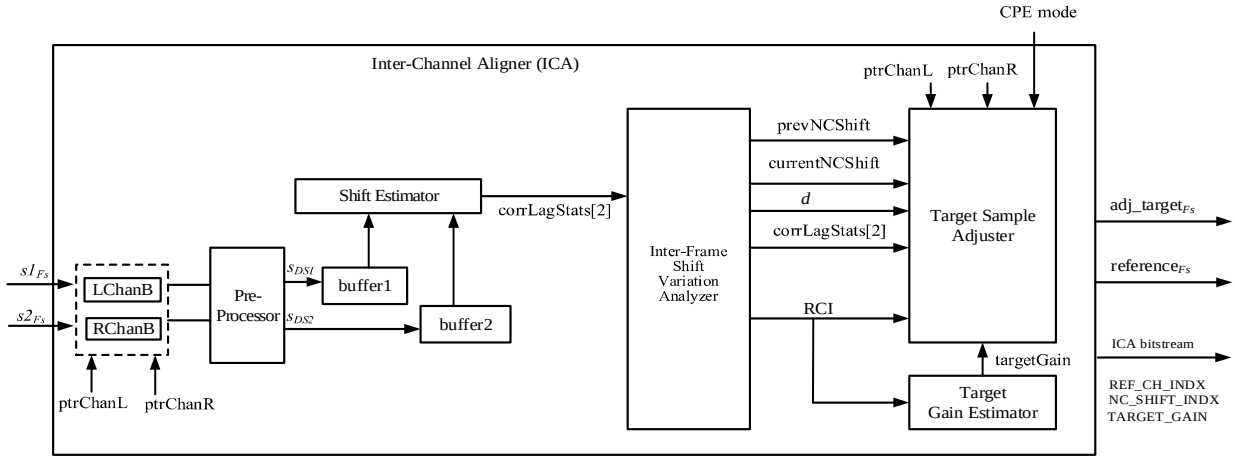


Figure 5.3-11: Block diagram of the Inter-Channel Aligner

The Inter-Channel Aligner determines a temporal shift value indicative of a shift of the first audio signal relative to the second audio signal. The shift value corresponds to an amount of temporal delay between receipt of the first audio signal at the first microphone and receipt of the second audio signal at the second microphone. The Inter-Channel Aligner determines the shift value on a frame-by-frame basis, e.g., based on each 20 milliseconds (ms) speech/audio frame and operates to align the two audio signals as described in the Overview Clause above. The output of the ICA is an Adjusted Target Signal and a Reference Signal. The Inter-Channel Aligner includes six sub-blocks: (a) Pre-Processor; (b) Shift Estimator; (c) Inter-Frame Shift Variation Analyzer; (d) Target Sample Adjuster; (e) Target Gain Estimator; and (f) Lookahead Target Buffer Adjuster. Each is described in the next clauses.

5.3.2.3.1.2.2 Pre-Processor

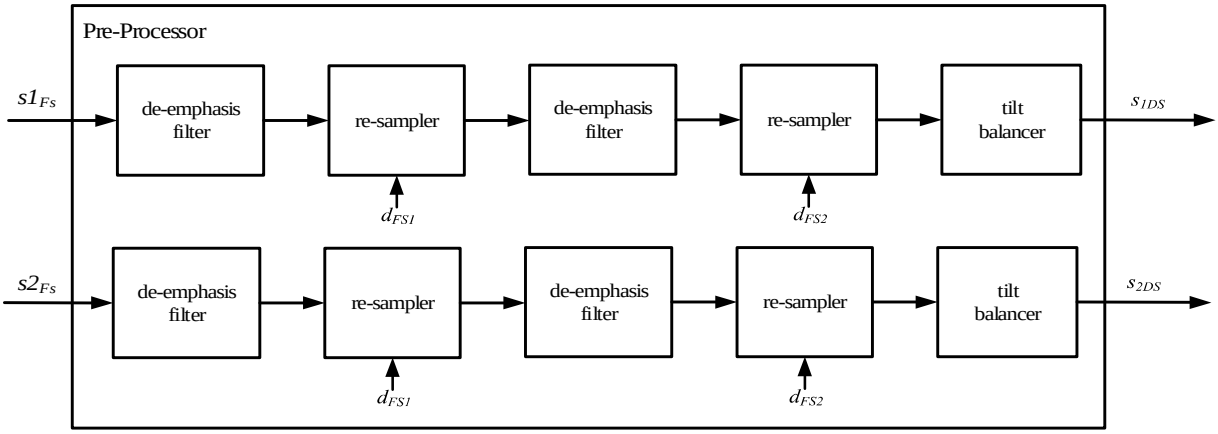


Figure 5.3-12: Block diagram of the Pre-Processor

The Pre-Processor includes de-emphasis filters and resamplers that convert an input signal from a first sample rate to a second sample rate, based on a sample rate factor (c). The resamplers function as down-samplers. The first audio signal after the de-emphasis filter is downsampled by sample rate factor (d_{FS1}), then passed through another deemphasis filter again and downsampled (d_{FS2}). Similarly, the second audio signal after the de-emphasis filter is downsampled by sample rate factor (d_{FS1}), then also passed through another de-emphasis filter again and downsampled (d_{FS2}).

The de-emphasis filters counteracts the effect of boosting higher frequencies that may have been done during analog to digital conversion. The de-emphasis filter is implemented using the following equation:

$$\frac{1}{1-u*z^{-1}}, \quad (5.3\text{-}42)$$

where u is the pre-emphasis factor.

The de-emphasis on the stereo signals is performed twice to ensure the audio signals are properly equalized at the lower sampling rate. Application of the de-emphasis filters before each stage of down-sampling ensures that the high-frequency components of the audio signals are reduced to preserve the quality of the audio signal at the lower sampling rate, i.e., there is not any unwanted distortion introduced into the signal. Performing de-emphasis before down-sampling minimize aliasing, as it reduces the high-frequency content that could potentially be aliased to lower frequencies when the sampling rate is reduced.

Each of downsampled signals are then passed into a tilt (spectral) balancer which applies a high-pass filter to the input signal to remove low-frequency rumble and compensate for a low pass effect due to the de-emphasis filters. The filter coefficients are defined based on the input sampling rate. The output of the pre-processor are first re-sampled signal, s_{1DS} , and the second re-sampled signal, s_{2DS}.

5.3.2.3.1.2.3 Shift Estimator

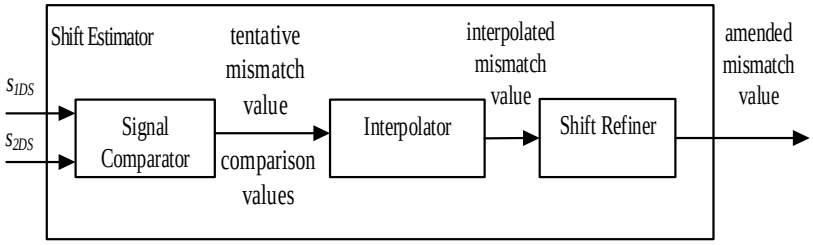


Figure 5.3-13: Block diagram of the Shift Estimator

The Shift Estimator includes three sub-blocks: (a) the Signal Comparator; (b) the Interpolator; and (c) the Shift Refiner.

(a) Signal Comparator

The signal comparator produces comparison values, $CompVal_N(k)$, i.e., cross-correlations to obtain an initial lag estimate of the correlation between the first and second audio downsampled signals. Though the reference channel and target channel will be determined at the end of the ICA process, each of these estimated comparison values (corrEst in code) are indicative of the amount of temporal mismatch between the reference frame of the reference channel and the target frame of the target channel. The initial lag estimate represents a tentative mismatch value. The signal comparator estimates the comparison values based on the first re-sampled signal and the second re-sampled signal stored in two buffers (buffer1 and buffer2). A sliding window cross-correlation calculation, $CompVal_N(k)$ between two re-sampled signals, s_{1DS} and s_{2DS}, measure the similarity between the left and right channels of a stereo audio signal at different time shifts (lags) over a lag (k) range. If the signals in buffer1 and buffer2 are perfectly aligned and identical, the cross-correlation will be maximum at zero lag, (k=0). If the signal in buffer1 leads or lags behind the signal in buffer2, the cross-correlation will be maximum at a positive or negative lag, respectively. By searching over a range of lags, the time shift that best aligns the two re-sampled signals is estimated. The cross-correlation at a specific lag, k, is calculated as the dot product of the signals in buffer1 and buffer2 shifted by the lag, and measures how much the signals overlap for the given time shift.

The process of measuring similarity is done for both negative and positive lags. The corrEst memory buffer is updated with the latest cross-correlation value for each lag(k). These lags define the range of shifts used to compare the two re-sampled signals. The lag search range includes a minimum and maximum lag to compute cross-correlation between the two resampled signals in buffer1 and buffer2. The cross-correlation for negative lags is computed when the signal in buffer1 lags behind the signal in buffer2. The cross-correlation for positive lags is computed when the signal in buffer1 leads the signal in buffer2. The initial estimate of the correlation lag (corrLogStats[0]) is obtained by finding the maximum value in the cross-correlation between the two audio channels and the corresponding lag over the lag search range (minlag-maxlag). The signal comparator smooths the comparison values to generate short-term smoothed comparison values. $CompVal_{ST\_N}(k)$. $CompVal_{ST\_N}(k)$ is generated by smoothing the comparison values of the frames in vicinity of the current frame being processed by the following expression:

$$CompValST\_N(k) = (CompValN(k) + CompValN-1(k) + CompValN-2(k))/3 \quad (5.3\text{-}43)$$

The short-term smoothed comparison values may be the same as the comparison values generated in the frame being processed $CompVal_{ST\_N}(k)$.

The signal comparator retrieves comparison values for previous frames of the re-sampled signals modifies the comparison values based on a long-term smoothing operation using the comparison values for previous frames.
The signal comparators smooths the comparison values to smooth the comparison values to generate first long-term smoothed comparison values based on a smoothing parameter, α. The comparison values include the long-term smoothed comparison value $CompVal_{LT\_N}(k)$ for a current frame (N) and is represented by:

$$CompValLT\_N(k) = (1 - \alpha) * CompValN(k) + \alpha * CompValLT\_N - 1(k), \qquad (5.3\text{-}44)$$

where $\alpha \in (0, 1.0)$.

The pseudo-code is as follows:

```
corrEstLTN = (alpha)*corrEstLTN-1
corrEstN =(1-alpha)*corrEstN
corrEstLTN = (1-alpha)* corrEstN  + (alpha)* corrEstLTN-1
```

The long-term smoothed comparison value is based on a correlation of short-term comparison values to long-term comparison values. Thus, the long-term smoothed comparison value $CompVal_{LT\_N}(k)$ is based on a weighted mixture of the instantaneous comparison value $CompVal_N(k)$ at frame N and the long-term smoothed comparison values, $CompVal_{LTN-1}(k)$ for one or more previous frames. As the value of α increases, the amount of smoothing in the long-term smoothed comparison value increases. When the comparison values of the current frame are very similar to the long-term smoothed comparison values, it is an indication of a stationary talker, and this could be used to control the smoothing parameters to further increase the smoothing (i.e., increase the value of α). On the other hand, when the comparison values as a function of the various shift values does not resemble the long-term smoothed comparison values, the smoothing parameters can be adjusted to reduce smoothing (i.e., decrease the value of α). The value of the smoothing parameters (α) are based on the short-term signal level (E_{ST}) [tempF in code] and the long-term signal level, E_{LT}, (ica_envVarLT in code] of the first and second re-sampled signals. The short-term signal level is calculated for the frame (N) being processed $E_{ST}(N)$ as the sum of the sum of the absolute values of the downsampled reference samples and the sum of the absolute values of the downsampled target samples.

The long-term signal level, E_{LT} [CorrEstLT in code], is a smoothed version of the short-term signal levels and shown below:

$$ELT(N) = 0.6 * ELT(N - 1) + 0.4 * EST(N) \qquad (5.3\text{-}45)$$

The value of the smoothing parameter, α, is adaptive is controlled according to the pseudo-code described as follows:

```
Set α initial value (0.93);
If EST > 4* ELT, modify the value alpha (.83)
If EST > 2*ELT and EST is less than or equal to (4* ELT), modify the value of alpha (.85),
else if (EST > ELT), set alpha to .90.
```

The value of the smoothing parameter is reduced if the short-term signal level indicators are greater than the long-term signal level indicators. The signal comparator calculates a cross-correlation value [*corrEst_ncorr in code] between the comparison values for a single frame ("instantaneous comparison values") and short-term smoothed comparison values, which may be for either the current frame or for a past frame. The cross-correlation value ($CrossCorr\_CompVal_N$) of the comparison values for the frame N provides instantaneous comparison values for the frame N, is calculated as

$$CrossCorr\_CompVal_N = (\sum_k CompVal_{ST_N}(k) * CompVal_N(k))/Fac, \qquad (5.3\text{-}46)$$

where the short-term smoothed comparison values (e.g., $CompVal_{ST_N}(k)$) is a single value estimated per each frame (N). 'Fac' is a normalization factor chosen such that the $CrossCorr\_CompVal_N$ is restricted between 0 and 1, where Fac is calculated as

$$Fac = \sqrt{(\sum_k CompVal_{ST_N}(k) * CompVal_{ST_N}(k)) * (\sum_k CompVal_N(k) * CompVal_N(k))}. \qquad (5.3\text{-}47)$$

The signal comparator provides the comparison values and the initial lag, i.e., the tentative mismatch value to the interpolator. The process just described by the signal comparator is illustrated in the figure below. Once the CrossCorr_CompVal are computed for the current frame, there may be a boost to some of the long term smoothed comparisons. The signal comparators also compares the cross-correlation value of the comparison values ($CrossCorr\_CompVal_N$) with a threshold and may adjust (boost) the first long-term smoothed comparison values in response to the determination that the cross-correlation value of the comparison values ($CrossCorr\_CompVal_N$)

exceeds the threshold. When the cross-correlation value of the comparison values ($CrossCorr\_CompVal_N$) is compared to a threshold (.8) and is bigger than or equal to the threshold, it indicates the cross-correlation value between comparison values is quite strong, indicating small or no variations of temporal shift values between adjacent frames. Thus, the estimated temporal shift value of the current frame (N) cannot be too far off from the temporal shift values of the previous frame (N-1). Therefore, in response to determining that the cross-correlation value exceeds the threshold, the signal comparator adjust the long-term smoothed comparison values by a factor of 1.2 (20% boost) by multiplying a scaling factor of 1.2 to the long term smoothed comparison values, to generate new long-term smoothed comparison values.

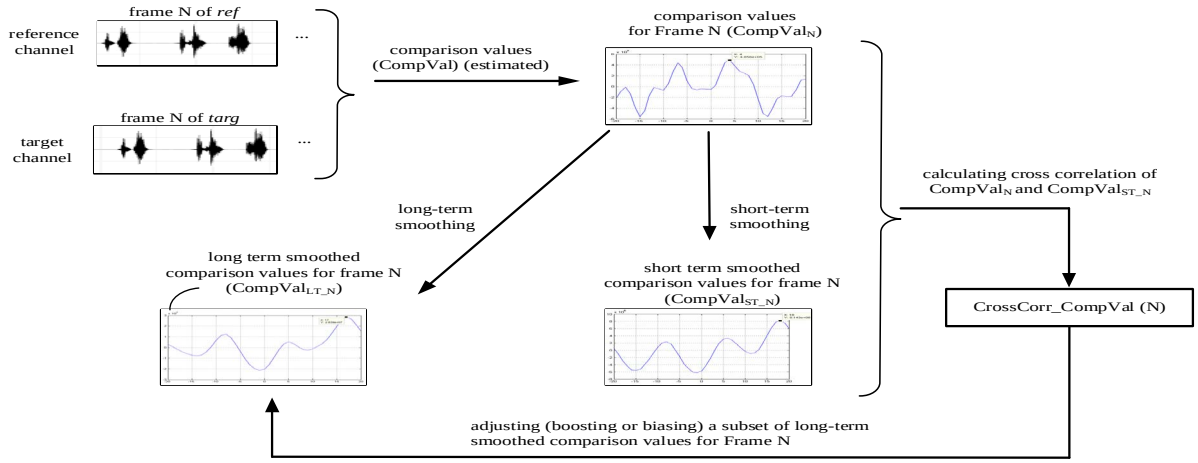


Figure 5.3-14: Diagram of the smoothing process in the Signal Comparator

Linear regression is performed to estimate the correlation of the previous frame. The regression is based on the delay_0_mem[], which stores the delay values for the previous MAX_DELAYREGLEN frames. The regression line is characterized by the slope beta_reg and the y-intercept alpha_reg. The slope beta_reg is calculated as the covariance of X and Y (where X is the frame index and Y is the delay value) divided by the variance of X. The y-intercept alpha_reg is then calculated as the mean of Y minus the slope beta_reg times the mean of X. The predicted correlation of the previous frame, reg_prv_corr, is then calculated as the slope beta_reg times the total number of frames MAX_DELAYREGLEN plus the y-intercept alpha_reg as shown in the equation below.

$$reg\_prv\_corr = beta\_reg * MAX\_DELAYREGLEN + alpha\_reg \quad (5.3\text{-}48)$$

The prediction of the correlation of the previous frame, reg_prv_corr, serves as an estimate of the temporal shift that needs to be applied to the current frame to align it with the previous frame. This is useful to exploit the temporal redundancies between successive frames to achieve efficient compression. The predicted correlation from the past frame (reg_prv_corr) is used as a reference point to check and adjust the initial correlation lag estimate (corrLagStats[0]). If the initial lag estimate deviates significantly from the predicted correlation from the past frame, the initial estimate may not be accurate. In such a case, the initial lag estimate is updated to be closer to the predicted correlation from the past frame. This helps to ensure a more accurate temporal adjustment of the current frame, leading to better audio quality and coding efficiency. Furthermore, the predicted correlation of the previous frame is also used to calculate a bias and width for a local weighting window loc_weight_win[]. The predicted correlation of the previous frame is then used to adjust the initial correlation estimate (corrEst[i]) by applying the local weighting window (loc_weight_win[]).

The pseudo-code below implements the adjustment of the correlation estimates:

```
for (i = 0, j = L_NCSHIFT_DS - TRUNC(reg_prv_corr); i < 2 * L_NCSHIFT_DS + 1; i++, j++)
      corrEst[i] *= loc_weight_win[j];
```

The local weighting window has a bias towards the estimated lag, which means that correlation estimates around the estimated lag are given more weight. This can help to emphasize or de-emphasize certain lags in the correlation estimate based on the predicted correlation of the previous frame.

While corrLagStats[0] is an initial estimate of the tentative shift value, i.e.., the lag, based on the new long-term smoothed comparison values, by finding the maximum value of the cross-correlation, it does not consider any temporal trends in the delay between the two audio channels. The regression analysis, on the other hand, fits a linear model to the delay values across different time points. This captures trends in the delay, such as if the delay is consistently increasing

or decreasing over time. The resulting estimate (reg_prv_corr) could be more accurate if such trends are present. Using both the regression and the correlation provides for a more robust lag estimate. corrLogStats[0] is updated based on the results of the regression analysis.

(b) Interpolator
A sinc interpolation window is used and is based on the downsample factor, dsFactor. The equation below scales the initial correlation lag estimate by the downsample factor to get an initial estimate of the correlation lag at the original sampling rate.

$$corrLagStats[1] = corrLagStats[0] * dsFactor, \qquad (5.3\text{-}49)$$

where dsFactor is dFs1*dFs2.

For each lag in the interpolation range [interpMin and interpMax], an interpolation of the cross-correlation values is computed as a weighted sum of the cross-correlation values at nearby lags. The weights are given by the values of the sinc interpolation window. The maximum value of the interpolated cross-correlation is used to refine the initial estimate of the correlation lag, which is stored in corrLagStats[1], and also the computed correlation lag statistics are saved for the current frame.

The interpolator extends the tentative mismatch value by generating an interpolated mismatch values corresponding to mismatch values that are close to the tentative mismatch value by interpolating the comparison values. The comparison values are based on a coarser granularity of the mismatch values previously calculated. The interpolator generates interpolated comparison values by performing interpolation on the comparison values that were generated in the Signal Comparator. The interpolator determines interpolated mismatch values based on the interpolated comparison values. The mismatch values that are output of the interpolator are based on the interpolated mismatch values. The interpolator modifies the comparison values after the regression analysis which used the past frame. Thus, the interpolated mismatch values may include values from past frames.

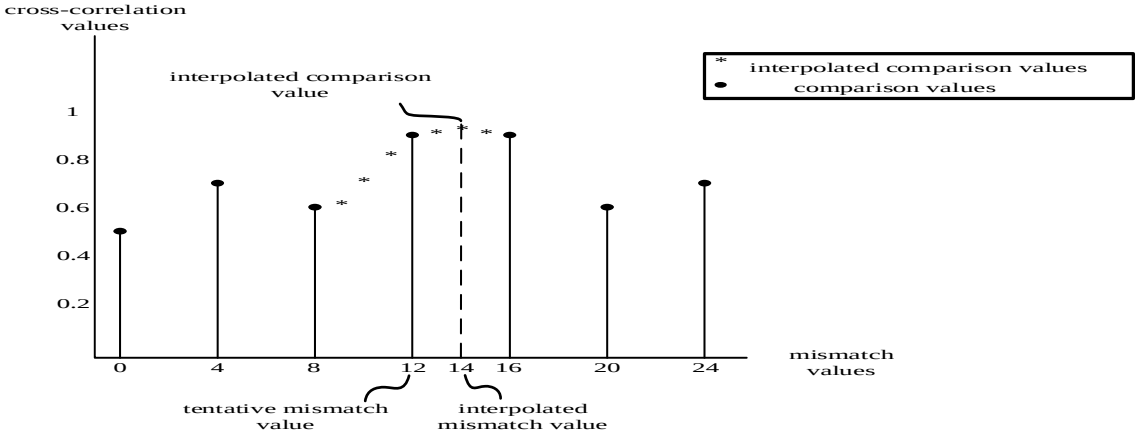


Figure 5.3-15: **Diagram of the Interpolation Process**

(c) Shift Refiner

The interpolator provides the interpolated mismatched values to the Shift Refiner. The Shift Refiner produces a more accurate measure of the temporal similarity by searching around the estimated interpolated mismatch value of the current frame and the final estimated mismatch value of the previous frame. This more accurate measure leads to a better ("amended") mismatch value. The amended mismatch value is further conditioned to estimate the final mismatch value by limiting any spurious changes in the mismatch value between frames and further controlled to not switch from a negative mismatch value to a positive mismatch value (or vice versa) in two successive frames. The Shift Refiner provides the amended mismatch value and to the Inter-Frame Shift Variation Analyzer and refines the ICA statistics. The non-causal shift value (corrLagStats[2]) determination is based on the tentative shift value. Though, the tentative shift value is modified when interpolated, and the interpolated shift value refined based on the ICA statistics to generate an amended shift value (corrLagStats[2]).

5.3.2.3.1.2.4 Inter-Frame Shift Variation Analyzer

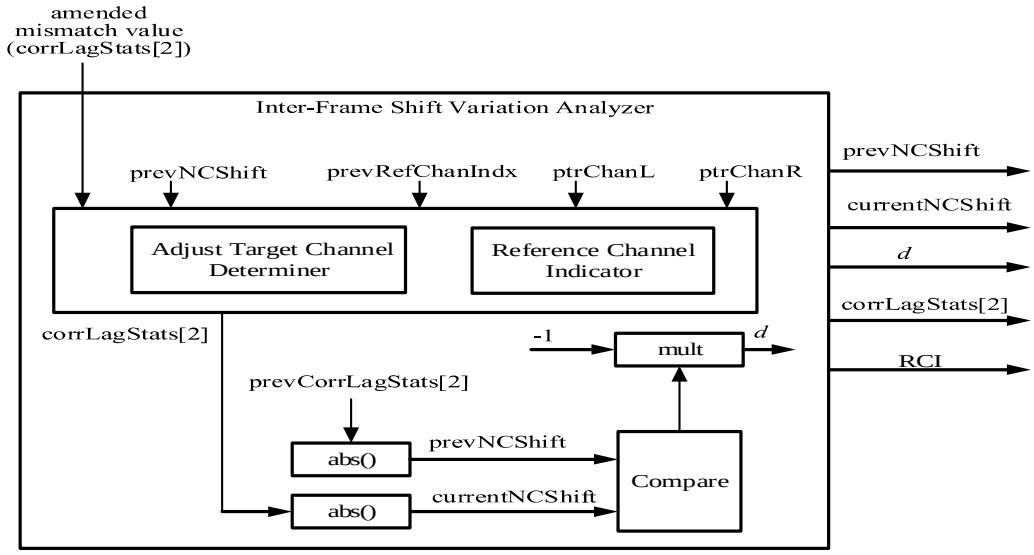


Figure 5.3-16: **Block diagram of the Inter Frame Shift Variation Analyzer**

The Inter-Frame Shift Variation Analyzer performs four functions: (a) reset of the amended shift value; (b) implements the logic of the Adjust Target Channel Determiner; (c) implements the logic of the Reference Channel Indicator (RCI); and (d) computes the inter-frame shift variation.

(a) Reset the amended shift value

The amended shift value, corrLagStats[2], from the Shift Estimator is reset if either of two conditions are met.

Condition 1: the current corrLagStats[2],from the Shift Estimator, is less than zero and previous corrLagStats[2] is greater than zero.

Condition 2: the current corrLagStats[2], from the Shift Estimator, is greater than zero and the previous corrLagStats[2] is less than zero.

The resetting of the corrLagStats[2] is the final updating of the amended mismatch value, i.e., the final mismatch value, which is used to compute the current non-causal shift value.

The following pseudo-code is used to implement the logic of resetting the amended shift value:

```
if (corrLagStats[2] <0) && (prevCorrLagStats[2] >0) || corrLagStats[2] >0 && prevCorrLagStats[2] < 0
corrLagStats[2] =0;
```

(b) Adjusted Target Channel Determiner

The Adjusted Target Channel Determiner tracks if there is a reverse in the timing between the first and second audio signals by determining whether the mismatch value indicates a switch or reverse in timing between the first audio signal and the second audio signal stored in Buf1 and Buf2, as described with reference to the figure below. A switch or reverse in timing indicates that a final mismatch value corresponding to the frame N-1 has a first sign that is distinct from a second sign of the amended mismatch value corresponding to the frame N or greater (i.e., a positive to negative transition or vice-versa). Case 1: a reverse or a switch in timing indicates that the first audio signal is received, at frame N-1, prior to the second audio signal. For a subsequent frame (e.g., N, or greater) the second audio signal is received prior to the first audio signal. Case 2: a reverse or a switch in timing indicates that, for the frame N-1, the second audio signal is received prior to the first audio signal, and, for a subsequent frame (N or greater) the first audio signal is received prior to the second audio signal. The figure below illustrates the state diagram used to determine the target channel. The (final) mismatch values are represented as T for the current amended mismatch value and Tprev for a previous mismatch value.

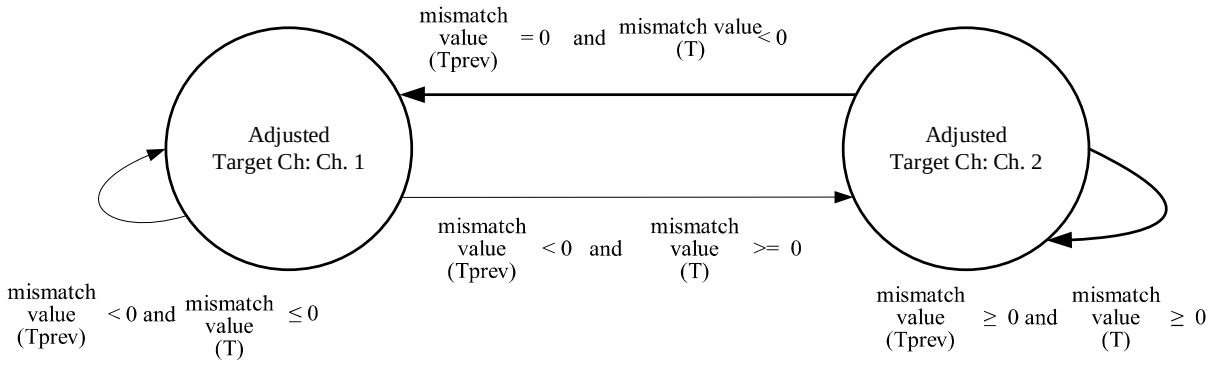


Figure 5.3-17: Block diagram of state machine in the Adjust Target Determiner

The state diagram above uses the final mismatch value (stored in corrLagStats[2]) to identify which of the two audio signals is the target channel based on shift variations over successive frames. As a default the target channel is assigned a pointer into the buffer that holds the right audio signal. The target channel pointer is assigned a pointer into the buffer that holds the left audio signal if the previous non-causal shift is zero and the current amended mismatch value is negative.

The following pseudo-code implements the logic of the state diagram:

```
target = ptrChanR;
target_idx = R_CH_INDX;
if ( ( prevNCShift == 0 && hStereoTCA->corrLagStats[2] < 0 ) || ( hStereoTCA->prevRefChanIndx ==
R_CH_INDX ) )
{
 target = ptrChanL;
 target_idx = L_CH_INDX;
}
```

(c) Reference Channel Indicator (RCI)

The reference channel is based on the sign of currentNCShift, and effectively determines which channel is designated as the leading or lagging in time. The reference channel indicator is determined based on whether currentNCShift is positive or negative. The quantized reference channel indicator, refChanIndx (0/1), is reference in the pseudo-code below. If currentNCShift (which is updated to corrLagStats[2]) is greater than or equal to 0, the reference channel index (refChanIndx) is set to the left channel index (L_CH_INDX), i.e., the left channel is the reference channel. If currentNCShift is less than 0, the reference channel index(refChanIndx) is set to the right channel index (R_CH_INDX), i.e., the right channel is the reference channel. It should be noted that once the reference channel is determined, the other channel, the non-reference channel is the target channel. Adjustment of the target channel aligns the target channel with the reference channel.

The following pseudo-code implements the logic of the Reference Channel Indicator:

```
if (corrLagStats[2] >= 0)
   refChanIndx = L_CH_INDX;
else
   refChanIndx = R_CH_INDX;
```

(d) Compute Inter-Frame Shift Variation

Inter-Frame Variation Shift Analyzer determines the previous non-causal shift and current non-causal shift by taking the absolute value of the previous (final) mismatch value (prevcorrLagStats[2]) and current(final) mismatch values. The previous NC Shift value is determined by taking the absolute value of the previous CorrLagStats[2]. The current NCShift value is determined by taking the absolute value of the current mismatch value (CorrLagStats[2]). The variation, d, is the difference between two mismatch values, the current NC shift value, and the previous NC shift value. The variation, d, is implemented as follows:

$$d = -1 * (currentNCShift - previousNCShift) \qquad (5.3\text{-}50)$$

The Target Sample Adjuster determines whether to adjust the set of target samples based on the variation. When the difference between the currentNCShift and previousNCShift is not zero, the Target Sample Adjuster operates. The absolute value of the variation, i.e. |currentNCShift-previousNCShift| determines target signal adjustment for temporal shift variations. That absolute value of the variation is also used to select which adjustment technique based on a

comparison of the variation with a threshold. The threshold is the maximum shift change permittable and depends on the input sampling frequency. When the absolute value of the variation is less than the threshold, the Target Sample Adjuster operates in TD mode (CPE=TD mode), otherwise it operates in DFT mode (CPE=DFT mode). The final mismatch value (corrLagStats[2]) is sent to the gain parameter generator.

5.3.2.3.1.2.5 Target Sample Adjuster

The Target Sample Adjuster (a) updates the reference channel and target channel pointers based on the reference channel indicator; (b) interpolates the target samples in TD mode; and (c) interpolates the target samples in DFT mode.

(a) Updating the reference channel and target channel pointers

The sample adjuster determines whether to adjust the set of target samples based on a reference channel indicator. The Target Sample Adjuster updates the reference channel and target channel pointers based on the reference channel indicator. If the reference channel index (refChanIndx) is equal to the left channel index (L_CH_INDX), the left channel pointer is used to retrieve the reference signal in buffer2, and the right channel pointer is used to retrieve the target signal in buffer1.

The pseudo-code below implements the logic of the reference and target pointers update:

```
if ( refChanIndx == L_CH_INDX )
    { ref = ptrChanL;   target = ptrChanR; }
    else
    { ref = ptrChanR; target = ptrChanL; }
```

(b) Interpolation of Target Samples when CPE mode is TD Mode

For the current frame, the Target Sample Adjuster will shift the target channel signal by the current non-causal shift (currentNCShift) value to modify the adjusted target channel signal that is temporally aligned with the reference channel signal. When the CPE mode is in TD mode the Target Sample Adjuster uses one set of adjustment techniques based on a Sinc interpolation, followed by a LaGrange interpolation in the form of a cubic interpolation, to adjust the set of target samples to generate an adjusted set of target samples. Inter-frame variations in the final shift values result in temporal discontinuities at the frame boundaries in the calculation of the mid/side signals. This variation of the shift leads to either skipping or repeating of samples in the target channel at the frame boundaries. The skipping or repeating samples leads to discontinuities in the waveform of the generated mid and side signals that are manifested as audible clicks during playback. To compensate for the discontinuity, the Target Sample Adjuster reduces the samples between frames by slow shifting the target signal. To perform slow shifting during interpolation of the samples of the target signal the difference in samples between frames spreads out the discontinuity over multiple samples.

The Target Sample Adjuster interpolates a subset of target samples using an interpolation factor that is based on the variation and a spreading factor. The output of the interpolation represents estimated samples. The mismatch value may vary based on the characteristic of the type of signal. Consequently, the variation may also vary on the type of audio signal. The Target Sample Adjuster replaces the subset of the target samples with the set of estimated samples to generate an adjusted set of target samples. The adjusted set of target samples reduce the amount of discontinuity near boundaries of target frames associated with the target channel signal. The interpolation is performed by Sinc interpolation followed by a Lagrange interpolation implemented as a cubic interpolation. An interpolation factor is used for each of the interpolations. The interpolation_factor is expressed as follows:

$$interpolation\_factor = D/N\_SPREAD, \quad (5.3\text{-}51)$$

where, D is the difference in number of samples between the first mismatch value (current NC shift value) and the second mismatch value from the past frame (previous NC shift value), and N_SPREAD is the number over which the discontinuity is spread out. D is based on the different of the mismatch values between frames when the values of the mismatch vaues are obtained at the same sampling rate. The larger the value of N_SPREAD, the smaller the difference between each estimated sample. Reducing the temporal difference between samples, i.e., spreading the one sample difference over several samples of the second audio channel using the estimated samples reduces the discontinuity at the frame boundary. To increase the spread of the samples, the sinc interpolation uses subtraction and addition of a SINCORDER1 from i and then scales by SFACTOR1 (.5). The range of the spread of the samples is a lower limit, lim1, to an upper limit, lim2, where

$$lim1 = round\_up\,[(i - 24]) * SFACTOR1 \quad (5.3\text{-}52)$$

and

$$m2 = round\_down[(i + 24) * SFACTOR2]. \quad (5.3\text{-}53)$$

As i varies, the samples are spread over a range of approximately twenty (24) samples, twelve samples on either side of i, due to the SFACTOR being half (.5). Effectively, making the number of samples over which the discontinuity is spread out (N_SPREAD) 24 samples.

For the cubic interpolation, the SFACTOR2 is 2*|d|, making the number of samples over which the discontinuity is spread out (N_SPREAD) also 2*|d|.

(c) Interpolation of Target Samples when CPE mode is DFT Mode

In DFT mode, the Target Sample Adjuster uses an overlap and add interpolation adjustment technique. The overlap and add interpolation is based on a first window function and a second window function, wherein the second window function is dependent on the first window function. The final samples, of the interpolated target channel signal, are based on a weighted combination of the shift values in the first and second buffers. The final samples, of the interpolated target channel signal, are expressed as:

$$target_{final}(i) = win(i) \times target\ (i) + (1 - win(i)) \times target\ (i + d) \quad (5.3\text{-}54)$$

where, $win(i)$ is a window function which smoothly decreases from 1 to 0, and d is the variation.

The pseudo-code to implement the logic is as follows:

```
if (input_FS is > FS_16K) { L_shift adapt i= 596 else L_shift_adapt = 290}
winSlope = 1/L_shift_adapt,
for (i=0; I < L_shift_adapt; i++){
target[i] = target[i]*winSlope (1-winSlope)*target [i +d]
winSlope = winSlope + winSlope
}
```

The currentGain (see next Clause) is applied in-place to the target channel, so the modified target channel samples replace the original samples. The goal of this process is to minimize the perceptual impact of the non-causal shift on the encoded audio signal. After the target channel has been adjusted the memory is updated with the last input and output samples.

To maximally align the reference signal target signal generation of new audio data is performed. New audio data samples are generated from the reference signal,

$$target\,[i\ +\ currentNCShift]\ =\ targetGain * reference[i] \quad (5.3\text{-}55)$$

5.3.2.3.1.2.6 Gain Estimator

The current gain, g_est, is estimated by calculating a ratio of the sum of the absolute values of the samples in the initial buffers, LChanB and RChanB.

$$g\_est\ =\ \frac{\sum_{i=0}^{N-1}|RChanB(i+i1)|}{\sum_{i=0}^{N-1}|LChanB(i+i2)|} \quad (5.3\text{-}56)$$

where the value of i1 and i2 are controlled by the reference channel indicator.
The current gain is estimated based on the reference channel indicator (RCI).

The pseudo code is as follows:

```
if (refChanIndx == L_CH_INDX )
{ i1 = 0;  i2 = ncShift;}
else  { i1 = ncShift;  i2 = 0;  }
```

The current target gain, targetGain, is estimated based on the following equation that includes the previous target gain:

$$targetGain\ =\ (alpha)\ * log10(prevTargetGain\ )\ +\ (\ 1\ -\ alpha\ )\ *\ log10f(g\_est) \quad (5.3\text{-}57)$$

5.3.2.3.1.2.7 Quantization of ICA parameters

The current target gain (TARGET_GAIN), currentNCShift (NC_SHIFT_INDX), and reference channel indicator (REF _CH_INDX) are quantized, included in the ICA bitstream, and transmitted to the decoder.

5.3.2.3.1.2.8 Delay estimation

In TD stereo coding, delay estimation is determined by a cross-correlation coefficient of the current frame, a delay track estimation value of the current frame based on buffered inter-channel time difference (ITD) information of at least one past frame, an adaptive window function of the current frame. The inter-channel time difference is calculated based on a weighted cross-correlation coefficient which is obtained by the delay track estimation value and the adaptive window function.

The adaptive window function wherein the first raised cosine width parameter is obtained through calculation by using the following calculation formulas:

$$\text{win\_width} = \text{TRUNC}(\text{width\_par} * (A * \text{L\_NCSHIFT\_DS} + 1))$$

wherein win_width is the first raised cosine width parameter, TRUNC indicates a rounding value, L_NCSHIFT_DS is a maximum value of an absolute value of an inter-channel time difference, A is greater than or equal to 4.

width_par is defined as following formulas:

$$\text{width\_par} = \text{a\_width} * \text{smooth\_dist\_reg} + \text{b\_width}$$

$$\text{width\_par} = \min(\text{width\_par}, \text{xh\_width})$$

$$\text{width\_par} = \max(\text{width\_par}, \text{xl\_width})$$

wherein,

$$\text{a\_width} = (\text{xh\_width} - \text{xl\_width})/(\text{yh\_dist} - \text{yl\_dist})$$

$$\text{b\_width} = \text{xh\_width} - \text{a\_width1} * \text{yh\_dist}$$

xh_width is an upper limit value of the first raised cosine width parameter, xl_width is a lower limit value of the first raised cosine width parameter, yh_dist is a smoothed inter-channel time difference estimation deviation corresponding to the upper limit value of the first raised cosine width parameter, yl_dist is a smoothed inter-channel time difference estimation deviation corresponding to the lower limit value of the first raised cosine width parameter, smooth_dist_reg is the smoothed inter-channel time difference estimation deviation of the previous frame of the current frame, and xh_width, xl_width, yh_dist, and yl_dist are all positive numbers.

win_bias is the first raised cosine height bias which is obtained through calculation by using the following calculation formula:

$$\text{win\_bias} = \text{a\_bias} * \text{smooth\_dist\_reg} + \text{b\_bias}$$

$$\text{win\_bias} = \min(\text{win\_bias}, \text{xh\_bias})$$

$$\text{win\_bias} = \min(\text{win\_bias}, \text{xh\_bias})$$

wherein,

$$\text{a\_bias} = (\text{xh\_bias} - \text{xl\_bias})/(\text{yh\_dist} - \text{yl\_dist})$$

$$\text{b\_bias} = \text{xh\_bias} - \text{a\_bias} * \text{yh\_dist}$$

wherein win_bias is the first raised cosine height bias, xh_bias is an upper limit 20 value of the first raised cosine height bias, xl_bias is a lower limit value of the first raised cosine height bias, yh_dist is a smoothed inter-channel time difference estimation deviation corresponding to the upper limit value of the first raised cosine height bias, yl_dist is a smoothed inter-channel time difference estimation deviation corresponding to the lower limit value of the first raised cosine height bias, and yh_dist, yl_dist, xh_bias, and xl_bias are all positive numbers.

The adaptive window function loc_weight_win is represented by using the following formulas:

when $0 \le k \le$ TRUNC(A * L_NCSHIFT_DS/2) – 2 * win_width – 1,

$$\text{loc\_weight\_win}(k) = \text{win\_bias}$$

when TRUNC(A * L_NCSHIFT_DS/2) – 2 * win_width $\le k \le$ TRUNC(A * 10 L_NCSHIFT_DS/2) + 2 * win_width – 1,

$$\text{loc\_weight\_win(k)} = 0.5 * (1 + win_{bias}) + 0.5 * (1 - win\_bias) * \cos\left(\frac{\pi * \left(k - TRUNC\left(A * L_{NCSHIFT_{DS}}/2\right)\right)}{2 * win\_width}\right)$$

when TRUNC(A * L_NCSHIFT_DS/2) + 2 * win_width ≤ k ≤ A * L_NCSHIFT_DS,

$$\text{loc\_weight\_win(k)} = \text{win\_bias}$$

wherein k = 0, 1, ..., A * L_NCSHIFT_DS

corrEst (x) is the weighted cross-correlation coefficient which is obtained through calculation by using the following calculation formula:

$$\text{corrEst(k)} = \text{corrEst(k)} * \text{loc\_weight\_win(k)}$$

smooth_dist_reg_update is the smoothed inter-channel time difference estimation deviation of the current frame which is obtained through calculation by using the following calculation formulas:

$$\text{smooth\_dist\_reg\_update} = (1 - \gamma) * \text{smooth\_dist\_reg} + \gamma * \text{dist\_reg}'$$

$$\text{dist\_reg}' = |\text{reg\_prv\_corr} - \text{corrLagStats}[0]|$$

γ is a first smoothing factor, and 0 < γ < 1; smooth_dist_reg is the smoothed inter-channel time difference estimation deviation of the previous frame of the current frame; reg_prv_corr is the delay track estimation value of the current frame; and corrLagStats is the inter-channel time difference of the current frame.

5.3.2.3.2 TD stereo coder overview

The TD stereo coder is a low-rate low-complexity parametric stereo technology where the stereo input signal is encoded in time domain. It is especially effective for scenes where the correlation between the left and the right channel of the input audio signal is low, when the background noise is fluctuating a when an interfering talker is present. The classifier of uncorrelated input signals and the cross-talk detector, described in 5.3.2.5, are used to select frames suitable for the TD stereo coder. The TD stereo coder comprises a method for combining the left and the right channel of the input stereo signal to obtain the primary and the secondary channel. The bitrate is adaptively distributed between the primary and the secondary channel to maximize the quality of the encoded stereo signal. The secondary channel usually requires lower amount of bits due to leveraging some parameters from the the primary channel.

The TD stereo encoder is schematically depicted in Figure 5.3-18.

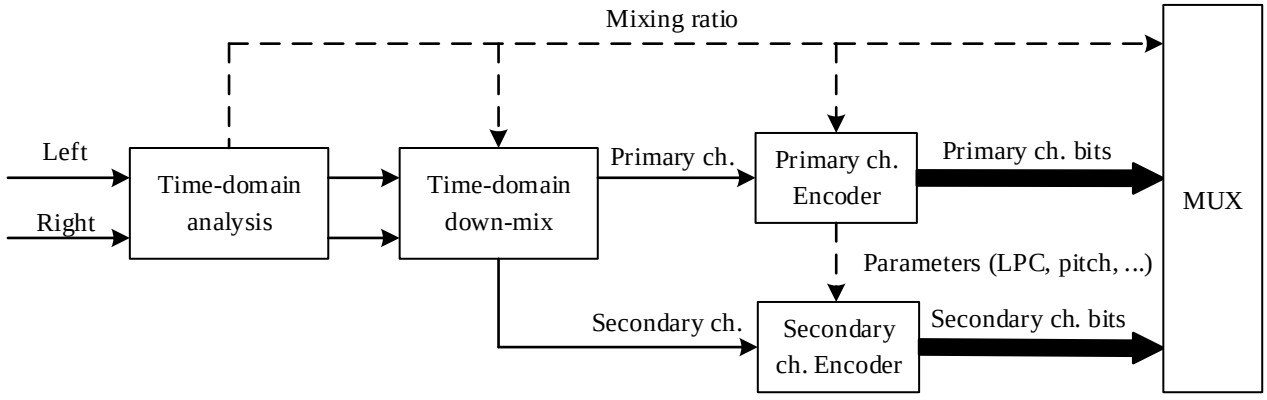


Figure 5.3-18: TD stereo encoder

The left channel and the right channel of the input stereo signal signal are mixed together such that a primary channel $Y(n)$ and a secondary channel $X(n)$ are formed in time domain. A minimum bitrate is allocaye for the encoding of the secondary channel and its associated bit budget varies on a frame-by-frame basis depending of the content. The bits that are not used in the encoding of the secondary channel are used in the encoding of the primary channel. The bit budget of the secondary channel is always less than the bit budget of the primary channel. The sum of the bitrates required by the primary and the secondary channel is equal to the total bitrate which is constant in every frame. The time-domain down-

mixing mechanism may be skipped in situations where the left channel and the right channel are incoherent, resulting in a Left-Right TD stereo submode (LRTD). In the LRTD submode the amount of bits in the primary channel and the secondary channel are close to equal. On the contrary, when a maximum emphasis is put on the primary channel, the bit budget of the secondary channel is aggressively forced to a minimum.

5.3.2.3.3 Time-domain stereo downmix

The TD stereo downmix converts the signals in the left channel and the right channel into primary channel and secondary channel. The functionality of the TD stereo downmix is schematically shown in Figure 5.3-19.

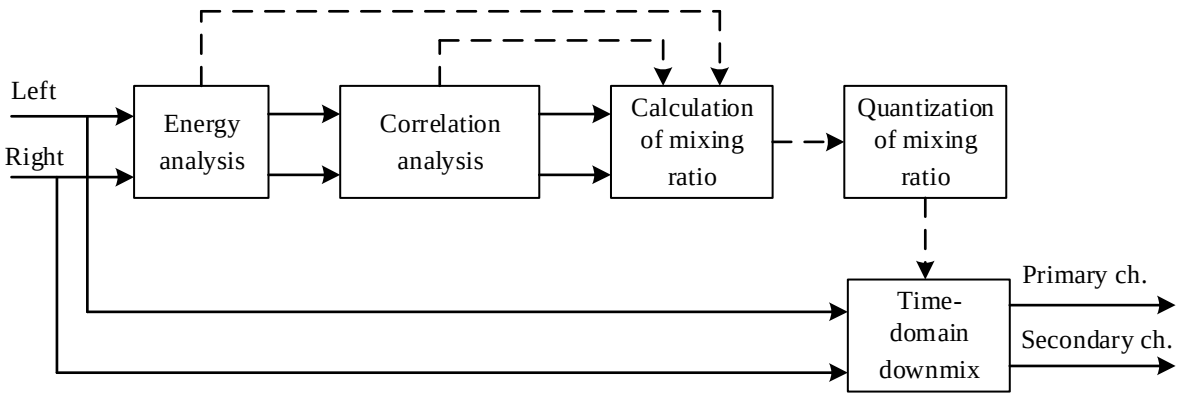


Figure 5.3-19: TD stereo downmix

The TD stereo downmix is based on the inter-channel (left-right) energy analysis. First, theThe TD stereo downmix is designed in such a way that the generated signals maintain their speech character allowing the ACELP codec to be efficiently applied on them. The TD stereo downmix is based on the inter-channel (left-right) energy analysis.

The passive mono downmix of the stereo input signal is calculated as

$$M(n) = \frac{L(n)+R(n)}{2} \tag{5.3-58}$$

and the side channel resulting from the passive mono downmix is calculated with

$$S(n) = \frac{L(n)-R(n)}{2} \tag{5.3-59}$$

The average energy of each input channel in the current frame is computed as

$$rms_L = \sqrt{\frac{\sum_{n=0}^{N-1} L(n)^2}{N}} \qquad rms_R = \sqrt{\frac{\sum_{n=0}^{N-1} R(n)^2}{N}} \tag{5.3-60}$$

where the subscripts L and R stand for the left channel and the right channel, respectively and N corresponds to the number of samples per frame. The average energy is then used to compute the long-term average energy of each channel as follows

$$\begin{aligned} \overline{rms}_L &= 0.6 \cdot \overline{rms}_L^{[-1]} + 0.4 \cdot rms_L \\ \overline{rms}_R &= 0.6 \cdot \overline{rms}_R^{[-1]} + 0.4 \cdot rms_R \end{aligned} \tag{5.3-61}$$

where the superscript $[-1]$ denotes the previous frame. The long-term average energy is then used to determine the energy trend in each channel as follows

$$\begin{aligned} \overline{rms\_dt}_L &= \overline{rms}_L - \overline{rms}_L^{[-1]} \\ \overline{rms\_dt}_R &= \overline{rms}_R - \overline{rms}_R^{[-1]} \end{aligned} \tag{5.3-62}$$

The energy trend contains the information whether temporal events captured in the input signals are fading out or if they are changing side. The long-term average energy and the energy trend are then used to determine the convergence speed of the long-term correlation difference as will be shown later.

The correlations of the left channel and the right channel against the passive mono downmix in the current frame are computed with

$$G_L = \frac{\sum_{n=0}^{N-1}(L(n)\cdot M(n))}{\sum_{i=0}^{N-1} M(n)^2}$$
$$G_R = \frac{\sum_{n=0}^{N-1}(R(n)\cdot M(n))}{\sum_{i=0}^{N-1} M(n)^2} \quad (5.3\text{-}63)$$

The correlations G_L and G_R are then smoothed with

$$\overline{G_L} = \alpha \cdot \bar{G}_L^{[-1]} + (1-\alpha)\cdot G_L$$
$$\overline{G_R} = \alpha \cdot \bar{G}_R^{[-1]} + (1-\alpha)\cdot G_R \quad (5.3\text{-}64)$$

where α is the convergence speed and the superscript $[-1]$ denotes the previous frame. Finally, the long-term correlation difference is calculated as

$$\overline{G_{LR}} = \overline{G_L} - \overline{G_R} \quad (5.3\text{-}65)$$

The convergence speed α can have the value of 0.8 or 0.5 depending on the long-term average energy computed in equation (5.3-61) and the energy trend computed in equation (5.3-62). The convergence speed α is set to the value of 0.8 if both long-term average energies are pointing in the same direction and the long-term correlation difference $\overline{G_{LR}}$ in the current frame and the long-term correlation difference $\bar{G}_{LR}^{[-1]}$ in the previous frame are both low and if at least one of the long-term average energies has a value above certain pre-defined threshold. This basically ensures that the energies in both channels are evolving smoothly and there is no abrupt change of energy from one channel to the other channel and at least one of the channels contains a meaningful amount of energy. Otherwise, when the long-term energies are pointing in opposite directions or when the long-term correlation difference $\overline{G_{LR}}$ is too high or if both input channels have a low energy, then the converegence speed α is set to the value of 0.5 to increase the adaptation speed of the long-term correlation $\overline{G_{LR}}$.

Once the long-term correlation difference is properly estimated it is converted into a mixing factor β which is then quantized with a scalar quantizer and the index is transmitted to the decoder in the bitstream. The mixing factor β represents two aspects of the stereo input combined in a single parameter. The mixing factor β represents the proportion of the input channels that are combined together to create the primary channel. The mixing factor β also represents an energy-scaling factor that, when applied to the primary channel, would result in a signal energetically close to the passive mono downmix. Thus, it allows the primary channel to be decoded as a mono signal without the need of side information (auxiliary stereo parameters). The mixing factor β may also be used to rescale the energy of the secondary channel before its encoding such that its energy is closer to the optimal energy range of the underlying encoder.

The mapping between the long-term correlation difference $\overline{G_{LR}}$ and the mixing factor β is done by limiting the long-term correlation difference to the interval of -1.5 to 1.5 and linearly scaling in the interval of 0.0 to 2.0. This is done as follows

$$G'_{LR} = \begin{cases} 0, & \overline{G_{LR}} \le -1.5 \\ \frac{2}{3}\cdot\overline{G_{LR}} + 1.0, & -1.5 < \overline{G_{LR}} < 1.5 \\ 2, & \overline{G_{LR}} \ge 1.5 \end{cases} \quad (5.3\text{-}66)$$

After the linearization, the long-term correlation difference G'_{LR} is converted to the mixing factor β using the cosine function. That is

$$\beta = \frac{1}{2}\cdot\left(1 - cos\left(\pi\cdot\frac{G'_{LR}}{2}\right)\right) \quad (5.3\text{-}67)$$

The TD stereo coder also calculates the instantaneous value of the mixing factor from the short-term correlation difference. That is

$$\beta_{inst} = \frac{1}{2}\cdot\left(1 - cos\left(\pi\cdot\frac{G''_{LR}}{2}\right)\right) \quad (5.3\text{-}68)$$

where

$$G''_{LR} = \begin{cases} 0, & \overline{\overline{G_{LR}}} \leq -1.5 \\ \frac{2}{3} \cdot \overline{\overline{G_{LR}}} + 1.0, & -1.5 < \overline{\overline{G_{LR}}} < 1.5 \\ 2, & \overline{\overline{G_{LR}}} \geq 1.5 \end{cases} \quad (5.3\text{-}69)$$

Note, that $\overline{\overline{G_{LR}}}$ is the short-term correlation difference which is calculated similarly as the long-term correlation difference $\overline{G_{LR}}$, i.e. through eqs. (5.3-64) and (5.3-65). The only difference is that the value of α is smaller, resulting in lower amount of smoothing and faster convergence.

The primary channel $Y(n)$ and the secondary channel $X(n)$ are then calculated as a weighted sum of the left channel and the right channel using the following YX stereo mixing scheme

$$\begin{aligned} Y(n) &= R(n) \cdot (1 - \beta) + L(n) \cdot \beta, \quad \text{for } n = 0, \ldots, N-1 \\ X(n) &= L(n) \cdot (1 - \beta) - R(n) \cdot \beta, \quad \text{for } n = 0, \ldots, N-1 \end{aligned} \quad (5.3\text{-}70)$$

For the purposes of the TD stereo coder the long-term correlation difference G'_{LR} is also converted to the energy-normalization factor ε using the following relation

$$\varepsilon = -\frac{1}{2}(G'_{LR} - 1)^2 + 1.0 \quad (5.3\text{-}71)$$

The mixing factor β is quantized with a 5-bit scalar quantizer and the resulting index β corresponds to one of the following coding schemes

$$\hat{\beta} = \begin{cases} 0 & \text{LR coding with } Y = R \text{ and } X = L \\ 1 \ldots 14 & \text{YX mixing} \\ 15 & \text{MS mixing} \\ 16 \ldots 29 & \text{YX mixing} \\ 30 & \text{LR coding with } Y = L \text{ and } X = -R \\ 31 & \text{skip downmix} \end{cases} \quad (5.3\text{-}72)$$

The index 0 corresponds to the Left/Right (LR) stereo coding scheme where the primary channel contains the exact copy of the right channel, and the secondary channel contains the exact copy of the left channel. Thus, there is no TD stereo downmix in this situation. For indices 1, ..., 14 and 16, ..., 29 the TD stereo downmix contains the weighted sum of the left channel and the right channel as calculated using equation (5.3-72). The index 15 corresponds to the Mid/Side (MS) stereo where the mid channel (which corresponds to the primary channel) is defined in equation (5.3-58) and the side channel (which corresponds to the secondary channel) is defined in equation (5.3-59). Thus, the MS mixing scheme is defined by equation (5.3-70) for $\beta = 0.5$. Finally, the index 30 corresponds to the situation where the primary channel contains the exact copy of the left channel, and the secondary channel contains the exact copy of the inverted phase right channel. The index 31 is reserved for special cases where the TD stereo downmix is skipped, for example when the IVAS coder switches from the MDCT stereo mode to the TD stereo mode.

When the mixing factor index changes from one frame to another, a fading-out/fading-in algorithm is applied to smooth the frame boundaries transition.

5.3.2.3.4 Near out-of-phase operation (NOOP)

5.3.2.3.4.1 NOOP signal detection

The TD downmix may have a problem if the signal in the left channel and the signal in the right channel are close to an opposite phase. In that case the passive mono downmix obtained by summing the left channel and the right channel would lead to canceling of certain harmonic components, making it unsuitable for encoding. To ensure that this problem does not occur the TD stereo coder contains a special sub-mode for input signals that are near-out-of-phase (NOOP). The NOOP decision process is based on the analysis of the passive mono downmix and the side signal defined in the previous clause and some other auxiliary parameters. The NOOP decision process is outlined below.

First, a preliminary out-of-phase signal detection is based on the energy difference between the side signal $S(n)$ defined in equation (5.3-59) and the passive mono downmix $M(n)$ defined in equation (5.3-58). The instantaneous side-to-mono energy difference is calculated as

$$S_m = 10 \log_{10}\left(\sqrt{\frac{\sum_{n=0}^{N-1} S(n)^2}{N}}\right) - 10 \log_{10}\left(\sqrt{\frac{\sum_{n=0}^{N-1} M(n)^2}{N}}\right) \quad (5.3\text{-}73)$$

Then, the long-term side-to-mono energy difference is calculated as

$$\overline{S_m} = \begin{cases} 0.9\overline{S_m}^{[-1]}, & \text{for INACTIVE content,} \\ 0.9\overline{S_m}^{[-1]} + 0.1S_m, & \text{otherwise} \end{cases} \quad (5.3\text{-}74)$$

where the superscript $[-1]$ indicates the previous frame. The content is considered INACTIVE when the VAD hangover counter, VAD_{hover_Y}, is different than 0.

In addition to the long-term side-to-mono energy difference $\overline{S_m}$ the last OL pitch correlation $C_{F|L}$ as defined in clause 5.1.10 of Reference [1] is calculated for both the primary channel and the secondary channel. The OL pitch correlation $C_{F|L}$ is then evaluated to decide whether the TD stereo coder should run in the NOOP sub-mode. Let $C_Y^{[-1]}$ denote the calculated maximum OL pitch correlation of the primary channel Y in the previous frame. Let $C_X^{[-1]}$ denote the calculated maximum OL pitch correlation of the secondary channel in the previous frame. A sub-optimality flag F_{sub} is then set to 1 based on the following criterion.

$$F_{sub} = \begin{cases} 1, & \text{if } (\overline{S_m} > 2.0 \text{ OR } S_m > 2.5) \text{ AND} \\ & C_Y^{[-1]} \in (0.85, \ldots, 0.92) \text{ AND } C_X^{[-1]} \in (0.85, \ldots, 0.92) \text{ AND} \\ & VAD_{hover\_Y} \leq 1 \text{ AND } VAD_{hover\_X} \leq 3 \\ 0, & \text{otherwise} \end{cases} \quad (5.3\text{-}75)$$

where VAD_{hover_Y} is the VAD hangover counter of the primary channel and VAD_{hover_X} is the VAD hangover counter of the secondary channel. The VAD hangover counters are calculated using the procedure defined in clause 5.1.10 of [3].

The sub-optimality flag F_{sub} indicates that the TD stereo coder is running in the NOOP sub-mode. When the sub-optimality flag F_{sub} is set to 0 the LRTD sub-mode is selected within the TD stereo coder.

To improve the stability of the NOOP decision a simple hangover logic is applied to the sub-optimality flag F_{sub}. The hangover logic uses the OL pitch stability measure calculated for the primary and the secondary channel as

$$\begin{aligned} T_{sta\_Y} &= \left|T_{OL\_Y}^{[1]} - T_{OL\_Y}^{[0]}\right| + \left|T_{OL\_Y}^{[2]} - T_{OL\_Y}^{[1]}\right| \\ T_{sta\_X} &= \left|T_{OL\_X}^{[1]} - T_{OL\_X}^{[0]}\right| + \left|T_{OL\_X}^{[2]} - T_{OL\_X}^{[1]}\right| \end{aligned} \quad (5.3\text{-}76)$$

The sub-optimality flag F_{sub} is set to 1 in the current frame if it was set to 1 in three consecutive previous frames and if the pitch stability is higher than a certain pre-defined threshold. That is

$$F_{sub} = \begin{cases} 1, & F_{sub}^{[-k]}\text{=1 for } k = 1,2,3 \text{ AND} \\ & T_{sta\_Y} > 64 \text{ AND } T_{sta\_X} > 64 \\ 0, & \text{otherwise} \end{cases} \quad (5.3\text{-}77)$$

where $T_{OL}^{[k]}$ for $k = 1,2,3$ is the OL pitch parameter of the *k*th subframe calculated using the OL pitch analysis specified in clause 5.1.10 of [3]. The complete procedure of NOOP detection is outlined in the schematic diagram in Figure 5.3-20.

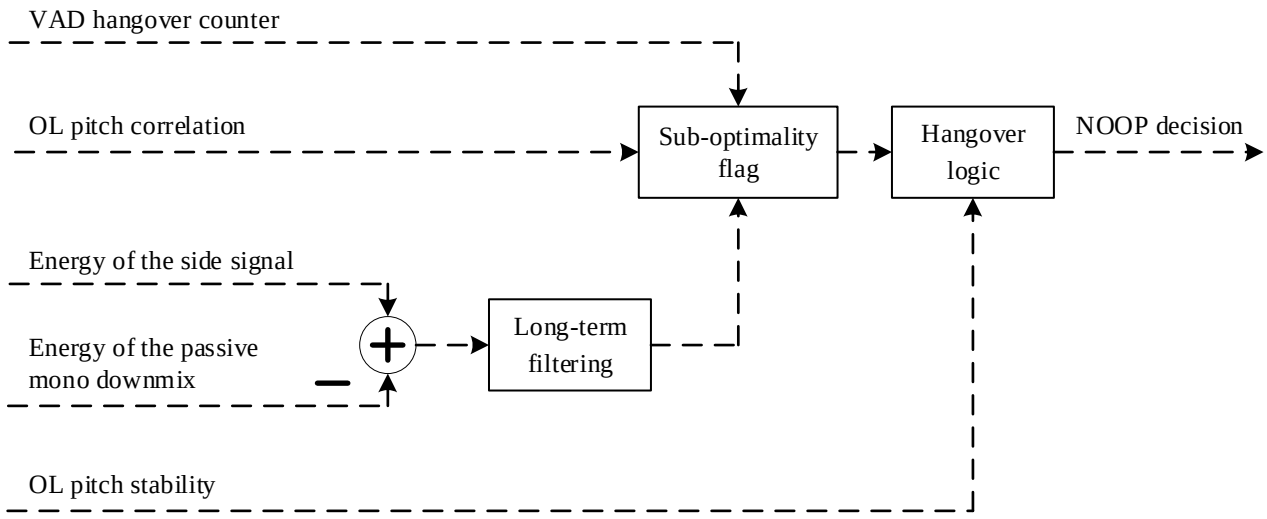


Figure 5.3-20: NOOP detection in the TD stereo mode

5.3.2.3.4.2 NOOP sub-mode selection

Following the NOOP signal detection described in clause 5.3.2.3.4.1 the selection of the NOOP sub-mode within the TD stereo coder is made using a series of conditions based on the signal classification of the primary and the secondary channel and the output of the NOOP signal detection block described in Fig. 5.3-20. Let the detected NOOP signal from Fig. 5.3-20 be denoted with a binary flag f_{NOOP}. Let the initial selection of the NOOP sub-mode be defined with the binary parameter $NOOP_{init}$ and the final selection of the NOOP sub-mode with the binary parameter $NOOP_{final}$. Note, that all binary parameters have either the value of 1 or the value of 0.

The initial selection of the NOOP sub-mode is made as follows

$$NOOP_{init} = \begin{cases} 0, & \text{if } f_{NOOP} = 0 \text{ AND } NOOP_{final}^{[-1]} = 0 \\ 1, & \text{if } f_{NOOP} = 1 \text{ AND } NOOP_{final}^{[-1]} = 1 \\ 0, & \text{if } f_{NOOP} = 0 \text{ AND } NOOP_{final}^{[-1]} = 1 \text{ AND } (rms_L < \Delta 1 \text{ AND } rms_r < \Delta 1) \\ 1, & \text{if } f_{NOOP} = 0 \text{ AND } NOOP_{final}^{[-1]} = 1 \text{ AND } (rms_L \geq \Delta 2 \text{ OR } rms_r \geq \Delta 2) \\ 1, & \text{if } f_{NOOP} = 1 \text{ AND } NOOP_{final}^{[-1]} = 0 \text{ AND } (rms_L < \Delta 1 \text{ AND } rms_r < \Delta 1) \\ 0, & \text{if } f_{NOOP} = 1 \text{ AND } NOOP_{final}^{[-1]} = 0 \text{ AND } (rms_L \geq \Delta 2 \text{ OR } rms_r \geq \Delta 2) \end{cases}$$

where [-1] indicates the value of the binary parameter in the previous frame.

The final selection of the NOOP sub-mode is then performed based on the initial selection of the NOOP sub-mode and a modification flag related to the NOOP-specific mixing ratio, f_{mod_SM} . The modification flag f_{mod_SM} indicates that the NOOP-specific mixing ratio β_{SM} needs or does not need to be modified. If the modification flag in the previous frame indicates that the mixing ratio needs to be modified, then the final selection of the NOOP sub-mode is set to 0, i.e. $NOOP_{final} = 0$. If the modification flag of the previous frame indicates that the mixing ratio does not need to be modified, then the final selection of the NOOP sub-mode is set based on a series of conditions involving the signal classification of the primary and the secondary channel. The following conditions are specified for the final selection of the NOOP sub-mode:

Condition1: the frame type of a primary channel signal in a previous frame (tdm_SM_last_clas) is UNVOICED_CLAS and its previous frame (tdm_SM_last2_clas) is VOICED_TRANSITION, or frame type of a secondary channel signal in a previous frame (tdm_SM_last_clas) is UNVOICED_CLAS and its previous frame (tdm_SM_last2_clas) is VOICED_TRANSITION;

Condition2: neither of the primary channel signal and the secondary channel signal in the previous frame (last_coder_type_raw) is a coding type corresponding to VOICED;

Condition3: a quantity of consecutive frames before the previous frame (tdm_NOOP_cnt) that use the channel combination scheme used by the previous frame is greater than 5;

Condition4: the frame type of the primary channel signal in the previous frame (tdm_SM_last_clas) is UNVOICED_CLAS, or the frame type of the secondary channel signal in the previous frame (tdm_SM_last_clas) is UNVOICED_CLAS;

Condition5: long-term root mean square energy values of the left channel (rms_L) and right channel (rms_R) are less than 400;

If Condition 1, Condition 2 and Condition 3 are all satisified at the same time, then the final selection of the NOOP sub-mode is set to 1. i.e. $NOOP_{final} = 1$. Alternatively, if Condition 2, Condition 3, Condition 4 and Condition 5 are all satisfied at the same time, then the final selection of the NOOP sub-mode is set to 1. i.e. $NOOP_{final} = 1$.

5.3.2.3.4.3 NOOP signal coding

When operating in the NOOP sub-mode, the TD stereo coder modifies the TD stereo downmix described in clause 5.3.2.3.3. In this mode, the left and right channels of the input stereo signal are combined using a NOOP-specific mixing factor, incorporating fade-in and fade-out smoothing techniques.

The primary and secondary channel signals in the current frame are calculated as follows:

$$\text{if} \quad 0 \le n < N - delay_{com}$$

$$\begin{bmatrix} Y(n) \\ X(n) \end{bmatrix} = M_{12} * \begin{bmatrix} X_L(n) \\ X_R(n) \end{bmatrix}$$

$$\text{if} \quad N - delay_{com} \le n < N - delay_{com} + NOVA\_1$$

$$\begin{bmatrix} Y(n) \\ X(n) \end{bmatrix} = fade\_out(n) * M_{12} * \begin{bmatrix} X_L(n) \\ X_R(n) \end{bmatrix} + fade\_in(n) * M_{22} * \begin{bmatrix} X_L(n) \\ X_R(n) \end{bmatrix}$$

$$\text{if} \quad N - delay_{com} + NOVA\_1 \le n < N$$

$$\begin{bmatrix} Y(n) \\ X(n) \end{bmatrix} = M_{22} * \begin{bmatrix} X_L(n) \\ X_R(n) \end{bmatrix}$$

$$fade\_in(n) = \frac{n - (N - delay_{com})}{NOVA\_1}$$

$$fade\_out(n) = 1 - \frac{n - (N - delay_{com})}{NOVA\_1}$$

where $n = 0,1, \ldots, N-1$ is the sample index, $fade\_in(n)$ indicates a fade-in factor and $fade\_out(n)$ indicates a fade-out factor. Furthermore, $NOVA\_1$ indicates a transition processing length which can be set to 0. The signal $X_L(n)$ indicates the left channel signal in the current frame, $X_R(n)$ indicates the right channel signal in the current frame, $Y(n)$ indicates the primary channel signal in the current frame that is obtained through the time-domain processing, and $X(n)$ indicates the secondary channel signal that is in the current frame and that is obtained through the time-domain processing. The parameter $delay_{com}$ indicates encoding delay compensation which can be set to 0. The matrix M_{22} is the downmixing matrix corresponding to the mixing ratio for the NOOP signal in the current frame and M_{12} is the downmixing matrix corresponding to the mixing ratio for the NOOP signal in the previous frame. The mixing ratio for the NOOP signal in the current frame is calculated based on the difference of two correlation measures between the left and right channel and the passive mono downmix. The downmixing matrices are defined as follows

$$M_{12} = \begin{bmatrix} \alpha_{1\_pre} & -\alpha_{2\_pre} \\ -\alpha_{2\_pre} & -\alpha_{1\_pre} \end{bmatrix}, M_{22} = \begin{bmatrix} \alpha_1 & -\alpha_2 \\ -\alpha_2 & -\alpha_1 \end{bmatrix}$$

where $\alpha_{1\_pre} = \beta_{SM}(n-1)$ and $\alpha_{2\_pre} = 1 - \beta_{SM}(n-1)$. The factor $\beta_{SM}(n-1)$ is the mixing ratio corresponding to the NOOP signal in the previous frame. Furthermore, $\alpha_1 = \beta_{SM}(n)$ and $\alpha_2 = 1 - \beta_{SM}(n)$ and $\beta_{SM}(n)$ is the mixing ratio corresponding to the NOOP signal in the current frame. The calculation of the mixing ratio corresponding top the NOOP signal is described in the next clause.

To obtain the primary channel signal and the secondary channel signal in the TD stereo mode for the NOOP signal, a segmented time-domain downmix process is performed based on the channel combination scheme for the current frame

and the channel combination scheme in the previous frame when the channel combination scheme for the current frame is different from the channel combination scheme for the previous frame. The channel combination scheme for the previous frame is the correlated signal channel combination scheme, and the channel combination scheme for the current frame is the anticorrelated signal channel combination scheme. To obtain the start/first middle segments of the primary and secondary channel signals, time-domain downmix processing on the start/first middle segments is performed by using the mixing ratio corresponding to the correlated signal channel combination scheme for the previous frame and a time-domain downmix processing manner corresponding to the correlated signal channel combination scheme for the previous frame. To obtain the end/second middle segments of the primary and secondary channel signals, time-domain downmix processing on the end/second middle segments is performed by using the mixing ratio corresponding to the anticorrelated signal channel combination scheme for the previous frame and a time-domain downmix processing manner corresponding to the anticorrelated signal channel combination scheme for the previous frame. To obtain the middle segments of the primary and secondary channel signals, a weighted summation processing on the first middle segments and the second middle segments is performed. The segments can be calculated according to

$$\begin{bmatrix} Y(n) \\ X(n) \end{bmatrix} = \begin{cases} \begin{bmatrix} Y_{11}(n) \\ X_{11}(n) \end{bmatrix}, & if \quad 0 \le n < N_1 \\ \begin{bmatrix} Y_{21}(n) \\ X_{21}(n) \end{bmatrix}, & if \quad N_1 \le n < N_2 \\ \begin{bmatrix} Y_{31}(n) \\ X_{31}(n) \end{bmatrix}, & if \quad N_2 \le n < N \end{cases}$$

wherein $X_{11}(n)$ indicates the start segment of the primary channel signal, $Y_{11}(n)$ indicates the start segment of the secondary channel signal, $X_{31}(n)$ indicates the end segment of the primary channel signal, $Y_{31}(n)$ indicates the end segment of the secondary channel signal, $X_{21}(n)$ indicates the middle segment of the primary channel signal, and $Y_{21}(n)$ indicates the middle segment of the secondary channel signal; $N_1 \ge 0$. $X(n)$ indicates the primary channel signal; $Y(n)$ indicates the secondary channel signal.

$$\begin{bmatrix} Y_{21}(n) \\ X_{21}(n) \end{bmatrix} = \begin{bmatrix} Y_{211}(n) \\ X_{211}(n) \end{bmatrix} * fade\_out(n) + \begin{bmatrix} Y_{212}(n) \\ X_{212}(n) \end{bmatrix} * fade\_in(n)$$

$fade\_in(n)$ indicates the fade-in factor, $fade\_out(n)$ indicates the fade-out factor, and a sum of $fade\_in(n)$ and $fade\_out(n)$ is 1; n indicates a sampling point number, and $n = 0,1,\cdots,N-1$, $N_1<N_2<N-1$; $X_{211}(n)$ indicates the first middle segment of the primary channel signal, $Y_{211}(n)$ indicates the first middle segment of the secondary channel signal, $X_{212}(n)$ indicates the second middle segment of the primary channel signal, and $Y_{212}(n)$ indicates the second middle segment of the secondary channel signal:

$$fade\_in(n) = \frac{n - N_1}{N_2 - N_1}$$

$$fade\_out(n) = 1 - \frac{n-N_1}{N_2-N_1}$$

5.3.2.3.4.4 Adaptive mixing ratio for the NOOP signal

The NOOP is a stereo signal whose phase difference between the left channel signal and the right channel signal falls within [180°-θ, 180°+θ], where θ is an angle between 0° and 90°. The mixing ratio for the NOOP signal is calculated using the same procedure as outlined in clause 5.3.2.3.3 with the notable difference that the passive mono downmix defined in eq. (5.3-58) is calculated in the following way

$$M(n) = \frac{L(n) - R(n)}{2}$$

and the side channel resulting from the passive mono downmix defined in eq. (5.3-59) is calculated as follows

$$S(n) = \frac{L(n) + R(n)}{2}$$

The mixing ratio for the NOOP signal is calculated based on the difference of two correlation measures, G_L and G_R defined by eq. (5.3-63). Note, that the calculation of the auxiliary parameters is done by following eqs. (5.3-60) to (5.3-52) using the mono downmix and the side channel as defined above.

The correlation measures G_L and G_R are then smoothed similarly as in eq. (5.3-64). That is

$$\bar{G}_L = \alpha_{SM} \cdot \bar{G}_L^{[-1]} + (1 - \alpha_{SM}) \cdot G_L$$
$$\bar{G}_R = \alpha_{SM} \cdot \bar{G}_R^{[-1]} + (1 - \alpha_{SM}) \cdot G_R$$

where α_{SM} is the convergence speed. Note, that α_{SM} has the same meaning as the convergence speed α used in eq. (5.3-64) but it's set specifically to the NOOP sub-mode. Finally, the long-term correlation difference is calculated as

$$\overline{G_{LR}} = \overline{G_L} - \overline{G_R}$$

i.e. similarly to eq. (5.3-65). The long-term correlation difference is then limited to the interval between -1.5 and +1.5. This is done as follows

$$\overline{G_{LR}} \leftarrow \begin{cases} +1.5 & \text{if } \overline{G_{LR}} > 1.5 \\ -1.5 & \text{if } \overline{G_{LR}} < -1.5 \\ \overline{G_{LR}} & \text{otherwise} \end{cases}$$

The long-term correlation difference is linearly scaled using piece-wise linear mapping. This is done as follows:

$$G'_{LR} = \begin{cases} 1.08 \cdot \overline{G_{LR}} + 0.38 & \text{if } \overline{G_{LR}} > 0.75 \\ 0.64 \cdot \overline{G_{LR}} + 1.28 & \text{if } \overline{G_{LR}} < -0.75 \\ 0.26 \cdot \overline{G_{LR}} + 0.995 & \text{otherwise} \end{cases}$$

After the linearization, the long-term correlation difference G'_{LR} is converted to the NOOP-specific mixing factor β_{SM} using the cosine function. That is

$$\beta_{SM} = \frac{1 - cos\left(\frac{\pi}{2} \cdot G'_{LR}\right)}{2}$$

5.3.2.3.5 LP filter coherence

The ACELP encoder in the secondary channel is optimized in such a way that it requires only a minimal amount of bits. To reduce the required amount of bits in the secondary channel certain parameters from the primary channel may be reused such as the LP filter or the OL pitch lag. First, a low-complexity pre-processing is performed on the signal in the secondary channel including the VAD, described in clause 5.2.2.2.5, LP analysis, described in clause 5.2.2.2.8, and OL pitch analysis, described in clause 5.2.2.2.9. Then, the characteristics of the secondary channel are analyzed and the signal is classified as GENERIC, UNOVICED or INACTIVE. The signal classification is done using the mechanism described in clause 5.2.2.3.1.2.

An important part of the bitrate consumption in the secondary channel is the quantization of LP filter coefficients. Given that the spectral content of the secondary channel is often close to that of the primary channel the TD stereo coder may decide on reusing the LP filter coefficients of the primary channel in the secondary channel. The TD stereo coder performs LP analysis as described in clause 5.2.2.2.8 on both the primary channel $Y(n)$ and the seconadry channel $X(n)$. Let's denote the LP filter coefficients of the primary channel as $A_Y(i)$, where $i = 0, \dots, M$ and M is the order of the LP filter. Using the same symbols, let's denote the LP filter coefficients of the secondary channel as $A_X(i)$. The TD stereo coder calculates two versions of LP residuals for the secondary channel. The first version of the LP residual is calculated by filtering the signal in the secondary channel with the LP filter of the primary channel. That is

$$r_Y(n) = \mathrm{X}(n) + \sum_{i=1}^{\mathrm{M}} \big(A_Y(i) \cdot \mathrm{X}(n-i)\big), \qquad n = 0, \dots, N-1 \tag{5.3-78}$$

The second version of the of the LP residual is calculated by filtering the signal in the secondary channel with the LP filter of the secondary channel. That is

$$r_X(n) = \mathrm{X}(n) + \sum_{i=1}^{\mathrm{M}} \big(A_X(i) \cdot \mathrm{X}(n-i)\big), \qquad n = 0, \dots, N-1 \tag{5.3-79}$$

The absolute energy of the secondary channel is calculated using the following relation

$$E_X = 10 \cdot log_{10}\left(\sum_{n=0}^{N-1} X(n)^2\right) \tag{5.3-80}$$

The TD stereo coder calculates the energy of the first version of the LP residual signal $r_Y(n)$ as

$$E_{rY} = 10 \cdot log_{10}\left(\sum_{n=0}^{N-1} r_Y(n)^2\right) \tag{5.3-81}$$

and the energy of the second version of the LP residual signal $r_X(n)$ as

$$E_{rX} = 10 \cdot log_{10}(\sum_{n=0}^{N-1} r_X(n)^2) \quad (5.3\text{-}82)$$

The TD stereo coder then calculates the LP prediction gains for both the first and the second version of the LP residual signal in the secondary channel by subtracting the energies of LP residuals from the absolute energy of the secondary channel. That is

$$\begin{aligned} G_Y &= E_X - E_{rY} \\ G_X &= E_X - E_{rX} \end{aligned} \quad (5.3\text{-}83)$$

Finally, the TD stereo coder calculates a ratio of gains $G_{Y|X}$ using the following relation

$$G_{Y|X} = \frac{G_Y}{G_X} \quad (5.3\text{-}84)$$

The decision on reusing the LP filter coefficients from the primary channel is also based on LP filter similarity between the primary and the secondary channel. The TD stereo coder calculates the similarity of LP filters by measuring the Euclidean distance between the line spectral pairs (LSPs) of the primary and the secondary channel. The LSPs of the primary and the secondary channel are converted from the respective LP filter coefficients using the procedure described in clause 5.1.9.5 of [3]. Let's denote the LSPs of the primary channel as $lsp_Y(i)$ where $i = 0, \ldots, M$ and M is the order of the LP filter. Similarly, let's denote the LSPs of the secondary channel as $lsp_X(i)$. The LSPs $lsp_Y(i)$ and $lsp_X(i)$ are weighted in such a way that emphasis is put on certain critical parts of the frequency spectrum. Then, the Eucidean distance between the LSPs is calculated with

$$dist = \sum_{i=0}^{M-1}\big(lsp_Y(i) - lsp_X(i)\big)^2 \quad (5.3\text{-}85)$$

Then, a series of conditions is performed to decide if the LP filter of the primary channel is reused during the encoding of the secondary channel. This is illustrated in Figure 5.3-21. In all cases, the transition frame counter, TC_{cnt}, defined as i_{TC} in clause 5.1.13.4 of [3], must be equal to 0 as it is better to use the LP filter of the secondary channel in case when signal transition is encountered.

The LP filter of the primary channel is reused when:

- the energy of the secondary channel E_X is below or equal to 30.0, or
- the ratio of gains $G_{Y|X}$ is higher than the pre-defined threshold τ, typically 0.92, and the LP filter similarity measured by the Euclidean distance $dist$ is lower than the threshold σ, typically 0.04, or
- the prediction gain of the secondary channel G_X is below the threshold of 3.0 and the Euclidean distance $dist$ is lower than 0.08, or
- the down-mixing ratio $\beta^{[-1]}$ in the previous frame is between 0.4 and 0.6 and the prediction ratio $G_{Y|X}$ is higher or equal to 0.86 and either the Euclidean distance $dist$ is below 0.08 or the prediction gain of the secondary channel G_X is below 3.0.

Furthermore, the LP filter of the primary channel is also reused in situations when the bitrate of the encoder is sufficiently high or if the input signal is considered as uncorrelated. The decision is based on the following set of, more restrictive, conditions:

- when the available bit rate is higher or equal to 48000 bps and if the ratio $G_{Y|X}$ is higher than 0.94, and the LP filter similarity measured by the Euclidean distance $dist$ is lower than the pre-defined threshold σ, typically 0.04, or
- when LRTD sub-mode has been selected and the available bit rate is lower than 16400 bps and the ratio of gains $G_{Y|X}$ is higher than 0.94, and the LP filter similarity measured by the Euclidean distance $dist$ is lower than 0.02, or
- when LRTD sub-mode has been selected and the available bit rate is higher or equal to 16400 bps and the ratio $G_{Y|X}$ is higher than 0.96, and the LP filter similarity measured by the Euclidean distance $dist$ is lower than 0.01.

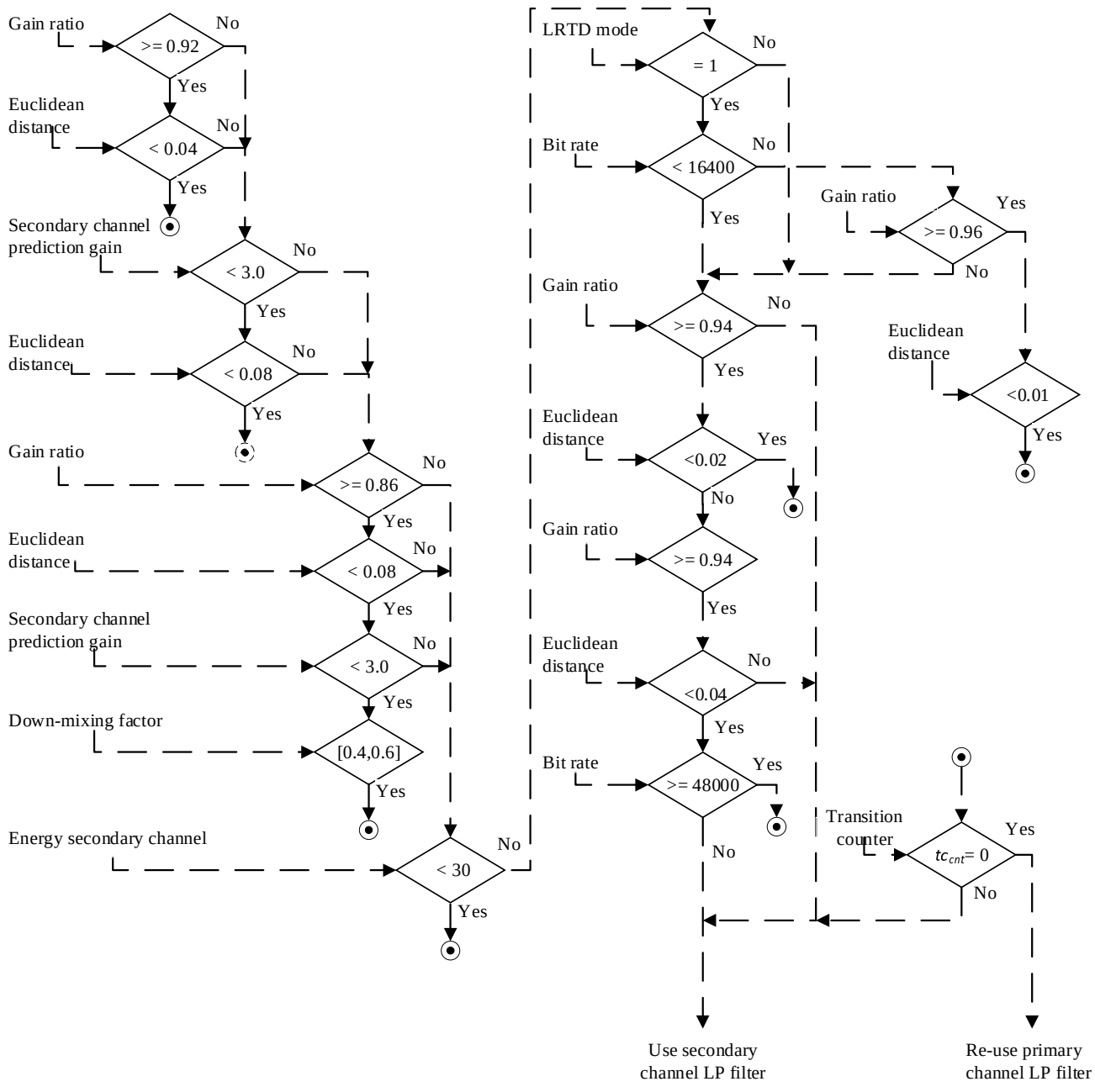


Figure 5.3-21: Re-using LP filter from the primary channel

In case the decision about re-using the LP filter from the primary channel is negative, then the encoder of the secondary channel will use the LP filter $A_X(i)$ for quantization and in further processing. As a consequence, the secondary channel will need to allocate more bits to transmit the information about the LP filter to the decoder. The decision logic on re-using the LP filter from the primary channel is depiced in the schematic diagram in Figure 5.3-21.

5.3.2.3.6 Open-loop pitch coherence

The TD stereo coder analyzes the pitch coherence between the primary and the secondary channel. The coherence analysis is based on the open-loop pitch, T_{OL}, calculated as described in clause 5.2.2.2.9. When the LRTD sub-mode is selected within the TD stereo coder the OL pitch coherence is calculated as

$$S_{pc} = \sum_{i=0}^{2} \left| T_{OL\_Y}(i) - T_{OL\_X}(i) \right| \quad (5.3\text{-}86)$$

where $T_{OL\_Y}(i)$ is the OL pitch of the primary channel in the ith half-frame and $T_{OL\_X}(i)$ is the OL pitch of the secondary channel in the ith half-frame. Otherwise, the OL pitch coherence is calculated as

$$S_{pc} = \left|\sum_{i=0}^{2} T_{OL\_Y}(i) - \sum_{i=0}^{2} T_{OL\_X}(i)\right| \quad (5.3\text{-}87)$$

If the OL pitch coherence is below a predetermined threshold Δ the fractional pitch lag from the primary channel, as described in clause 5.2.3.1.4.1 of [3], may be re-used in the adaptive codebook search of the secondary channel. The re-using of the fractional pitch lag from the primary channel is further conditioned on the available bit budget of the secondary channel and on the coder type of both the primary and the secondary channel. The decision upon re-using the fractional pitch lag from the primary channel is described in Figure 5.3-22.

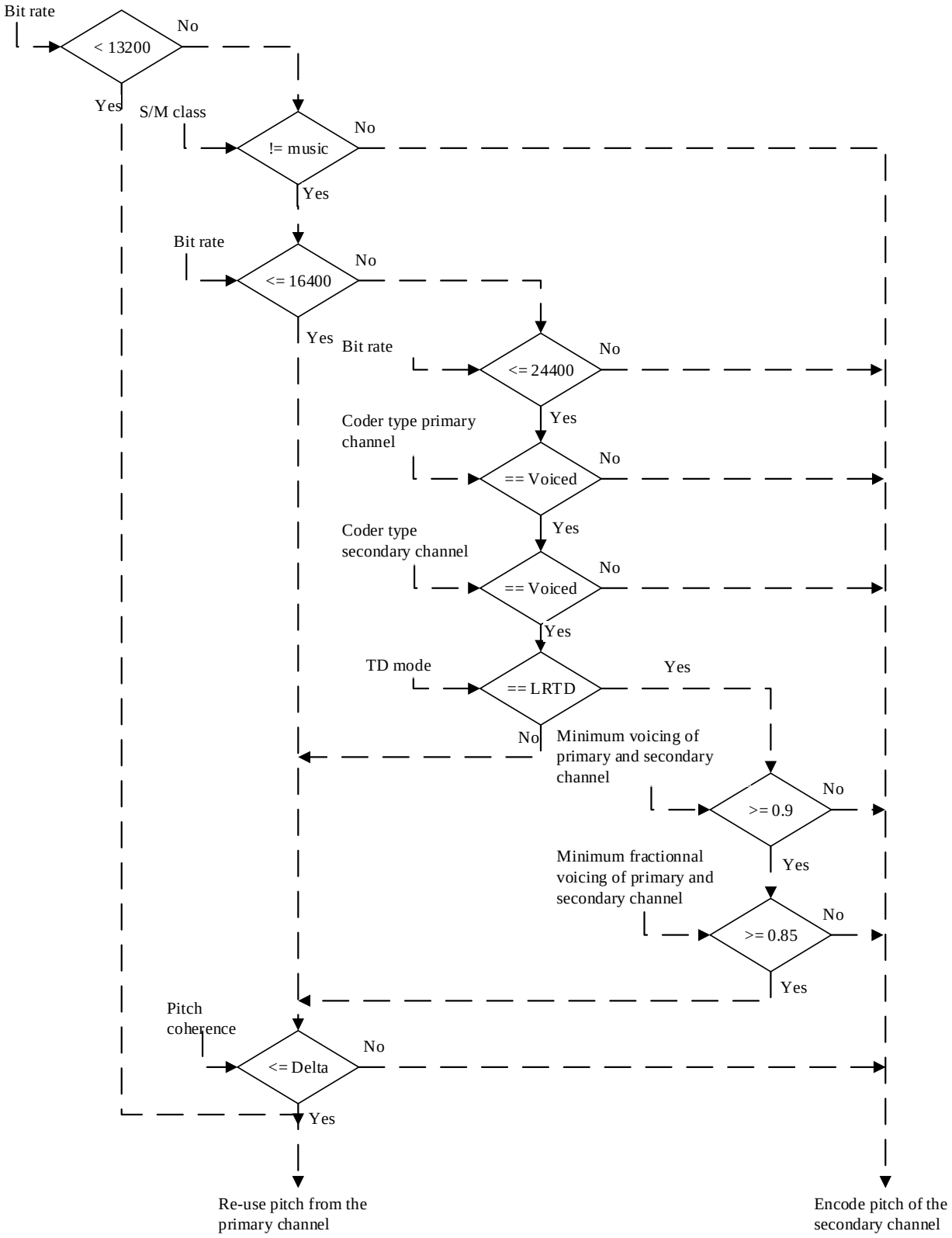


Figure 5.3-22: Re-using fractional pitch lag from the primary channel

5.3.2.3.7 Excitation coding in the secondary channel using two or four subframes

The TD stereo coder adaptively allocates bits among the primary channel and the secondary channel, or among the left channel and the right channel in case the down-mixing ratio β is either 0 or 1. The schematic block diagram in Figure 5.3-23. shows that the ACELP encoder in the secondary channel varies between a 4-subframe and 2-subframe scheme depending on the following parameters

- LP filter coherence between the primary channel and the secondary channel
- re-usage of the OL pitch information
- total bitrate

The primary channel is encoded using the ACELP/TCX encoder as described in clause 5.2.2.3.2. Note, that the ACELP encoder of the primary channel supports the encoding of all coder types.

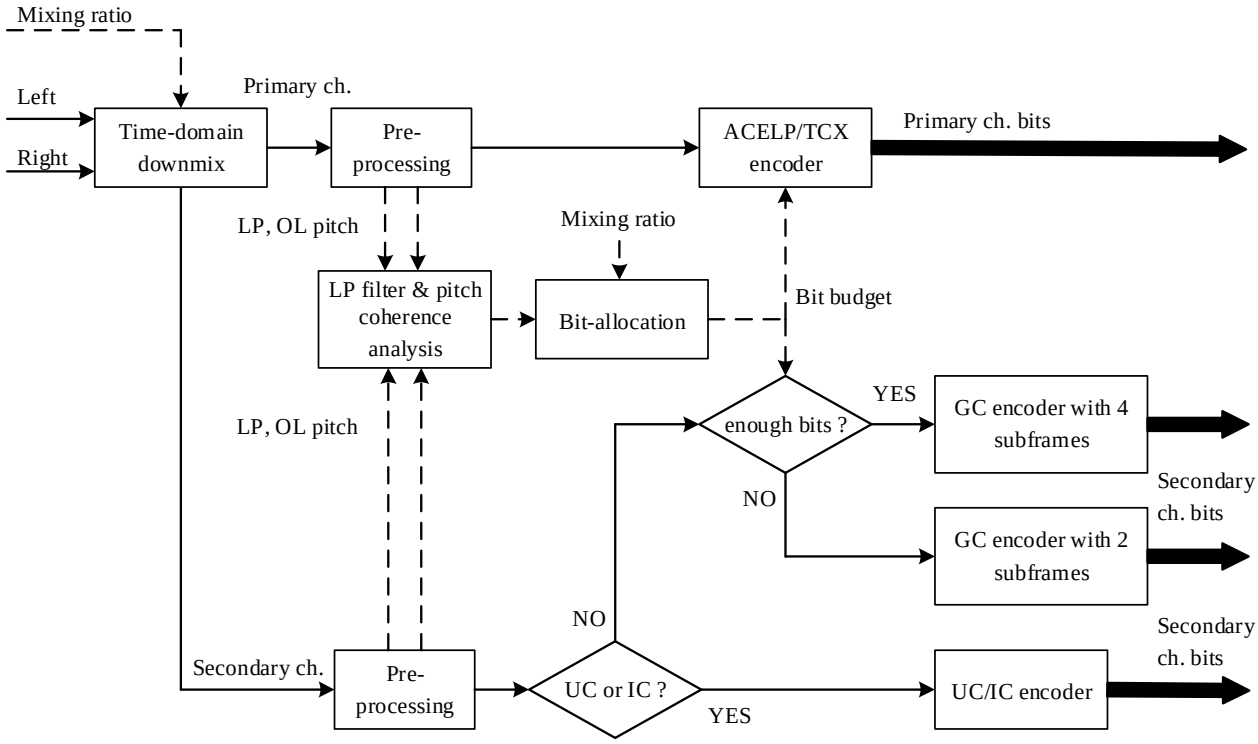


Figure 5.3-23: Dynamic encoding of the secondary channel in TD stereo mode

When the signal in the secondary channel is classified as GENERIC and once the possibility of re-using the LP filter and/or the OL pitch information from the primary channel is taken into account and the bit budget remaining for the encoding of the secondary channel is high enough, then the ACELP encoder as described in clause is used with four subframes per frame. Otherwise, the ACELP encoder assumes only two subframes per frame. The encoding of UNVOICED and INACTIVE coder types is done as described in clause 5.2.2.3.2.

The decision on using four or two subframes per frame in the GENERIC coder type depends on the bit budget allocated to the TD stereo coder and whether there is a sufficient number of bits available in the secondary channel. To encode the signal in the secondary channel with the four subframes per frame, at least 40 bits must be available to encode the indices of the algebraic codebooks. The number of bits available for the algebraic codebooks are calculated by subtracting the bitrate of all other quantized parameters incl. e.g. the LP filter, pitch or gains from the bitrate allocated to the secondary channel.

If the ACELP encoder uses two subframes per frame the excitation encoding of the secondary channel is performed as specified in 5.2.3.1 of [3] with the exception that the length of each subframe is changed from 64 samples to 128 samples. The interpolation of LP filter coefficients in the secondary channel is done as described in 5.1.9.6 of [3] with the following modification. Using the notation in clause 5.1.9.6 of [3] the interpolation of LP filter coefficients in the secondary channel is done as

$$q_i^{[0]} = 0.25q_{end-1,i} + 0.75q_{mid,i}$$
$$q_i^{[1]} = 0.25q_{mid,i} + 0.75q_{end,i} \quad (5.3\text{-}88)$$

for $i = 1, \ldots, 16$, where $q_i^{[0]}$ denotes the LP filter coefficients in the first subframe and $q_i^{[1]}$ denotes the LP filter coefficients in the second subframe. Note, that $q_{mid,i}$ and $q_{end,i}$ are estimated on the input signal of the secondary channel. Thus, the LP filter coefficients from the primary channel are not re-used in case when equation (5.3-88) is

applied. In case when LP filter coefficients of the primary channel are re-used in the secondary channel the mid-frame LP coefficients, $q_{mid,i}$, are not used for interpolation. That is

$$q_i^{[0]} = 0.5q_{end-1,i} + 0.5q_{Y\_end,i}$$
$$q_i^{[1]} = q_{Y\_end,i} \quad (5.3\text{-}89)$$

for $i = 1, \ldots, 16$.

In certain specific situations the bitrate available for the encoding of the secondary channel may become very low. For example, when the signal in the secondary channel is classified as INACTIVE the bitrate might be reduced to 1.65 kb/s. In that case, the IC encoder described in clause 5.2.3.5 of [3] is used but modified in such a way that the temporal contribution is skipped, the signal is converted to the frequency domain and only the first half of the spectral band energy is encoded. The procedure of splitting the signal into spectral bands and calculating per-band spectral energy is described in clause 5.2.3.5.7 of [3]. The remaining spectral band energy and the difference spectrum, calculated as described in clause 5.2.3.5.9 of [3], may not be encoded. When the encoding of the second half of the spectrum is not performed, noise-filling mechanism will be applied at the decoder side instead. Furthermore, in case the signal in the secondary channel is classified as UNVOICED and when only a minimal bitrate is available for the encoding of the secondary channel, a similar approach may be taken by coding only the spectral gains and not the spectral bins. It is noteworthy to say that the LP filter quantization in the UNVOICED coder type takes more bits when compared to the INACTIVE coder type.

5.3.2.3.8 Bit allocation

The bit allocation mechanism calculates the bit budget available for the encoding of the primary channel and the secondary channel. The bit allocation mechanism uses the following information and parameters to determine the optimal distribution of bits among the primary and the secondary channel:

- mixing ratio β calculated in equation (5.3-67)
- the LRTD sub-mode
- re-use of the LP filter coefficients from the primary channel described in clause 5.3.2.3.5
- OL pitch coherence as described in clause 5.3.2.3.6
- coder type of the primary and the secondary channel
- detected bandwidth of the primary and the secondary channels
- OL pitch coherence S_{pc}, calculated in equation (5.3-86)

First, the minimum bitrate of the secondary channel is set based on the following table:

Table 5.3-2: Minimum bitrate of the secondary channel

Available bitrate [kbps]	Coder type of the secondary channel		
	INACTIVE	UNVOICED	GENERIC
≤ 13.2	1.65	3.50	4.40
≤ 16.4	1.65	3.50	5.00
≤ 24.4	1.65	3.50	6.00
≤ 32	1.65	6.05	10.0
≤ 48	1.65	6.05	13.0

For the INACTIVE coder type the bitrate of the secondary channel is initially set to the minimum bitrate according to the table above, i.e. 1.65 kb/s. In case the LRTD sub-mode has been selected within the TD stereo coder the bitrate may be increased depending on the mixing ratio β.

The bitrate allocated to the secondary channel is set based on the following parameters:

- coder type of the secondary channel
- index corresponding to the mixing ratio β, calculated in equation (5.3-67)
- index corresponding to the instantaneous mixing ratio β_{inst}, calculated in eq. (5.3-68)

If the coder type is UNVOICED and the LP filter from the primary channel is not re-used in the encoding of the secondary channel, then the bitrate assigned to the secondary channel is increased by the number of bits required for the transmission of the quantized LP filter coefficients.

Else, in case the input format is different than ISM or OMASA and, at the same time, the secondary channel coder type is different than UNVOICED or the LRTD sub-mode has been selected in the TD stereo coder, or if at the same time the input format is OMASA and an ISM encoder is used and the coder type is GENERIC, then secondary channel is adapted in the following way depending on the usage of the LRTD sub-mode or not.

In case LRTD sub-mode has been selected in the TD stereo coder the minimum bitrate is modified to a value between 30% of the total available bitrate for INACTIVE coder type and 50% of the total available bitrate for GENERIC or UNVOICED coder type. Thus, the minimum bitrate can be as low as 4 kbps for the INACTIVE coder type and up to 16 kbps for the GENERIC or UNVOICED coder type. In the next step, supplementary bitrate is calculated based on the difference between the index corresponding to the instantaneous mixing ratio β_{inst} and the index located at the middle of the quantization scale. The supplementary bitrate is added to the bitrate allocated to the secondary channel so far. The bit allocation mechanism ensures that the channel with higher energy (primary or secondary) has the higher bitrate. This means that, sometimes, the secondary channel may have higher bitrate than the primary channel in the LRTD sub-mode. This is particularly important in situations where the TD stereo coder is prepared to switch the primary and the secondary channel, but it has not done so yet due to various reasons.

If LRTD sub-mode has not been selected in the TD stereo coder, then the first estimation of the bit budget distribution between the primary and the secondary channel is computed by subtracting from the total available bit rate, twice the minimum bit rate allocated to the secondary channel given by Table 5.3-2. Then, a supplementary bitrate is computed based on the difference between the index corresponding to the mixing ratio β and the index located at the middle of the quantization scale. The supplementary bitrate, which can be negative, and the bit budget available for distribution are added to the bitrate allocated to the secondary channel given by Table 5.3-2. Please, note that the bitrate of the secondary channel cannot be higher than 45% of the total available bitrate if LRTD sub-mode has not been selected in the TD stereo coder.

After bitrate allocation, a series of verification steps is performed depending on the bandwidth, the available bitrate and the selection of the LRTD sub-mode to ensure that the secondary channel has enough bits to encode all its parameters but never more than 18 kbps.

Another verification step is performed in case the coder type is INACTIVE or UNVOICED. First, the bitrate of the secondary channel is limited to 11 kbps. If the bitrate of the secondary channel is higher or equal to 9 kbps and the LP filter is not re-used from the primary channel or if the bitrate of the secondary channel bitrate is greater or equal to 8.2 kbps, then the excitation signal will be encoded for both the UNVOICED or INACTIVE coder type as described in clauses 5.2.3.3 and 5.2.3.5 of [3]. Otherwise, a more specific encoding strategy will be applied as explained in clause 5.3.3.3.4.5.

Finally, if the LP filter is not re-used and the bitrate of the secondary channel is lower than 7 kbps and the coder type is UNVOICED or if the LP filter is not re-used and the bitrate of the secondary channel minus the bitrate needed to encode the LP filter coefficients is lower than the minimum bit rate, as defined in Table 5.3-2, the bitrate of the secondary channel is increased to accommodate the LP filter coefficients.

The bitrate of the primary channel bitrate is initialized to the difference between the available bitrate and the bit rate allocated to the secondary channel. If the resulting bitrate is not sufficient to encode the primary channel with the selected coder type including the bandwidth extension, then it is necessary to reduce the bitrate allocated to the secondary channel. As an example, in case the encoder of the primary channel uses the internal sampling rate of 16 kHz, then the minimum bitrate of the primary channel is set to 14 kbps plus the bitrate needed to encode the BWE. In case the encoder of the primary channel uses the internal sampling rate of 12.8 kHz, then the minimum bitrate is set either to 7.2 kbps, in case of the TRANSITION coder type, or to 5.9 kbps for all other coder types plus the bitrate needed to encode the BWE.

In its last step, the TD stereo encoder performs a verification to ensure that the bitrate of the secondary channel is higher than the absolute minimum of 1.65 kbps for the INACTIVE coder type or 3.5 kbps for all other coder types.

5.3.2.3.9 Quantization of LP coefficients for the secondary channel

As described in clause 5.3.2.3.8, when the LP-filter coefficients for the secondary channel are not directly reusing the primary channel parameters, the quantization is based on the primary channel quantized LP coefficients. Therefore, as mentioned in clause 5.3.2.3.5, an additional bit is allocated for the encoding of supplementary information relating to

the secondary channel LP coefficients. The LP coefficients of a channel provide an estimation of the spectral envelope of the channel and for both channels they are used in the line spectrum frequencies (LSF) representation.

The values of the quantized LP coefficients of the secondary channel are estimated by using prediction based on the first channel quantized LP coefficients with a predictor matrix, A. The predictor matrix A is tri-diagonal matrix whose elements are predictor coefficients given by table 5.3-3. The predicted LP filter coefficients for the secondary channel are given by

$$LSF_{pred}^{[1]} = A\left(\begin{pmatrix} mod\_lsf_1^{[0]} \\ \dots \\ mod\_lsf_M^{[0]} \end{pmatrix} - \overline{LSF}_{in}\right) + \overline{LSF}_{out}$$

$$mod\_lsf_i^{[0]} = \gamma * lsf_i^{[0]} + (1-\gamma) * \overline{lsf}_i$$

where $mod\_lsf_i^{[0]}, i = 1{:}\, M$ is a modified version of the primary channel coefficients by bringing it closer to the average LSF vector, $\overline{LSF}_{in}$, $\overline{LSF}_{out}$,$\overline{LSF}$ are constant LSF average vectors whose values are given in table 5.3-4, γ is a constant dependent on the IVAS codec total bitrate as presented in the table 5.3-5. The upper index [0] and [1] indicate parameters corresponding to the primary and to the second channel respectively. Capital letter notation represents the vector and small letters notation represent the vector components.

The prediction error is then computed as the difference between the LP filter coefficients of the secondary channel and the predicted filter coefficients of the secondary channel $LSF_{pred}^{[1]}$.

Table 5.3-3: Tri-diagonal prediction matrix

First lower diagonal	Main diagonal	First upper diagonal
0.7040	0.1203	
-0.1119	0.7340	0.1803
0.0253	0.6702	0.1901
-0.1847	0.7892	0.3010
-0.0418	0.8716	0.1837
0.0033	0.6915	0.2394
0.0213	0.6728	0.2441
0.0705	0.7549	0.1983
0.0752	0.7152	0.2173
0.0886	0.6163	0.3067
0.0217	0.8121	0.2021
0.1584	0.7311	0.1746
0.1204	0.7296	0.1978
0.1231	0.7502	0.1234
0.1709	0.6372	0.1060
	0.1193	0.6574

Table 5.3-4: LSF average vectors

$\overline{lsf}_{in}$	$\overline{lsf}_{out}$	$\overline{lsf}$
288.540	286.414	391.31345
535.469	522.366	608.50453
899.738	887.297	968.00585
1342.313	1347.961	1354.23965
1730.736	1725.604	1709.71084
2107.070	2102.356	2080.49872
2491.455	2511.703	2450.64009
2859.828	2853.093	2796.96588
3239.279	3211.319	3196.19608
3625.673	3612.072	3554.17092
3992.540	3970.889	3915.02370
4356.748	4327.774	4283.81121
4752.356	4732.423	4707.59835
5153.685	5154.984	5109.79026
5567.107	5572.849	5526.44936
5972.623	5964.332	5903.42625

Table 5.3-5: γ value for calculation of mod_lsf

Available bitrate [kbps]	γ
≤ 13.2	0.87
≤ 16.4	0.94
≤ 24.4	0.91
≤ 32	0.92
>32	0.91

The prediction error is quantized using the lattice quantizer structures as described in clause 5.2.2.1.4 of [3] using the available number of bits. The weighted Euclidean quantization distortion of the reconstructed LSF values is calculated. Simultaneously, a second estimate based on an AR predictor similar to the AR prediction mode from clause 5.2.2.1.4 of [3], with respect to the previous frame LSF values of the secondary channel is calculated. The AR prediction error is quantized and the weighted Euclidean quantization distortion of the corresponding reconstructed LSF values is calculated. The prediction based on the primary channel coefficients is considered to be as a safety net predictor because it is relying on data of the current frame. The selection between the prediction based on the current frame primary channel LP coefficients and the prediction based on the previous frame secondary channel coefficients is made based on the calculated quantization distortions and using the additional selection criteria between safety net and predictive modes as in clause 5.2.2.1.4.1 of [3].

5.3.2.4 DFT-based stereo

5.3.2.4.1 General

The DFT-based stereo is a joint Mid/Side (M/S) stereo coding exploiting some spatial cues, where the Mid-channel is coded by a primary mono core-coder as described in clause 5.2.2. The Side-channel is predicted parametrically and optionally the residual of its estimation coded by a secondary core-coder as described below.

The main encoding processing is depicted in Figure 5.3-24. The DFT stereo operates mainly in the frequency domain after a STFT analysis of the time-aligned or partly time-aligned stereo channels and producing for the Mid signal a downmix signal which is conveyed back to the time domain after STFT syntheses to the different core-coding schemes, that means either to ACELP and the associated TD-BWE, or to one of the two MDCT-based core-coders, namely TCX associated to IGF and HQ-Core. The Side-signal is usually solely modelled by the stereo parameters, but can be partly and discretely transmitted at higher bitrates, 32 kbps and above, through its prediction residual signal coding of the first 1 kHz. The residual coding of the Side signal operates in a separated MDCT where a scalar quantization is followed by an entropy coding.

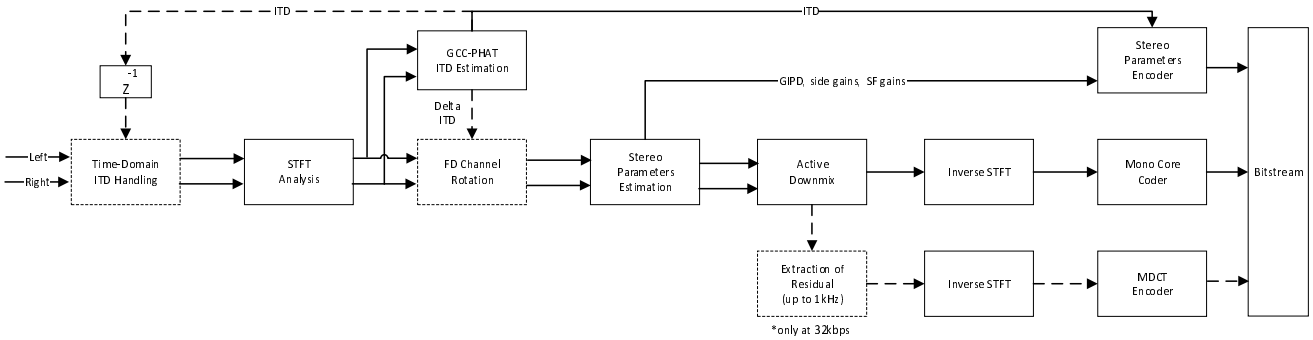


Figure 5.3-24: block diagram of DFT stereo encoder

The DFT stereo encoding scheme consists of the following steps: a time-domain Inter-channel Time Difference (ITD) – compensation, a STFT analysis, a GCC-PHAT ITD estimation, a stereo parameter estimation, and active downmixing, a stereo parameters encoder, an optional residual coding, and inverse STFTs providing the downmix channel at different sampling rates and in different frequency bands. The different steps are detailed in the following clauses.

5.3.2.4.2 Time Domain ITD compensation

5.3.2.4.2.1 General

The time alignment of the two channels is achieved by compensating the ITD by time shifting one of the two channels at the input sampling rate and in time domain. The lagging channel is advanced, and for not engendering extra delay, the samples beyond the available samples are extrapolated.

Moreover, since the ITD is estimated in the frequency domain after the time-domain ITD compensation occurs, the ITD estimated at the previous frame is employed. The delta between the ITD used in TD and the current estimated ITD will be used in the frequency-domain ITD compensation.

5.3.2.4.2.2 Lagging channel extrapolation

For the extrapolation of the lagging channel the assumption is made that the lagging channel can be locally described as

$$l(t) = g \cdot L(t - ITD) + n(t) \quad (5.3\text{-}90)$$

where: $l(t)$ denotes the lagging channel;

$L(t)$ the leading channel;

ITD the ITD value computed in the previous frame,

$n(t)$.the residual signal;

and g a scaling gain.

The extrapolation itself is done as a mix of LPC-based extension of the lagging channel and a prediction based on the leading channel. The residual $n(t)$ is thereby extrapolated using a properly scaled version of the LPC extrapolation.

The scaling gain g is calculated as

$$g = \frac{\langle L(\cdot - ITD), l \rangle}{\langle L(\cdot - ITD), L(\cdot - ITD) \rangle} = \frac{\sum_t L(t-ITD)\, l(t)}{\sum_t L(t-ITD)^2}, \quad (5.3\text{-}91)$$

where $t = 0, \ldots, frame + dft\_ovl - ITD$ with $frame$ the frame size in samples and dft_ovl the DFT overlap in samples.

Furthermore, an estimate of the noise-to-signal ratio NSR is calculated as

$$NSR = 1 - \frac{\langle L(\cdot - ITD), l \rangle^2}{\langle l, l \rangle \langle L(\cdot - ITD), L(\cdot - ITD) \rangle} = \frac{\sum_t L(t-ITD) l(t)^2}{\sum_t l(t)^2 \sum_t L(t-ITD)^2}, \quad (5.3\text{-}92)$$

where $t = 0, \dots, frame + dft\_ovl - ITD$ with $frame$ the frame size in samples and dft_ovl the DFT overlap in samples.

The computed values for g and NSR are then bounded as

$$g \leftarrow \min(\max(g, -1), 1.5) \tag{5.3-93}$$

and

$$NSR \leftarrow \min(\max(NSR, 0), 1). \tag{5.3-94}$$

From NSR a gain g_{lpc} for the LPC extrapolation is derived:

$$g_{lpc} = \sqrt{NSR} \tag{5.3-95}$$

Subsequently, linear prediction coefficients are derived via autocorrelation and Levinson-Durban algorithm. This is done in the same manner and with the same number of coefficients (LPC order 10) as for the TBE LP analysis which is described in clause 5.2.6.1.2 of [3]. However, a rectangular window of 320 samples is used instead of the TBE LP analysis window.

The residual signal $n(t)$ is obtained by filtering the last 320 samples of the lagging channel through an FIR filter with the computed LPCs. The residual is then extended via

$$n(t) = n(t - ITD) \;\; for\; t = 320, \dots, 320 + ITD - 1 \tag{5.3-96}$$

Now, an extended lagging channel $\widehat{l(t)}$ is obtained by applying the inverse LPC filter to the extended residual signal. Combining this with the scaling gain g_{lpc} and the leading channel prediction results in the final extrapolated signal

$$\widehat{l(t)} \leftarrow g_{lpc}\widehat{l(t)} + gL(t - ITD) \tag{5.3-97}$$

for the missing ITD samples of the lagging channel.

Finally, a linear cross-fade between the original lagging channel and the extrapolated version is performed for the last 2 ms of the original signal to create a smooth transition to the newly extrapolated samples and avoid discontinuities.

5.3.2.4.3 STFT analysis

Most of stereo analysis and processing is performed in frequency domain after a Short-term Fourier Transform (STFT) analysis on the stereo channels of the input audio signal at the input sampling-rate. An analysis window is applied for allowing a good auditory scene analysis and stereo processing in the frequency domain. The windowed STFT (Short Time Fourier Transform) analysis formula can be expressed as:

$$S(c, m, \mathrm{k}) = \sum_{n=0}^{\mathrm{N}-1} s[c, n + m.M - zp] w_{ana}[n] e^{-j\frac{2\pi \mathrm{k} n}{N}}, \tag{5.3-98}$$

where $s[c, n]$ is the time-domain input signal for the channel index *c*, $w_{ana}[n]$ is the analysis window function, m is the index of the *M* stride size (stride parameter), zp is an offset corresponds to the zero padding of the analysis windows, k is the frequency index, *N* the DFT size corresponding to a total duration of 40 ms at the signal sampling rate, and *j* is the imaginary unit.

For analysis the audio signal is divided into overlapping windows, wherein the stride size *M* corresponds to the frame duration of core-coder, i.e. 20 ms and the overlapping region to the encoder lookahead of the core-coders, i.e. 8.75 ms. For getting even finer frequency resolution and to counterbalance the circular shift when time aligning the channels in frequency domain, zero padding of 5.625 ms is added at both sides to the analysis windows for a total window length of 40 ms as depicted in Figure 5.3-25.

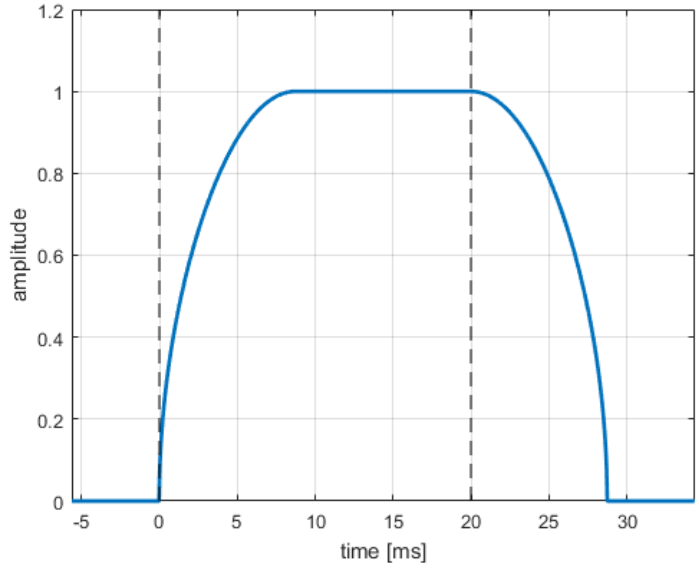


Figure 5.3-25: STFT analysis window with a stride of 20 ms and an overlapping of 8.75 ms

The overlapping window part is 8.75 ms and corresponds to the lookahead needed for the pre-processing and LP analysis used by the core-code. It also corresponds to overlapping region of the MDCT used in TCX and HQ-Core. The analysis window is defined as:

$$w_{ana}[n] = \begin{cases} \sqrt{sin(\frac{\pi.(n-zp+0.5)}{2.L})} \; for \; zp \le n < zp + L \\ 1, for \; zp \; + \; L \le n < M \\ \sqrt{\sin\left(\frac{\pi.(n-zp+L-M+0.5)}{2.L}\right)} for \; M \le n < M + L \end{cases} \quad (5.3\text{-}99)$$

and zero otherwise.

5.3.2.4.4 GCC-PHAT ITD estimation

5.3.2.4.4.1 Description of GCC-PHAT implementation

The inter-channel time difference (ITD) is computed by estimating the Time Delay of Arrival (TDOA) using an adapted version Generalized Cross Correlation with Phase Transform (GCC-PHAT), where the inter-channel cross-correlation spectrum, the normalization as well as the maximum finding are adapted to the characteristic of the input signal. ITD estimation can summarized by the following equation:

$$ITD_m = argmax'(IDFT(\frac{S_{01,LP}(m,k)}{|S_{01,LP}(m,k)|^{\alpha}})) \quad (5.3\text{-}100)$$

where $S_{01,LP}(m,k)$ is the smoothed version of the cross-correlation spectrum over time for the frame index *m*, derived from the cross-correlation spectrum computed in frequency domain, where α is a normalization factor depending on the noisiness characteristic of the signal, and $argmax'$ a peak picking algorithm, more evolved than a simple argmax operation. The spectrum of cross-correlation function is first derived between the left and right channels as:

$$S_{01}(m,k) = S_0(m,k)S_1(m,k)^* \quad (5.3\text{-}101)$$

where $S_0(m,k)$ and $S_1(m,k)$ are the frequency spectra of the left and right input channels respectively for the frame index *i*, derived from the STFT analysis described above, where *k* is the DFT frequency index, and where * is the complex conjugate operator.

The cross-correlation spectrum is then smoothed depending on the Spectral Flatness Measurement (SFM), a spectral characteristic of the input signal. The SFM is defined as the ratio between the geometric mean over the arithmetic mean of the power spectrum. The SFM is bounded between 0 and 1. In case of noise-like signals, the SFM will tend to be high (i.e. around 1) and the smoothing weak. On the other hand, in case of tone-like signal, SFM will tend to be low and the smoothing will become stronger. SFM is computed on the same DFT spectra as used for the DFT stereo processing

and for the spectrum computation of the cross-correlation, and this for both the left and right channels. The SFM is given by:

$$SFM\_L = \frac{\exp\left(\frac{1}{N_{FFT}}\sum_{k=0}^{N-1} \log|S_0(m,k)|\right)}{\frac{1}{N_{FFT}}\sum_{k=0}^{N-1} |S_0(m,k)|} \quad (5.3\text{-}102)$$

where N is the number of DFT bins till a maximum of 8 kHz of audio bandwidth. The SFM for the right channel SFM_R is computed the same way- The final *SFM* value used for the cross-correlation spectrum smoothing is the maximum of the two SFM measures. The smoothed cross-correlation spectrum over time is then obtained with the following filtering:

$$S_{01,LP}(m,k) = \mathrm{SFM} \cdot S_{01}(m,k) + (1 - SFM) \cdot S_{01,LP}(m-1,k) \quad (5.3\text{-}103)$$

The smoothed cross-correlation is then normalized by its amplitude or one of its norm before being transformed back to time domain. The normalization corresponds to the Phase –transform of the cross-correlation, and is known to show better performance than the normal cross-correlation in low noise and relatively high reverberation environments. However, to make it even more robust the norm of the numerator is adapted depending of the nature of the signal, leading to the normalized cross-correlation function $r_{01}(m,l)$ defined in time as:

$$r_{01}(m,l) = DFT^{-1}\left(\frac{S_{01,LP}(m,k)}{\left|S_{01,LP}(m,k)\right|^{\alpha}}\right) \quad (5.3\text{-}104)$$

where $\alpha = 1$ for clean speech and $\alpha = 0.8$ when noisy speech is detected in the pre-processing stage as described in 5.2.2.2.

The so-obtained time domain function is first filtered for achieving a more robust peak peaking. The index corresponding to the maximum amplitude corresponds to an estimate of the time difference between the Left and Right Channel (ITD). If the amplitude of the maximum is lower than a given threshold, which is determined as described in clause 5.3.2.4.4.3, then the estimated of ITD is not considered as reliable and depending on the logic described in clause 5.3.2.4.4.4 it may be set to zero.

5.3.2.4.4.2 DTX hangover adaptation

To handle the transitions from active segments to inactive segments, an additional filtered cross-spectrum $S_{01,LPCNG}(m,k)$ is maintained. It is evaluated during the hangover period in the start of an inactive segment.

$$S_{01,LPCNG}(m,k) = \beta_{CNG} S_{01}(m,k) + (1 - \beta_{CNG}) S_{01,LPCNG}(m-1,k) \quad (5.3\text{-}105)$$

where β_{CNG} is an adaptive filter coefficient. If we are in the first frame of a VAD hangover period, the filtered cross-spectrum memory is conditionally reset to zero if there are more than 20 preceding frames of active coding, $S_{01,LPCNG}(m-1,k) := 0$, where the filtered cross-spectrum from the previous frame also represents the filter memory, meaning that the filtered cross-spectrum for this frame is defined according to

$$S_{01,LPCNG}(m,k) = \beta_{CNG} S_{01}(m,k). \quad (5.3\text{-}106)$$

During DTX hangover and in CNG frames a faster adaptation of $S_{01,LPCNG}(m,k)$ is achieved by increasing the low-pass filter coefficient β_{CNG} according to

$$\beta_{CNG} = \max\left(sfm, \min\left(0.8, \frac{8}{N_{exp}+N_{upd}}\right)\right) \quad (5.3\text{-}107)$$

where N_{exp} is the number of expected updates to $S_{01,LPCNG}(m,k)$ during the hangover period before it is used and N_{upd} is the actual number of updates since the first hangover frame where the filtered cross-spectrum memory was reset to zero. In the first hangover frame, N_{exp} is calculated as

$$N_{exp} = 1 + \min\left(N_{rem\_dtx\_ho0}, N_{rem\_dtx\_ho1}\right) \quad (5.3\text{-}108)$$

where $N_{rem\_dtx\_ho0}, N_{rem\_dtx\_ho1}$ are the remaining hangover frames as indicated by the Front VAD as described in 5.3.2.2.1.1. For the remaining hangover frames, N_{exp} is updated according to

$$N_{exp} = \begin{cases} N_{exp}^{[-1]} + 1 + \min\left(N_{rem\_dtx\_ho0}, N_{rem\_dtx\_ho1}\right), N_{upd} \geq N_{exp} \\ N_{exp}^{[-1]}, \quad otherwise \end{cases} \quad (5.3\text{-}109)$$

In the first CNG frame after the hangover period, the low-pass filtered CNG cross-spectrum is copied to the low-pass filtered cross-spectrum

$$S_{01,LP}(m,k) = S_{01,LPCNG}(m,k) \quad (5.3\text{-}110)$$

and the GCC-PHAT algorithm is evaluated as in 5.3.2.4.4.1. When the CNG period has started, the purpose of $S_{01,LPCNG}(m,k)$ is shifted from fast ITD adaptation to instead be used for coherence estimation, as described in 5.3.5.1.5. During the following CNG frames, a slow adaptation of $S_{01,LPCNG}(m,k)$ is maintained by setting $\beta_{CNG} = 1/32$.

5.3.2.4.4.3 Peak picking algorithm for ITD estimation

For picking the maximum peak, the threshold is defined by first calculating the mean of a rough computation of the envelope of the magnitude of the cross-correlation function $r_{01}(m,l)$ defined by Equation (5.3-104) within the [-ITD_MAX, ITD_MAX] region as shown in Figure 5.3-26. ITD_MAX is set to 160 samples at a 32 kHz sampling rate. The average is then weighted accordingly depending on the SNR estimation.

The SNR estimation is based on the mean energy ratio mE_r as defined in :

$$snr = 20\, log\, m\, E_r \quad (5.3\text{-}111)$$

and where mE_r is calculated by the average of $E_r\,(m,k)$ over the N stereo bands, where E_r is computed by the expression:

$$E_r(\mathrm{m,k}) = \frac{\left(S_{00,LP}(m,k)+S_{11,LP}(m,k)\right)+\sqrt{\left(S_{00,LP}(m,k)-S_{11,LP}(m,k)\right)^2+4\left|S_{01,LP}(m,k)^2\right|}}{\left(S_{00,LP}(m,k)+S_{11,LP}(m,k)\right)-\sqrt{\left(S_{00,LP}(m,k)-S_{11,LP}(m,k)\right)^2+4\left|S_{01,LP}(m,k)^2\right|}} \quad (5.3\text{-}112)$$

where: $S_{01,LP}(m,k)$ is defined in clause 5.3.2.4.4.1, $S_{00,LP}(m,k)$ and $S_{11,LP}(m,k)$ is the autocorrelation of the left and right channel respectively defined by $S_{00,LP}(m,k) = S_0(m,k)S_0(m,k)^*$ and $S_{11,LP}(m,k) = S_1(m,k)S_1(m,k)^*$ and smoothed over time as defined in Equation (5.3-103).

The step-by-step description of the algorithm for determining the threshold is described below:

- The cross-correlation function $r_{01}(m,l)$, is rearranged from negative to positive time lags as shown in Figure 5.3-26 of an example frame.
- The cross-correlation function is divided in three main areas: the area of interest namely [-ITD_MAX, ITD_MAX] and the area outside the ITD_MAX bounds, namely time lags smaller than –ITD_MAX (max_low) and higher than ITD_MAX (max_high). The maximum peaks of the “out of bound” areas are detected and saved to be compared to the maximum peak detected in the area of interest.
- In order to determine whether a valid ITD is present, the sub-vector area [-ITD_MAX, ITD_MAX] of the cross-correlation function is considered. The sub-vector is divided into N sub-blocks as shown in Figure 5.3-26.

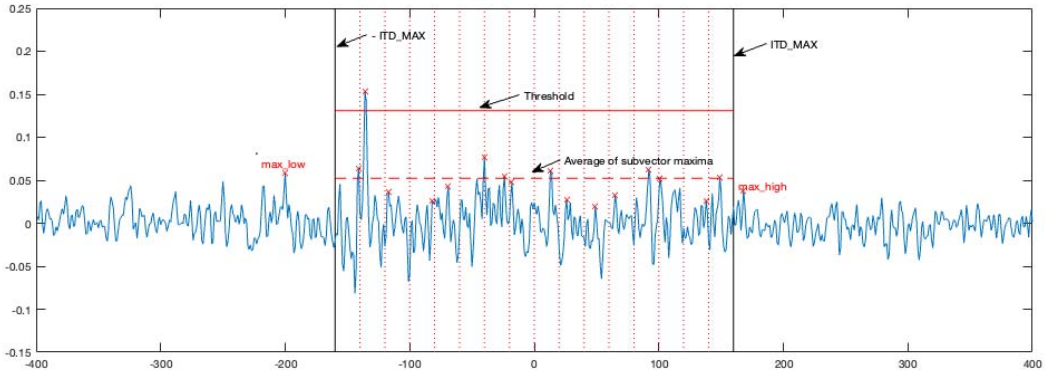


Figure 5.3-26: Peak picking from sub-blocks division

- For each sub-block the maximum peak magnitude r_{sub} and the equivalent time lag position l_{sub} is found and saved.
- The maximum of the local maxima r_{max} is determined and will be compared to the threshold to determine the existence of a valid ITD value.
- The maximum value r_{max} is compared to max_low and max_high. If r_{max} is lower than either of the two than no itd handling is signaled and no time alignment is performed. Because of the ITD handling limit of the system, the magnitudes of the out of bound peaks do not need to be evaluated.

The mean of the magnitudes of the peaks is calculated:

$$r_{mean}(m) = \sum_{0}^{N} r_{sub}/N$$

If the estimated SNR from Equation (5.3-111) is higher than 25 dB than the cross-correlation vector is low-pass filtered prior to applying the threshold calculation. Otherwise, the threshold $thres$ is then computed by weighting $peak_{mean}$ with an SNR depended weighting factor a_w as described in the pseudocode:

```
if ( snr <= 20.f && snr > 15.f )
{
    wfac = snr * 0.1f + 0.5f;
}
else
{
    wfac = 2.5f;
}
```

For very noisy conditions of low SNR and when $peak_{mean}$ is very close to the maximum peak, there is fine tuning of the weighting factor to accommodate instabilities as defined in the pseudocode below:

```
thres_diff = wfac * avg_max - *max_max;

if ( vad && thres_diff > 0.f && ( ( thres_diff < 0.05f && ( snr <= 15 && snr > 7.f ) ) || (
thres_diff < 0.01f && ( snr > 15.f && snr < 30.f ) ) ) )
{
     wfac = 2.0f;
}

if ( flag_noisy_speech_snr == 1 )
{
     if ( vad == 0 )
     {
         wfac = 2.5f;
      }
       else if ( detected_itd_flag == 0 && *max_max > 1.5f * avg_max && *prev_max > 1.5f *
*prev_avg_max && abs( *index - *prev_index ) <= 2 )
        {
            wfac = 1.5f;
         }
         else
         {
            wfac = 2.0f;
         }
}
```

The threshold $r_{thr}(m)$ is then calculated by applying the weighting factor on the mean peak, hence $r_{thr}(m) = wfac \cdot r_{mean}$. Finally, $if\ r_{max}(m) > r_{thr}(m)$ the equivalent time lag l is returned as the estimated ITD, elsewise no itd handling is signalled (ITD=0).

5.3.2.4.4.4 ITD stabilization hangover

The ITD candidate $ITD_{cand}(m)$ of the current frame m is identified as the lag l which yields the maximum absolute value $r_{max}(m)$ of $r_{01}(m, l)$, corresponding to the inter-channel correlation.

$$\begin{cases} ITD_{cand}(m) = \underset{l}{\operatorname{argmax}} |\, r_{01}(m,l)| \\ r_{max}(m) = \left| r_{01}\left(ITD_{cand}(m)\right)\right| \end{cases} \quad (5.3\text{-}113)$$

However, background noises and reverberation may introduce noise in the cross-correlation, making the identification of a maximum lag more difficult. For this reason, $r_{max}(m)$ is compared to an adaptive threshold $r_{thr}(m)$. In case $r_{max}(m) > r_{thr}(m)$ the candidate ITD is selected as the valid ITD of the current frame, $ITD(m) = ITD_{cand}(m)$. In case $r_{max}(m)$ is hovering near the threshold $r_{thr}(m)$ for several consecutive frames m, the stability of the ITD may suffer. To stabilize the ITD the following measures are taken. $r_{max}(m)$ is low-pass filtered to form a stability estimate of $ITD_{cand}(m)$.

$$r_{stab}(m) = \begin{cases} \alpha_r r_{max}(m) + (1-\alpha_r) r_{stab}(m-1), & r_{max}(m) > r_{thr}(m) \\ 0, & r_{max}(m) \le r_{thr}(m) \end{cases} \quad (5.3\text{-}114)$$

where the low-pass filter coefficient α_r is set using a fast-attach-slow-decay strategy according to

$$\alpha_r = \begin{cases} 0.9, & r_{max}(m) > r_{stab}(m-1) \\ 0.1, & r_{max}(m) \le r_{stab}(m-1) \end{cases} \quad (5.3\text{-}115)$$

An ITD counter $ITD_{cnt}(m)$ is maintained to keep track of consecutive valid ITDs.

$$ITD_{cnt}(m) = \begin{cases} \min(ITD_{cnt,max}, ITD_{cnt}(m-1)+1), & r_{max}(m) > r_{thr}(m) \\ 0, & r_{max}(m) \le r_{thr}(m) \end{cases} \quad (5.3\text{-}116)$$

where the maximum value of the ITD counter is $ITD_{cnt,max} = 2$. If an ITD is not found to be valid, i.e. $r_{max}(m) \le r_{thr}(m)$, and a sufficient number of valid ITD estimates are found in the preceding frames, meaning the maximum value of the ITD counter was reached in the preceding frame $ITD_{cnt}(m-1) = ITD_{cnt,max}$, an ITD hangover counter is calculated according to

$$ITD_{HO}(m) = \max\left(0, \min\left(6, 11 + r_{stab}(m-1)\tfrac{50}{3}\right)\right) \quad (5.3\text{-}117)$$

If a valid ITD is not found $r_{max}(m) \le r_{thr}(m)$ and the hangover count exceeds zero $ITD_{HO}(m) > 0$, we are in the ITD hangover time and the previous ITD is selected for the current frame, i.e. $ITD(m) = ITD(m-1)$. If the hangover count is at zero $ITD_{HO}(m) = 0$, the ITD of the current frame is zeroed $ITD(m) = 0$.

Hereinafter, or to simplify notation, the stride index m is replaced by the frame index i, the stride size being the same as the frame size, and the spectrum $S(c,m,k)$ is replaced by $L_i(k)$ for c=0, for the left channel, and $R_i(k)$ for c=1, the right channel.

5.3.2.4.4.5 Refined ITD control mechanism

The ITD is critical to keep the stereo image stable. Sometimes ITD values calculated by the frequency domain correlation are not continuous. A difference parameter is used to represent a difference between the ITD of the current frame $ITD(m)$ and the ITD of the previous frame $ITD(m-1)$. ITD is determined based on the difference parameter and a characteristic parameter of the current frame. The characteristic parameter is calculated as follows.

Divide a low frequency part of the left-channel frequency-domain signal of the current frame into M sub-bands, where each sub-band includes N frequency domain amplitude values. Calculate a correlation parameter of the current frame and a previous frame according to

$$cor(i) = \frac{\sum_{j=0}^{N-1} |L(i*N+j)| \cdot \left|L^{(-1)}(i*N+j)\right|}{\sqrt{\sum_{j=0}^{N-1} |L(i*N+j)| \cdot |L(i*N+j)| \cdot \sum_{j=0}^{N-1} |L^{(-1)}(i*N+j)| \cdot |L^{(-1)}(i*N+j)|}}$$

i = 0,1, … , M − 1, where $|L(i*N+j)|$ represents a j^{th} frequency domain amplitude value of an i^{th} sub-band in the low frequency part of the left-channel frequency-domain signal of the current frame, $\left|L^{(-1)}(i*N+j)\right|$ represents a j^{th} frequency domain amplitude value of an i^{th} sub-band in a low frequency part of a left-channel frequency-domain signal of the previous frame, and $cor(i)$ represents a normalized cross-correlation value corresponding to an i^{th} sub-band in the M sub-bands.

Calculate a peak-to-average ratio of each sub-band of the current frame $par\_L(i)$.

If the ITD value of the current frame and an ITD value of the previous frame meet at least of one of the preset conditions, determine whether to reuse the ITD value of the previous frame for the current frame. The preset conditions set to

- condition_1: the absolute ITD value of the previous frame is greater than the absolute ITD value of the current frame,

condition_1 = $|ITD(m-1)| > 0.2 * |ITD(m)|$

- condition_2: the average value of the normalized cross-correlation values of the sub-bands is greater than 0.85,

condition_2 = avrg($cor(i)$) > 0.85

- condition_3: the average value of the normalized cross-correlation values of the sub-bands is greater than 0.7, and a normalized cross-correlation value of a sub-band is greater than 0.9,

condition_3 = avrg($cor(i)$) > 0.7 and ($cor(0) > 0.9$ or $cor(1) > 0.9$ or $cor(2) > 0.9$)

- condition_4: the average value of the peak-to-average ratios of the sub-bands is greater than 0.6,

condition_4 = avrg($par\_L(i)$) > 0.6

- condition_5: the ITD value of the previous frame is not equal to 0,

condition_5 = $ITD(m-1) \neq 0$

- condition_6_a: a product of the ITD value of the previous frame and the ITD value of the current frame is negative,

condition_6_a = itd * prev_itd < 0

- condition_6_b: a product of the ITD value of the previous frame and the ITD value of the current frame is 0,

condition_6_b = itd * prev_itd = 0

- condition_6_c: an absolute value of a difference between the ITD value of the previous frame and the ITD value of the current frame is greater than half of a target value, where the target value is an ITD value whose absolute value is larger in the ITD value of the previous frame and the ITD value of the current frame,

$$\text{condition\_6\_c} = \begin{cases} |itd|, |\text{itd} - \text{prev\_itd}| > 0.5 * (|\text{itd}| > |\text{prev\_itd}|) \\ |prev_{itd}|, |\text{itd} - \text{prev}_{\text{itd}}| \leq 0.5 * (|\text{itd}| > |\text{prev\_itd}|) \end{cases}$$

The ITD fine control result itd_fine sets to

$$itd\_fine = \begin{cases} \text{prev}_{\text{itd}}, \text{condition}_{\text{total}} \text{ is true}) \\ itd, \text{others} \end{cases}$$

where

$\text{condition}_{\text{total}}$ =
(condition_1234 and ((condition_5 and condition_6_b) or condition_6_c)) or (condition_1234 and condition_6_a)
, condition_1234 = condition_1 and (condition_2 or condition_3 or condition_4)

when the signal-to-noise ratio meets the signal-to-noise ratio condition, stopping reusing the ITD value of the previous frame as the ITD value of the current frame. The signal-to-noise ratio condition is set as the SNR value is less than 0.006 or the SNR value is greater than 2 000 000.

In addition, the ITD of the current frame could reuse the ITD of the previous frame ITD(m-1) based on the hangover counter. The hangover counter is the quantity of target frames that are allowed to appear consecutively. A characteristic information which is described by the signal-to-noise ratio and the peak feature of cross correlation coefficients of the stereo signal is used to control the hangover counter. When the signal-to-noise ratio and the peak feature of the cross-correlation coefficients meet at least one of preset conditions, the hangover counter will be reduced by adjusting at least one of a target frame count and a threshold of the target frame count.

5.3.2.4.5 FD circular time-shift

The ITD compensation is achieved in both time and frequency domain. The application of time shift in FD is limited since it can only be achieved only up to a certain maximal lag without introducing strong window mismatch between channels. Nonetheless, the zero padding of the analysis window prevents any severe artefacts which could come from the circular shifting, since the maximum ITD of 5 ms is less than the both-sided zero-padding guards of 5.625 ms. Therefore, the channel time alignment is split in a way the Frequency Domain part is lower possible and correspond to a delta ITD. The Delta ITD is the difference between the ITD estimated at the previous frame and applied in TD at the present frame, and the estimated ITD at the current frame. The channels are time shifted in a way that the lagging channel is advanced in time, i.e. if the ITD is negative, the lagging left channel is advanced.

$$\begin{cases} L(k) = L(k)e^{-j2\pi k\frac{\Delta ITD}{NFFT}} if\ ITD < 0 \\ R(k) = R(k)e^{+j2\pi k\frac{\Delta ITD}{NFFT}} if\ ITD > 0 \end{cases} \quad (5.3\text{-}118)$$

To prevent that an eventual too strong window mismatch between the channels occur after a circular time shifting in FD, the attenuation factor is computed if the ΔITD is above 2.5 ms. This attenuation factor will be served to attenuate the residual signal if coded, since less relevant in case of strong window mismatch.

5.3.2.4.6 Stereo parameters estimation

Encoding of stereo parameters and computation of the downmix signal is done in frequency domain on the time-frequency vectors L_i and R_i of the left and right channel. The DFT bins are then grouped into subbands $(L_t, k)_k \in I_b$ resp. $(R_t, k)_k \in I_b$, where I_b denotes the set of subbands indices.

Different subband partitioning schemes are used, depending on the bitrate, but also on the processing (see Table 5.3-6). They roughly follow a scaled version of the equivalent rectangular bandwidth (ERB). For stereo parameter estimation, the subband partitioning B is used, which is dependent on the bitrate. For the downmix parameter estimation, the partitioning B2 is used, which is a superset of B with finer frequency resolution, further decomposing the subband partitioning B. Finally, the downmix equalization is performed on the highest subband resolution, which combines 2 consecutive frequency bins within a subband partitioning band B. If the number of frequency bins within a subband B is not a multiple of 2, then the last subband b3 consists of 3 frequency bins and is called triple.

Table 5.3-6: Subband portioning for stereo and downmix parameters.

Partitioning name	Used for	Subband upper limits in STFT frequency bin index
ERB 8	≤16.4 kbps, stereo parameters (partitioning B)	1, 5, 18, 41, 84, 214, 470, 601
ERB 4	Downmix parameters (partitioning B2) and stereo parameters (partitioning B) for >16.4 kbps	1, 3, 5, 10, 18, 26, 41, 56, 84, 132, 214, 342, 470, 601

5.3.2.4.7 IPD calculation, stabilization and encoding scheme

5.3.2.4.7.1 IPD calculation and stabilization

For the downmix, a single global inter-channel-phase-difference (IPD) *gIPD* is calculated over the first 8 subbands of the ERB 4 partitioning (up to DFT bin 84) as

$$gIPD = \arg(\sum_{k \in l_{1,..,8}} L_{t,k}\, R^*_{t,k}) \quad (5.3\text{-}119)$$

where $R^*_{t,k}$ denotes the complex conjugate of $R_{t,k}$.

To provide a more stable *gIPD* estimate, a stability mechanism is employed which is described in detail below and shown more comprehensively as a flow diagram in Figure 5.3-27.

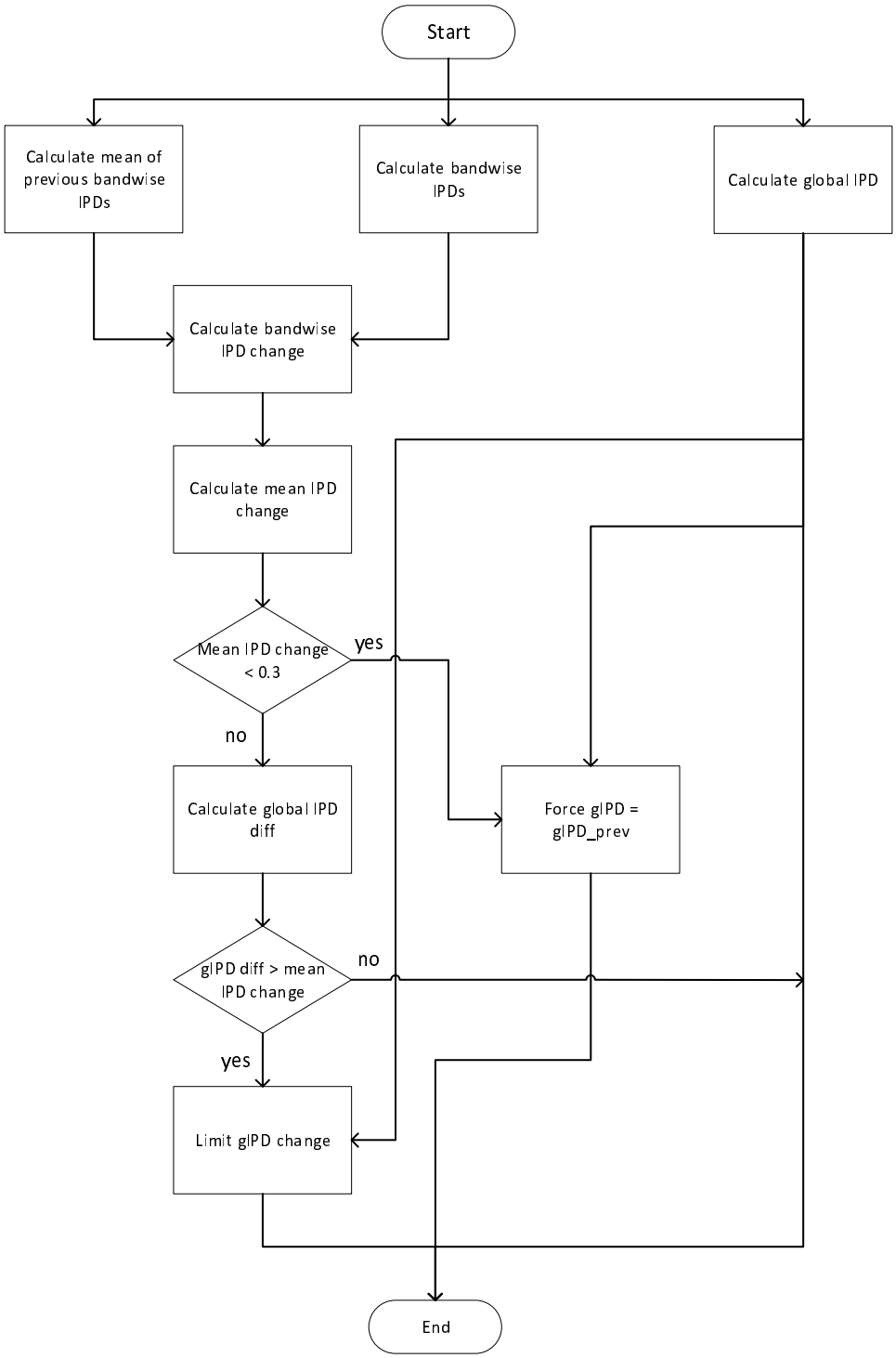


Figure 5.3-27: Flow diagram of global IPD stabilization

The stabilization first requires the calculation of additional bandwise phase differences $IPD_{t,b}$ as

$$IPD_{t,b} = \arg\left(\sum_{k \in l_b} L_{t,k}\, R^*_{t,k}\right) \quad (5.3\text{-}120)$$

for each of the 8 subbands over which the global IPD is calculated. Note that also for bitrates ≤ 16.4 kbps the ERB 4 bands are used for IPD calculation.

Additionally, in each subband bandwise mean IPDs over the 5 previous frames are calculated. Since distances between phases are ambiguous (2 possible directions on a circle) a meaningful bandwise mean IPD cannot always be calculated by standard averaging (only if all phases are within the same semi-circle). Instead, the bandwise mean IPD of a subband, denoted as $IPD_{mean,b}$, may be initialized with 0 and then updated iteratively with

$$IPD_{mean,b} = \frac{i}{i+1} IPD_{mean,b} + \frac{1}{i+1} IPD_{prev,b}[i] \quad (5.3\text{-}121)$$

where $i = 0, \ldots, 4$ is the index over the previous IPD values of the band. After each iteration, the distance of the current result to the next value in the IPD_{b_prev} buffer is calculated:

$$IPD_{diff} = \left| IPD_{mean,b} - IPD_{prev,b}[i+1] \right| \quad (5.3\text{-}122)$$

If IPD_{diff} is greater than π, i.e. more than a half-circle rotation in the given direction, $IPD_{mean,b}$ needs to be temporarily shifted outside of the $[-\pi, \pi]$ range by adding or subtracting 2π depending on which side of the circle it lies on:

$$IPD_{mean,b} = IPD_{mean,b} + 2\pi, \ if\ IPD_{mean,b} < 0 \quad (5.3\text{-}123)$$

or

$$IPD_{mean,b} = IPD_{mean,b} - 2\pi, \ if\ IPD_{mean,b} > 0 \quad (5.3\text{-}124)$$

Then the mean will be updated using this shifted version which now has a distance of less than π to the next value in $IPD_{prev,b}$. If after the update $IPD_{mean,b}$ is still outside $[-\pi, \pi]$ the shift is reversed before the next iteration.

Now the bandwise IPD change, denoted as $IPD_{change,b}$, between the current bandwise $IPD_{t,b}$ and bandwise mean $IPD_{mean,b}$ is computed for each subband with

$$IPD_{change,b} = \left| IPD_{t,b} - IPD_{mean,b} \right| \quad (5.3\text{-}125)$$

with

$$IPD_{change,b} = 2\pi - IPD_{change,b}, \ if\ IPD_{change,b} > \pi \quad (5.3\text{-}126)$$

From the individual bandwise IPD changes in each subband a mean bandwise change, denoted as $IPD_{change,}$ over all subbands is computed:

$$IPD_{change} = \frac{\sum_{b=1}^{nBands} IPD_{change,b}}{nBands} \quad (5.3\text{-}127)$$

The mean bandwise IPD change is taken as an overall indication of the stability of the bandwise IPD in the current frame and is now used to force a similar level of stability on the global IPD estimate, i.e. to calculate a stabilized IPD estimate using the current global IPD estimate, the transmitted stabilized estimate of the previous frame and the mean bandwise IPD change. For small values of the mean bandwise IPD change (smaller than 0.3) the current global IPD is overwritten with the stabilized IPD estimate of the previous frame:

$$gIPD = gIPD_{prev}, \qquad if\ IPD_{change} < 0.3 \quad (5.3\text{-}128)$$

For larger values, a modulus of a difference between the transmitted IPD of the previous frame and the global IPD of the current frame, denoted as $gIPD_{diff}$, is computed*:*

$$gIPD_{diff} = \left| gIPD - gIPD_{prev} \right| \quad (5.3\text{-}129)$$

with

$$gIPD_{diff} = 2\pi - gIPD_{diff}, \ if\ gIPD_{diff} > \pi \quad (5.3\text{-}130)$$

If

$$gIPD_{diff} > IPD_{change} \quad (5.3\text{-}131)$$

which means that the modulus of the difference between the transmitted IPD of the last previous frame and the global IPD estimate of the current frame is larger than the mean bandwise IPD change, the maximum allowed difference to the previously transmitted IPD is limited to the mean bandwise IPD change, so that the global IPD is calculated as:

$$gIPD = gIPD_{prev} + IPD_{change}, \qquad if\ gIPD > gIPD_{prev} \quad (5.3\text{-}132)$$

or

$$gIPD = gIPD_{prev} - IPD_{change}, \qquad if\ gIPD < gIPD_{prev} \quad (5.3\text{-}133)$$

If, however,

$$gIPD_{diff} \leq IPD_{change}, \quad (5.3\text{-}134)$$

which means that the modulus of the difference between the previously transmitted IPD and the global IPD estimate of the current frame is equal to or smaller than the mean bandwise IPD change, the original global IPD estimate *gIPD* is used for the current frame.

5.3.2.4.7.2 IPD encoding scheme

In addition, a reference parameter is used to determine the IPD parameter encoding scheme. The reference parameter includes at least one of the signal characteristic parameter and the signal characteristic parameters of previous frame. The signal characteristic parameter is calculated as the correlation between left channel and right channel of the current frame. The signal characteristic parameters of previous frame include at least one of the correlations between left channel and right channel of previous frame, an ITD parameter of previous frame, a signal type of previous frame.

The correlation between left channel and right channel is obtained by using the following calculation formula:

$$corr = \sum_{b=0}^{N} \frac{[E_l(b) + E_r(b) + 2xD_r(b)]}{[E_l(b) + E_r(b) + 2\sqrt{D_r^2(b) + D_i^2(b)}]}$$

wherein

$$E_l(b) = \sum_{k=0}^{L} |L(k)|^2$$

$$E_r(b) = \sum_{k=0}^{L} |R(k)|^2$$

$$D_r(b) = \sum_{k=0}^{L} [L_r(k)\,xR_r(k) + L_i(k)xR_i(k)]$$

$$D_i(b) = \sum_{k=0}^{L} [L_i(k)\,xR_r(k) - L_r(k)xR_i(k)]$$

$$L(k) = \sum_{n=0}^{Length-1} x_L(n)xe^{-j\frac{2\pi xnxk}{L}}, 0 \leq k < L$$

and

$$R(k) = \sum_{n=0}^{Length-1} x_R(n)xe^{-j\frac{2\pi xnxk}{L}}, 0 \leq k < L$$

wherein $E_l(b)$ indicates an energy sum of left channel, $E_r(b)$ indicates an energy sum of right channel, $L_r(k)$ indicates a real part of a k[th] frequency value of left channel frequency domain signal, $R_r(k)$ indicates a real part of a k[th] frequency value of right channel frequency domain signal, $L_i(k)$ indicates an imaginary part of the k[th] frequency value of left channel frequency domain signal, $R_i(k)$ indicates an imaginary part of the k[th] frequency value of right channel frequency domain signal, L indicates a quantity of sub-band spectral coefficients, and N indicates a quantity of sub-bands, n indicates an index value of a time domain signal, k indicates an index value of a frequency domain signal, $Length$ indicates a frame length, $x_L(n)$ indicates left channel time domain signal, $x_R(n)$ indicates right channel time domain signal, $L(k)$ indicates a k[th] frequency value that is of left channel frequency domain signal and that is used to calculate the IPD parameter, and $R(k)$ indicates a k[th] frequency value that is of right channel frequency domain signal and that is used to calculate the IPD parameter, wherein $x_L(n)$ and $x_R(n)$ indicate sequences of real numbers.

If the correlation between the left channel and right channel of the current frame is greater than or equal to 0.75, the IPD parameter encoding scheme of the current frame is skipping encoding an IPD parameter; If the IPD parameter encoding scheme of the previous frame is skipping encoding an IPD parameter, and the signal type of previous frame is music,

the IPD parameter encoding scheme of the current frame is skipping encoding an IPD parameter. If the IPD parameter encoding scheme of the current frame is not skipping encoding an IPD parameter, the IPD parameter encoding scheme of the current frame is encoding sub-band IPD parameters of some or all of sub-bands of the current frame.

5.3.2.4.8 Calculation of the side and residual prediction gains

In case of the residual prediction gain is transmitted or if the residual is directly coded in the frequency band b currently considered, the side gain stereo parameter is computed as the optimal gain for predicting the side signal $S_i[k] = L_{i[k]} - R_i[k]$ by the mid signal $M[k] = L[k] + R[k]$, such that the energy of the remainder

$$p[k] = S[k] - g[b]M[k] \tag{5.3-135}$$

is minimal. The optimal prediction gain can be calculated from the energies in the subbands

$$E_L[\mathrm{b}] = \sum_{k\in l_b} |\ \mathrm{L[b]}|^{\ 2} \quad \text{and} \quad E_R[\mathrm{b}] = \sum_{k\in l_b} |\mathrm{R[b]}|^{\ 2} \tag{5.3-136}$$

and the absolute value of the inner product of L and R

$$X_{L/R}[b] = |\ \sum_{k\in l_b} L[k]R^*[k]\ | \tag{5.3-137}$$

as

$$g[b] = \frac{E_L[\mathrm{b}] - E_R[b]}{E_L[b] + E_R[b] + 2X_{L/R}[b]} \tag{5.3-138}$$

From this it follows that $g[b]$ lies in [-1,1].

In bands, where the residual gain is transmitted, the side gain is calculated similarly as previously defined. The residual gain is then computed form the side gain, the channel energies and the inner product of the channels as:

$$r[b] = \sqrt{\frac{(1-g[b])E_L[b] + (1+g[b])E_R[b] - 2X_{L/R}[b]}{reg[b] + E_L[b] + E_R[b] + 2X_{L/R}[b]}} \tag{5.3-139}$$

where $reg[b]$ is regularization factor obtained by adding a coherent low energy contribution to avoid singularity for very low energy signals:

$$reg[b] = (n(I_b) * NFFT_{32})^2 \tag{5.3-140}$$

where the $n(I_b)$ gives the cardinality of I_b, the set of frequency indexes laying in the frequency band b, and $NFFT_{32}$ is the size of the analysis DFT at 32 kHz.

It implies that

$$0 \le r[b] \le \sqrt{1 - g[b]^2}. \tag{5.3-141}$$

In particular, this shows that $r[b] \in [0,1]$. This way, the stereo parameters can be calculated independently from the downmix by calculating the corresponding energies and the inner product.

In the case of non-zero ΔITD and time alignment in the frequency domain, there is a mismatch in the DFT analysis windows between the two channels, which biases the calculation of the residual prediction gain. An offset is therefore calculated, involving a normalized autocorrelation of the DFT analysis windows, modeling the mismatch of the two windows after a circular shift of ΔITD samples. The normalized autocorrelation function is precalculated and stored in the table given in Table 5.3-17, for a resolution of 8 samples at 32 kHz. The delta ITD is then first converted to 32 kHz, and the corresponding offset is found by linear interpolation of the two Wn_table entries.

Based on this normalized autocorrelation function $\widehat{W}_X(n)$, the offset parameter $\hat{r}_t$ is calculated as:

$$\hat{r} \quad = \frac{2\ c}{c+1}\sqrt{2\frac{1-\widehat{W}_X(\Delta ITD)}{1+c^2+2\ c\ \widehat{W}_X(\Delta ITD)}}, \tag{5.3-142}$$

where

$$\mathrm{c} = \frac{E_L[b]}{E_R[b]}. \tag{5.3-143}$$

The residual gains r[b] is then corrected as given in the equation:

$$r[b] \leftarrow \max\{0, r[b] - \hat{r}\ \}. \quad (5.3\text{-}144)$$

Table 5.3-7: Normalized cross-correlation function of the analysis window at a resolution of 8 samples at 32 kHz

$\widehat{W}_X(\Delta ITD)$
1.0000000f, 0.9992902f, 0.9975037f, 0.9948399f, 0.9914063f, 0.9872797f, 0.9825207f, 0.9771799f, 0.9713015f, 0.9649245f, 0.9580846f, 0.9508143f, 0.9431443f, 0.9351030f, 0.9267174f, 0.9180131f, 0.9090145f, 0.8997447f, 0.8902261f, 0.8804801f, 0.8705271f, 0.8603868f, 0.8500779f, 0.8396186f, 0.8290262f, 0.8183170f, 0.8075067f, 0.7966103f, 0.7856416f, 0.7746140f, 0.7635394f, 0.7524292f, 0.7412935f, 0.7301411f, 0.7189796f, 0.7078147f, 0.6966495f, 0.6854842f, 0.6743189f, 0.6631536f, 0.6519884f, 0.6408231f, 0.6296578f, 0.6184926f, 0.6073273f, 0.5961620f, 0.5849968f, 0.5738315f, 0.5626662f, 0.5515010f

Finally, in case of active speech, which is detected if VAD flag is set and if speech is detected in the pre-processing step, the residual prediction gain is further limited for mitigating potential synthetic artefacts in the stereo upmix and ambience generation. First if ΔITD changed from the previous frame the residual prediction gain is replaced by the minimum between the present and previous gain:

$$r_i[b] \leftarrow \min\{r_{i-1}[b], r_i[b]\} \quad (5.3\text{-}145)$$

Then it is compared to an IIR filtered and smoothed gain threshold, used to clip the actual residual gain value to be used:

$$r'_i[b] = 0.9r'_{i-1}[b] + 0.1r_i[b] \quad (5.3\text{-}146)$$

$$r_i[b] \leftarrow \min\{1.1r'_i[b], r_i[b]\} \quad (5.3\text{-}147)$$

In case neither residual gain nor residual coding are used in the frequency band, the side gain is computed based the Inter-channel Level difference, which is more psycho-acoustically motivated and quasi optimal assuming that the channels are time and phase aligned. The side gain is then given by:

$$g[b] = \frac{c-1}{c+1}, \quad (5.3\text{-}148)$$

where

$$c = \frac{E_L[b]}{E_R[b]}. \quad (5.3\text{-}149)$$

5.3.2.4.9 Stereo parameter coding

The global IPD is quantized on 4 bits in active frames and on 2 bits on SID frames, by the following uniform quantization of the angle:

$$gIPD_{index} = \left\lfloor \frac{gIPD+\pi}{deltaIPD}' + 0.5 \right\rfloor \quad (5.3\text{-}150)$$

where $deltaIPD = 2\pi/gIPD_{index\_max}$, where $gIPD_{index\_max} = 2^{gIPD_{bits}}$.

The index is coded in modulo 2π, and an index of 16 is then converted to index 0:

$$gIPD_{index} = 0 \text{ if } gIPD_{index} = gIPD_{index\_max} \quad (5.3\text{-}151)$$

The dequantized global IPD value is then used in the active downmix and computes as follows:

$$\widehat{gIPD} = gIPD_{index}.deltaIPD - \pi \quad (5.3\text{-}152)$$

The global IPD index $gIPD_{index}$ is written in raw coding in the bitstream.

The quantization of side and residual gains is done jointly, since there is a strong dependence of the residual gain on the side gain, since the latter determines the range of the first. Quantizing the side gain g and the residual gain r independently by choosing quantization points in [−1, 1] and [0, 1] is therefore inefficient, since the number of possible

quantization points for r would decrease as g tends towards ±1. The joint quantization is done by first computing the inherent absolute ILD, using the following equality:

$$|ILD| = 10\log_{10}(\frac{(1+|g|)^2+r'^2}{(1-|g|)^2+r'^2}), \quad (5.3\text{-}153)$$

where is the bounded residual prediction gain r', is bounded between 0 and the theoretical maximum depending on the side gain:

$$r' = \min(r, \sqrt{1-g^2}) \quad (5.3\text{-}154)$$

The resulting absolute ILD $|ILD|$ is bounded between 0 and 50 dB, before being quantized on the 16 values defined as follows:

±{0,2,4,6,8,10,13,16,19,22,25,30,35,40,45,50},

Once the quantized level ILD_{level} of the absolute ILD is found, a 2-dimensional 8-points quantization is selected for quantization the absolute value g and the bounded value of r'. This gives rise to an overall codebook with 256 entries, which is organized as a 16 × 8 tables of quantization points holding the values corresponding to non-negative values of g and r' and a sign bit for the sign of g (see Table 5.3-8). This gives rise to a 8 bit integer representation of the quantization points (g, r') where the first bit specifies the sign of g, the next four bits holding the row index in the 16 × 8 table and the last three bits holding the column index. Quantization of (|g|, r'') is done by an exhaustive codebook search by minimizing the mean squared error:

$$gains_{index} = \underset{i}{\text{argmax}}\ ((|g| - gains_{cdbk}[8 * ILD_{level} + i][0])^2 + (r' - gains_{cdbk}[8 * ILD_{level} + i][1])^2) \quad (5.3\text{-}155)$$

Table 5.3-8: Codebook for the joint quantization of the absolute side gain and the residual prediction gain function of the absolute ILD level

$\|ILD\|$=0	$\|ILD\|$=2	$\|ILD\|$=4	$\|ILD\|$=6	$\|ILD\|$=8
0.000000,0.000000	0.114623,0.000000	0.226274,0.000000	0.332279,0.000000	0.430506,0.000000
0.000000,0.116982	0.116171,0.115424	0.229210,0.110915	0.336318,0.103909	0.435293,0.095060
0.000000,0.226991	0.120448,0.22385	0.237306,0.214802	0.347423,0.200786	0.448405,0.183172
0.000000,0.340693	0.127733,0.335704	0.251046,0.321340	0.366155,0.299242	0.470357,0.271705
0.000000,0.464549,	0.138966,0.45714	0.272098,0.435947	0.394585,0.403641	0.503282,0.363897
0.000000,0.605079	0.155840,0.59427	0.303429,0.563535	0.436296,0.517359	0.550751,0.461614
0.000000,0.776250	0.182248,0.760034	0.351766,0.714464	0.499282,0.647470	0.620606,0.568856
0.000000,1.000000	0.226274,0.974064	0.430506,0.902588	0.598480,0.801138	0.726386,0.687287
$\|ILD\|$=10	$\|ILD\|$=13	$\|ILD\|$=16	$\|ILD\|$=19	$\|ILD\|$=22
0.519494,0.000000	0.634158,0.000000	0.726386,0.000000	0.798235,0.000000	0.852825,0.000000
0.524665,0.085097	0.639318,0.069554	0.731048,0.054862	0.802164,0.042077	0.855976,0.031585
0.538769,0.163459	0.653296,0.132950	0.743597,0.104384	0.812683,0.079739	0.864374,0.059662
0.562182,0.241193	0.676201,0.194597	0.763914,0.151643	0.829536,0.115098	0.877717,0.085671
0.596843,0.320512	0.709442,0.255549	0.792863,0.197003	0.853180,0.148173	0.896203,0.109492
0.645875,0.402058	0.755149,0.315316	0.831670,0.239542	0.884212,0.178009	0.920064,0.130302
0.716076,0.487487	0.818046,0.373555	0.883261,0.278217	0.924333,0.203506	0.950256,0.147176
0.818182,0.574960	0.904547,0.426375	0.950993,0.309212	0.975135,0.221614	0.987460,0.157870
$\|ILD\|$=25	$\|ILD\|$=30	$\|ILD\|$=35	$\|ILD\|$=40	$\|ILD\|$=45
0.893520,0.000000	0.938693,0.000000	0.965056,0.000000	0.980198,0.000000	0.988816,0.000000
0.895958,0.023331	0.940202,0.013738	0.965951,0.007932	0.980717,0.004528	0.989113,0.002568
0.902435,0.043958	0.944192,0.025807	0.968314,0.014873	0.982085,0.008481	0.989895,0.004807
0.912657,0.062866	0.950441,0.036736	0.971997,0.021112	0.984212,0.012019	0.991109,0.006806
0.926684,0.079898	0.958923,0.046392	0.976963,0.026561	0.987068,0.015088	0.992735,0.008532
0.944559,0.094403	0.969577,0.054375	0.983149,0.030984	0.990608,0.017552	0.994746,0.009911
0.966814,0.105673	0.982605,0.060264	0.990633,0.034143	0.994866,0.019278	0.997156,0.010865
0.993695,0.112114	0.998002,0.063182	0.999368,0.035554	0.999800,0.019998	0.999937,0.011246
$\|ILD\|$=50				
0.993695,0.000000 0.993864,0.001451 0.994308,0.002715 0.994996,0.003842 0.995917,0.004813				

0.997054,0.005586 0.998414,0.006117 0.999980,0.006324				

The sign of side gain and ILD_{level} are combined in a 5-bit wise parameter. The parameter can be coded either in raw coding, or using an adaptive Golomb-rice coding up to order 3, and an inter-frame delta coding. The optimal coding is selected are signalling in the bitstream.

The 3-bit wise column index is coded separately. The parameter can also be coded using different coding strategies: in raw coding, an adaptive Golomb-Rice coding up to order 2, or an inter-frame delta coding. Best and selected coding is signaled in the bitstream. The raw coding and the adaptive Golomb-Rice coding represent two variants of an absolute coding mode, while the inter-frame delta coding is a predictive coding mode.

After scalar quantization is used to produce the index, the adaptive Golomb-Rice coding, in addition to checking different values for the Golomb-Rice order, also adapts the order of the symbols to be encoded based on the value of the index to be encoded. The reordering map is used to create a reordered value of the input index, *map[in]*. The reordered index is then subjected to the process of Golomb Rice encoding in order to produce a Golomb Rice code of the reordered index. In other words, reordering or remapping is first applied to the input index *in* before Golomb Rice encoding instead of directly creating the Golomb Rice code from the input index *in*. After the reordering and encoding of an input index, the reordering map is changed to another reordering map for use with a following input index, where the other reordering map is determined dependent on the input index (the index previous to the following index). Table 5.3-7 shows the reordering maps for 8 symbols (or indexes), where the first column depicts values of an input index, and the second column contains an associated array which determines the reordering to be used for a following input index. The value of the input index determines the reordering map for the following input index. For example, for an input index of value 1 the reordering map for the following input index is given by the array {2, 0, 1, 3, 4, 5, 6, 7}. I.e. a following index value of zero is reordered to a value of 2, a following index value of 1 is reordered to a value of 0,a following index value of 2 is reordered to a value of 1 etc. The reordering map for the first input index is determined by the initial mapping array. The reordering mappings for 32 number of symbols is presented in the following table. The initial mapping is separately mentioned for 8 symbols and the one corresponding to input symbol 14, for the 32 symbols case.

Table 5.3-9: Reordering mapping for 8 symbols

Input symbol	Mapping for next input symbol, *map*[]
0	0, 1, 2, 3, 4, 5, 6, 7,
1	2, 0, 1, 3, 4, 5, 6, 7,
2	6, 2, 0, 1, 3, 4, 5, 7,
3	7, 5, 2, 0, 1, 3, 4, 6,
4	7, 6, 4, 3, 1, 0, 2, 5,
5	7, 6, 5, 3, 2, 1, 0, 4,
6	7, 6, 5, 4, 3, 2, 0, 1,
7	7, 6, 5, 4, 3, 2, 0, 1,
Initial mapping	6, 5, 4, 3, 1, 0, 2, 7,

Table 5.3-10: Reordering mapping for 32 symbols

In	Mapping for next input symbol, *map*[]
0	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
1	1, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
2	15, 4, 0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
3	12, 9, 4, 1, 0, 2, 3, 5, 6, 7, 8, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
4	16, 14, 8, 4, 2, 0, 1, 3, 5, 6, 7, 9, 10, 11, 12, 13, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
5	18, 16, 14, 10, 5, 0, 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 15, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
6	21, 19, 17, 15, 8, 4, 2, 0, 1, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 16, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30
7	21, 19, 17, 15, 12, 8, 4, 0, 1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 16, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30
8	21, 19, 17, 15, 13, 11, 9, 3, 0, 1, 2, 4, 5, 6, 7, 8, 10, 12, 14, 16, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30
9	24, 22, 20, 18, 16, 14, 12, 9, 6, 0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 13, 15, 17, 19, 21, 23, 25, 26, 27, 28, 29, 30
10	25, 23, 21, 19, 17, 15, 13, 11, 9, 6, 0, 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 27, 28, 29, 30
11	27, 25, 23, 21, 19, 17, 15, 13, 11, 8, 5, 0, 1, 2, 3, 4, 6, 7, 9, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 29, 30
12	26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2, 1, 0, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 28, 29, 30
13	28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2, 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 30
14	29, 27, 25, 23, 21, 19, 17, 15, 13, 11, 9, 7, 5, 3, 0, 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30
15	29, 27, 25, 23, 21, 19, 17, 15, 13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30
16	30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2, 1, 0, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29
17	30, 29, 27, 25, 23, 21, 19, 17, 15, 13, 11, 9, 7, 5, 3, 1, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28
18	30, 29, 28, 27, 25, 23, 21, 19, 17, 15, 13, 11, 9, 7, 5, 3, 0, 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26
19	30, 29, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 9, 7, 6, 4, 3, 2, 1, 0, 5, 8, 11, 13, 15, 17, 19, 21, 23, 25, 27
20	30, 29, 28, 27, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 7, 5, 4, 3, 2, 1, 0, 6, 9, 11, 13, 15, 17, 19, 21, 23, 25
21	30, 29, 28, 27, 26, 25, 23, 21, 19, 17, 15, 13, 11, 10, 8, 7, 5, 4, 3, 2, 1, 0, 6, 9, 12, 14, 16, 18, 20, 22, 24
22	30, 29, 28, 27, 26, 25, 24, 23, 22, 20, 18, 16, 14, 12, 10, 8, 7, 6, 5, 4, 2, 1, 0, 3, 9, 11, 13, 15, 17, 19, 21
23	30, 29, 28, 27, 26, 25, 24, 23, 22, 20, 18, 16, 14, 13, 11, 10, 9, 7, 6, 5, 3, 2, 1, 0, 4, 8, 12, 15, 17, 19, 21
24	30, 29, 28, 27, 26, 25, 24, 23, 22, 20, 18, 16, 14, 13, 12, 11, 10, 9, 7, 6, 5, 3, 1, 0, 2, 4, 8, 15, 17, 19, 21
25	30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 17, 15, 13, 12, 11, 9, 8, 7, 6, 4, 3, 2, 1, 0, 5, 10, 14, 16, 18
26	30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 15, 13, 12, 11, 10, 9, 7, 6, 5, 3, 1, 0, 2, 4, 8, 14, 16
27	30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 11, 10, 8, 7, 6, 5, 3, 2, 0, 1, 4, 9, 12
28	30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 3, 2, 1, 0, 4, 15
29	30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 0, 1
30	30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

When encoding with adaptive Golomb Rice the indexes corresponding to the residual prediction gain for each sub band, a special case is considered by verifying first if the value for the first sub band quantization index is zero or not. If the index values in all the sub bands of the frame are not all zero, the encoding is reverted to fixed rate raw encoding of the indexes and the use of fixed rate raw encoding is signalled. If the index values of all sub bands are zero then entropy coding is used by encoding with the above adaptive Golomb Rice coding procedure only the index, that has value 0, corresponding to the first sub band and encoding is stopped for the indexes associated with the other sub bands of the frame. By encoding this value as index corresponding to the first sub band, the decoder will be informed that the indexes corresponding to all sub bands are zero.

Since the side gain and residual prediction gain generally varies slowly between frames, the inter-frame delta coding mode usually yields the lowest bit rate. However, this predictive scheme suffers from error propagation in case of packet loss. For this reason, the number of consecutive predictive encoding modes is limited to $N_{PREDMAX} + 1 = 11$. The flag $M_{try-diff}$ is determined to indicate whether predictive coding is allowed. It is initialized to

$$M_{try-diff} = sp\_aud\_decision0 \tag{5.3-156}$$

where $sp\_aud\_decision0$ is the primary speech/music flag as defined in 5.2.2.2.11. The flag is then updated according to

$$M_{try-diff} := \begin{cases} 0, & N_{gpred} = N_{PREDMAX} \\ M_{try-diff}, & otherwise \end{cases} \tag{5.3-157}$$

Where N_{gpred} is a counter of consecutive frames of predictive side gain encoding mode. If the counter N_{gpred} has not reached the maximum number of predictive frames $N_{PREDMAX}$, it means $N_{gpred} < N_{PREDMAX}$ and the value of $sp\_aud\_decision0$ is kept. However, if the maximum number of predictive frames $N_{PREDMAX}$ has been reached, number of allowed predictive frames is exceeded and $M_{try-diff}$ is cleared. Finally, if the previous frame was a NO_DATA frame or a SID frame, the predictive coding is disabled according to

$$M_{try-diff} := \begin{cases} 0, & R_{core}^{[-1]} \leq 2400 \\ M_{try-diff}, & otherwise \end{cases} \quad (5.3\text{-}158)$$

where $R_{core}^{[-1]}$ is the core bit rate for the previous frame and $R_{core}^{[-1]} \leq 2400$ indicates the last frame was either a NO_DATA frame or a SID frame. In the case of $M_{try-diff} = 0$, the counter N_{gpred} is reset to zero and the absolute coding mode is selected and set to be used. The required number of bits for absolute encoding using the variable rate AGR B_{AGR} is calculated. The fallback solution is to encode each index using 5 bits, which yields $B_{ABS} = 5N_{bands}$ bits. If the predictive coding mode is allowed, i.e. $M_{try-diff} = 1$, the number of required bits for the predictive coding scheme B_{PRED} is obtained. A bit rate difference is then obtained according to

$$B_{diff} = \min(B_{ABS}, B_{AGR}) - B_{PRED} \quad (5.3\text{-}159)$$

The bit rate difference B_{diff} represents the additional number of bits to encode the side gain using an absolute coding scheme like the AGR or the fallback binary index encoding and is typically larger than zero. The bit rate difference is low-pass filtered across frames according to

$$B_{diff,LP} = 0.06B_{diff} + 0.94B_{diff,LP}^{[-1]} \quad (5.3\text{-}160)$$

Then, the predictive mode is selected and set to be used if the following condition is met

$$B_{diff} > 0.8B_{diff,LP}\, N_{gpred}/(N_{PREDMAX} + 1) \quad (5.3\text{-}161)$$

where N_{gpred} is the number of consecutive frames using predictive coding mode for the side gain. Since N_{gpred} is incremented before it is used, the effective maximum number of predictive frames is $N_{PREDMAX} + 1$. Further, the predictive mode counter is incremented $N_{gpred} := N_{gpred} + 1$. If the condition in (5.3-161) is not met, the best performing absolute coding mode is selected and set to be used, meaning the one yielding the lowest number of bits $\min(B_{ABS}, B_{AGR})$ out of the AGR and the fallback binary index coding. The predictive mode counter N_{gpred} is also reset to zero.

5.3.2.4.10 Active Downmix

The global IPD if used is first compensated by rotating the Left channel and leave untouched the Right channel. It is achieved as follows:

$$\begin{cases} L'_i[\mathrm{k}] = L_i[k].\, e^{-j2\pi.\overline{gIPD}} \\ R'_i[\mathrm{k}] = R_i[k] \end{cases} \quad (5.3\text{-}162)$$

The so-obtained globally phase-aligned channels are further processed, and depending on the frequency band and if the residual coding is used in this band. For frequency bands where the residual coding will be applied a passive downmix and a prediction is used to compute the Mid and Side channels respectively. A sum difference transformation is first performed on the time and phase aligned spectra of the two channels in a way that the energy is conserved in the Mid signal.

$$\begin{cases} M_i[k] = 0.5 * (L'_i[k] + R'_i[k]) \\ S_i[k] = 0.5 * (L'_i[k] - R'_i[k]) \end{cases} \quad (5.3\text{-}163)$$

The Side channel is then predicted from the Mid channel and the transmitted stereo parameter side gain:

$$Res_i[\mathrm{k}] = S_i[k] - g[b].\, M_i[k] \quad (5.3\text{-}164)$$

The residual signal is further attenuated in case a delta ITD compensation was performed in the frequency domain. This is done to counter the window mismatch engendered by the time shift.

$$Res'_i[\mathrm{k}] = Res_i[\mathrm{k}] * \min(1.0, (\max(0.2(2.6 - 0.02 * |\Delta ITD_{32}|)) \quad (5.3\text{-}165)$$

where ΔITD_{32} is the delta ITD applied in the current frame reported at 32 kHz.

For frequency bands, where no residual coding is performed, an active downmixing is performed by a controlled energy-equalization of the sum of the left and right channels mixed with a complementary signal, which is selected as being the left channel. The complementary signal, i.e. left channel, is there to avoid singularities during the sum, when the left and right channels are almost coherent and out-of-phase. The energy-equalization of the sum is controlled for avoiding problems at the singularity point but also to minimize significantly signal impairments due to large

fluctuations of the gain. The complementary signal is there to compensate the remaining energy loss or at least a part of it. The general form of the downmix for a given frequency band index *k* is be expressed as:

$$M[k] = W_1[b3](L[k] + R[k]) + W_2[b3]L[k] \quad (5.3\text{-}166)$$

where *b3* is the index of the frequency band of the partition used to compute the mixing factors and to which frequency index *k* belongs.

The downmixing generates the sum channel L+R as it is done in conventional passive and active downmixing approaches, where the gain $W_1[b3]$ aims at equalizing the energy of the sum of the channels for matching the average energy of the input channels or a regularization of it. However, unlike conventional active downmixing approaches, $W_1[b3]$ is limited to avoid instability problems and to avoid that the energy relations are restored based on an impaired sum of channels. The mixing with a complementary signal is there to avoid that the energy vanishes when $L[k]$ and $R[k]$ are out-of-phase. $W_2[b3]$ compensates the energy-equalization due to the limitation introduced in $W_1[b3]$. The downmixing can then be rewritten as:

$$M[b] = W_L[b3]L[k] + W_R[b3]R[k] \quad (5.3\text{-}167)$$

where

$$W_R[b3] = W_1[b3] = \frac{\sqrt{(\sum_{k\in Ib3}|L[k]|^2 + \sum_{k\in Ib3}|R[k]|^2)(1+\alpha[b2])}}{2\cdot(\sum_{k\in Ib3}|L[k]| + \sum_{k\in Ib3}|R[k]|)} \quad (5.3\text{-}168)$$

and

$$W_L[b3] = W_R[b3] + \left(1 - \frac{\sum_{k\in Ib3}|L[k]+R[k]|}{\sum_{k\in Ib3}|L[k]| + \sum_{k\in Ib3}|R[k]|}\right) \quad (5.3\text{-}169)$$

where $\alpha[b2]$ is a band-wise parameter running on the frequency band partition *b2*, and used to stabilize and regularize the target energy of the mixed signal. It is computed as:

$$\alpha[b2] = \frac{\sqrt{\sum_{i=0}^{-2}\sum_{k\in Ib2} L_i[k]R_i^*[k]}}{\sum_{i=0}^{-2}\sum_{k\in Ib2}|L_i[k]|^2 + \sum_{i=0}^{-2}\sum_{k\in Ib2}|R_i[k]|^2} \quad (5.3\text{-}170)$$

by average energies and inner product norm over last three frames.

5.3.2.4.11 STFT synthesis

From the downmixed spectrum M, a time domain signal is synthesized by an inverse DFT:

$$m_i[n] = \sum_{k=0}^{N-1} M_i[k] \cdot e^{\frac{2\pi jkn}{N}}, \quad \text{for } 0 \le n < N \quad (5.3\text{-}171)$$

Finally, an overlap-add operation allows reconstructing a frame of M samples:

$$m'[i\cdot \mathrm{M} + n - zp] = \begin{cases} m_{i-1}[M+n]\cdot w_{syn}[M+n] + m_i[n]\cdot w_{syn}[n], & \text{for } zp \le n < zp + L \\ m_i[n], & \text{for } zp + L \le n < zp + M \end{cases} \quad (5.3\text{-}172)$$

The synthesis window is not the same as analysis window, but are together complementary in energy. The synthesis window depicted in Figure 5.3-28.

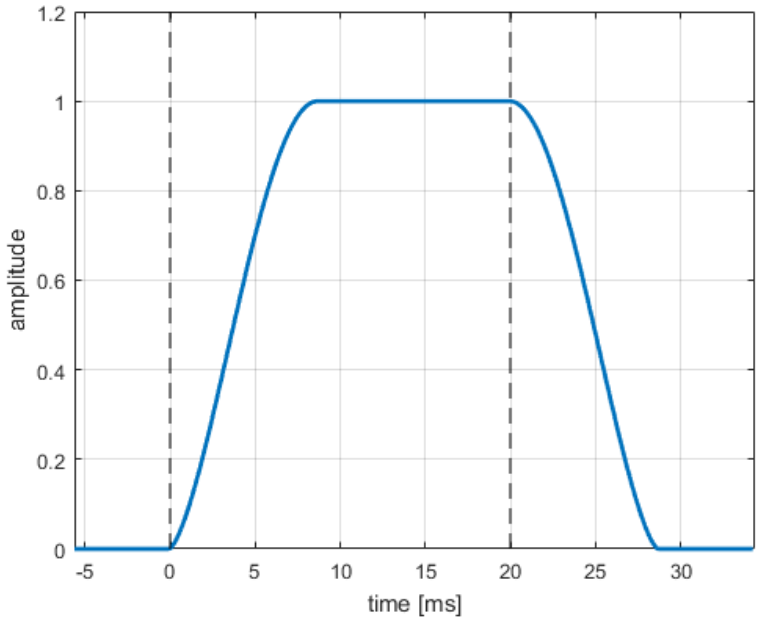


Figure 5.3-28: STFT synthesis window with a stride of 20 ms and an overlapping of 8.75 ms

The overlapping window part is 8.75 ms and corresponds to the lookahead needed for the pre-processing and LP analysis used by the core-code. It also corresponds to overlapping region of the MDCT used in TCX and HQ-Core. The synthesis window is defined as:

$$w_{syn}[n] = \begin{cases} \left(sin(\frac{\pi.(n - zp + 0.5)}{2.L})\right)^{1.5} & for\ zp \le n < zp + L \\ 1, for\ zp + L \le n < M \\ \left(\sin\left(\frac{\pi.(n - zp + L - M + 0.5)}{2.L}\right)\right)^{1.5} & for\ M \le n < M + L \end{cases} \quad (5.3\text{-}173)$$

and zero otherwise.

For reducing the delay engendered by overlapping region with the next frame, corresponding to the lookahead used in the pre-processing steps and exploited by the core-coders, is already estimated by an un-windowing of the solely current synthesized frame, without relying on the following frame. For this the inverse of the analysis is applied to the right part of the synthesis window, as follows:

$$m'[i \cdot \mathrm{M} + n - zp] = \frac{m_i[n] - d}{w_{ana}[n]} + d, n \in zp + [M, M + L[\quad (5.3\text{-}174)$$

where *d* is an approximated DC offset for avoiding instabilities and large values engendered by inverting the analysis window, especially at the right overlapping region bound, corresponding also to the lookahead bound. This un-windowing adaptation dependent on the characteristic of the processed synthesized signal m_i, which is used to approximate the DC offset by taking one of his values within the right zero padding range of the analysis window after processing and inverse DFT. The approximated DC offset is the expressed as:

$$d = m_i[zp + L + M] \quad (5.3\text{-}175)$$

The size of the inverse DFT and therefore the output sampling rate of the STFT synthesis is depending on the modules requiring the downmixed signal for the subsequent processing and the core-coding. By truncating or zero-padding the STFT spectrum, and using the appropriate normalization, a resampling is achieved allowing conveying the same downmixed signal at different modules and at different sampling-rates. The output sampling-rates can be different from the input sampling-rate of the input stereo signal used in the STFT analysis. The supported sampling-rates are 12.8 and 16 kHz when conveying and resampling the downmix signal to the core-coder ACELP. For the MDCT core-coders, the output sampling-rate can the 16, 32 and 48 kHz, while being at 16 kHz for high-band downmixed signal used in the Time-Domain BWE of ACELP.

5.3.2.4.12 Residual coding

5.3.2.4.12.1 Overview

The residual coding is achieved after synthesis the signal $Res'_i[\mathrm{k}]$ back in time-domain at a sampling-rate of 8 kHz through the inverse DFT. No overlap-adding is realized since the windowed synthesized signal is directly transformed by a forward MDCT after applying the same windows as the analysis STFT window to achieve an analysis MDCT window equivalent to a sine window.

The residual coding operates in the MDCT domain at a target quantization SNR, specified together with the maximum number of bits that are allowable for each frame. If the target SNR requires a larger number of bits than the maximum specified, the SNR is gradually decreased so that the actual number of bits will satisfy the bit constraint.

The target SNR is derived from the psychoacoustic consideration that quantization errors are more perceptible if the restored stereo channels are out-of-phase. Therefore, the target SNR is made dependent on an out-of-phase estimator. Considering the stereo upmix, the left are right channels are simplistically generated by a mid signal, a side gain as well a residual signal,

$$\begin{cases} L[k] = M[k] + g[b]M[k] + R[k] \\ R[k] = M[k] - g[b]M[k] - R[k] \end{cases} \quad (5.3\text{-}176)$$

Since $|g[b]| < 1$, one can consider that the components $M[k] - g[b]M[k]$ and $M[k] + g[b]M[k]$ are always in-phase, while $+R[k]$ and $-R[k]$ are obviously out-of-phase. The out-of-phase ratio retained is the maximum out-of-phase ratio among the two channels:

$$oop_{ratio}[b] = \max\left(\frac{E_{R[b]}}{(1-g[b])^2 E_{M[b]}+E_{R[b]}}, \frac{E_{R[b]}}{(1+g[b])^2 E_{M[b]}+E_{R[b]}}\right) = \frac{E_{R[b]}}{(1-|g[b]|)^2 E_{M[b]}+E_{R[b]}} \quad (5.3\text{-}177)$$

given $|g[b]| < 1$. The in-phase ratio is the 1-complementary of oop_{ratio}, and can be expressed as:

$$ip_{ratio}[b] = \frac{(1-|g[b]|)^2 E_{M[b]}}{(1-|g[b]|)^2 E_{M[b]}+E_{R[b]}} \quad (5.3\text{-}178)$$

The target SNR is the maximum of the interpolations calculated for each frequency band between a SNR of 10 dB for in-phase components and 40 dB for out-of-phase components:

$$SNR_{target} = \max_b(10.\, ip_{ratio}[b] + 40.\, ip_{ratio}[b]) \quad (5.3\text{-}179)$$

The real-valued MDCT coefficients of the residual signal is truncated above a maximum frequency given as input configuration parameter, and form the $input$ vector. The output of the encoder is the global gain index $ggi \in \{0, \ldots, 127\}$ and the entropy coded bits generated by the arithmetic coder. The target SNR is achieved by choosing a suitable global gain index ggi, which is then used to quantize the $input$ vector into the integer q_input vector that will be entropy coded.

The global gain index ggi is dequantized to the global gain gg by the relation

$$gg = dequantize\_gain(gg) = 10^{\frac{90}{20 \cdot 127} ggi}, \text{ for } ggi \in \{0, \ldots, 126\} \quad (5.3\text{-}180)$$

and the global gain gg is quantized to the global gain index ggi by the relation

$$ggi = quantize\_gain(gg) = \left\lfloor \frac{20 \cdot 127}{90} \log_{10} gg + 0.4898 \right\rfloor, \text{ for } gg \in [1, 29145] \quad (5.3\text{-}181)$$

where 0.4898 is used for rounding instead of 0.5 to achieve optimal mean-squared-error reconstruction.

The special global gain index value 127 is used to indicate that all values in the q_input vector are zero. The global gain index is coded raw using 7 bits, and it is always placed before the entropy coded bits, therefore using the special value 127 signals there are no entropy coded bits to follow.

The $input$ vector is converted to the quantized q_input vector using the dequantized global gain gg derived from the chosen global gain index ggi by the relation

$$q\_input[i] = \left\lfloor \frac{input[i]}{gg} + 0.5 \right\rfloor, \text{ for } i \in \{0, \ldots, N-1\} \quad (5.3\text{-}182)$$

which represents scaling by gg and uniform scalar quantization with rounding to the nearest integer.

Let the block on position $k \in \{0, \dots, \left\lceil \frac{N}{8} \right\rceil - 1\}$ be extracted as $block[k][i] = q\_input[8 \cdot k + i]$, for $i \in \{0, \dots, blk\_length[k] - 1\}$, where the block sizes are $blk\_length[k] = \min(8, N - 8 \cdot k)$. For each block, a parameter $param[k] \in \{0, 15\}$ which identifies the model used for coding the block is selected and coded as side information. The value $param[k] = 0$, indicating the very low entropy case, uses a model which allows for coding of $nz\_count \in \{0, \dots, 3\}$ nonzero values of ± 1, while the rest of the values are 0. This includes the case where all the values in the block are 0. The values $param[k] \geq 1$ use a model assuming the values are generated by a Laplace distribution with scale parameter $2^{param[k]-1}$.

The parameter value $param[k] = 0$ can be used to code only the blocks that satisfy the corresponding model constraints, the other parameter values can encode any arbitrary block, however with different number of bits. For a block, the encoder selects the optimal parameter from those that can be used to code it, such that the total number of bits for coding both the parameter and the block is minimized.

Entropy coding for $param[k] = 0$ of a block starts by coding nz_count, the number of nonzero values of ± 1, with raw coding using 2 bits. Then, the nonzero mask is coded, which contains nz_count ones and $blk\_length - nz\_count$ zeros, with raw coding of the sign bits of the nonzero positions.

```
encode_low_entropy_block(block, blk_length)
{
  nz_count = 0
  for (i = 0; i < blk_length; i++)
  {
    if (block[i] != 0)
    {
      nz_count++
    }
  }

  rc_uni_enc_encode_bits(nz_count, 2)
  left_1 = nz_count
  left_0 = blk_length - nz_count

  for (i = 0; i < blk_length; i++)
  {
    val = block[i]

    if ((left_0 == 0) || (left_1 == 0))
    {
      /* only ones left or only zeros left */
    }
    else
    {
      count_0 = left_0 * ECSQ_tab_inverse[left_0 + left_1]
      rc_uni_enc_encode_bit_prob_fast(abs(val), count0, 14)
    }

    if (val != 0)
    {
      rc_uni_enc_encode_bits(get_sign(val), 1)
      left_1--
    }
    else
    {
      left_0--
    }
  }
}
```

The precomputed table is computed as $ECSQ\_tab\_inverse[k] = \left\lfloor \frac{2^{14}}{k} + 0.5 \right\rfloor$, for $k \in \{1, \dots, 8\}$. Also, a helper function is used to obtain the sign bit, $get\_sign(x) = 1$, if $x < 0$ and 0, if $x \geq 0$.

During coding of a block, if there are only ones or zeros left, the nonzero mask is already determined. Otherwise, the mask is coded with an adaptive probability model giving the probability of a zero as $p_0 = \frac{left_0}{left_0 + left_1}$. This probability is approximately mapped to a 14-bit frequency count, without using a division operation as $count_0 = \left\lfloor 2^{14} * \frac{left_0}{left_0 + left_1} + 0.5 \right\rfloor \approx left_0 * \left\lfloor \frac{2^{14}}{left_0 + left_1} + 0.5 \right\rfloor$, where both $left_0 \neq 0$ and $left_1 \neq 0$, and the second term in the approximation is available in a precomputed table. The value of $count_1$ is derived implicitly from the relation $count_0 + count_1 = 2^{14}$.

The obtained code length in bits of the nonzero mask is exactly the same as would be obtained by optimal combinatorial coding, which would use $\log_2 \left\langle {8 \atop nz\_count} \right\rangle = \log_2 \frac{8!}{nz\_count! * (8 - nz\_count)!}$ bits.

Entropy coding with $param[k] \geq 1$ of a block starts by computing $shift = \max(0, param[k] - 3)$, which represents the number of least significant bits of the absolute values that are coded approximately uniformly or with raw coding. The most significant bits of each absolute value are coded using a probability model selected by $param[k]$ together with escape coding. For $shift \leq 4$, coding of LSBs takes into account that for absolute values the probability of zero (0 maps to one value) is half of the probability of nonzeros (1 maps to two values, ± 1). For larger shifts, the length difference is negligible and raw coding is used using $shift$ bits. Finally, if the value is nonzero, the sign is coded raw.

```
encode_normal_block(block, blk_length, param)
{
  shift = max(0, param - 3)

  for (i = 0; i < blk_length; i++)
  {
    val = block[i]
    sym = abs(val)

    if (shift != 0)
    {
      lsbs = sym & ((1 << shift) – 1)
      sym = sym >> shift

      arith_encode_prob_escape(ECSQ_tab_vals[param – 1], 16, sym)

      if ((sym > 0) || (shift > 4))
      {
        rc_uni_enc_encode_bits(lsbs, shift)
      }
      else /* (sym == 0) && (shift <= 4) */
      {
        rc_uni_enc_encode_symbol_fast(lsbs, ECSQ_tab_abs_lsbs[shift], 14)
      }
    }
    else
    {
      arith_encode_prob_escape(ECSQ_tab_vals[param – 1], 16, sym)
    }

    if (val != 0)
    {
      rc_uni_enc_encode_bits(get_sign(val), 1)
    }
  }
}
```

The encoding of the entire quantized q_input vector can be expressed in terms of the previous two functions, which encode low entropy blocks and normal blocks, together with a helper function *find_optimal_parameter*, which computes for a block the optimal parameter to use for encoding.

```
encode_raw_vector(q_input, N)
{
  block_cnt = (N + 7) / 8
  for (k = 0; k < block_cnt; k++)
  {
    blk_length[k] = min(8, N – 8 * k)

    for (i = 0; i < blk_length[k]; i++)
    {
      block[k][i] = q_input[8 * k + i]
    }

    param[k] = find_optimal_parameter(block[k], blk_length[k])
    rc_uni_enc_encode_symbol_fast(param[k], ECSQ_tab_param, 14)

    if (param[k] == 0)
    {
      encode_low_entropy_block(block[k], blk_length[k])
    }
    else
    {
      encode_normal_block(block[k], blk_length[k], param[k])
    }
```

```
    }
}
```

5.3.2.4.12.2 Adaptive residual signal encoding

5.3.2.4.12.2.1 Adaptive residual signal encoding parameter

For the 32 kbps WB coding mode, an adaptive residual signal encoding parameter is used to determine whether to encode the residual signals of the M sub-bands in the current frame. The residual signal encoding parameter is calculated based on downmixed signal energy and residual signal energy of each of M sub-bands in the current frame, wherein spectral coefficients of the current frame are divided to obtain N sub-bands, the M sub-bands are at least some of the N sub-bands, N is a positive integer greater than 1, $M \leq N$, and M is a positive integer. The residual signal encoding parameter of the current frame is determined based on the dmx_res_all, frame_nrg_ratio and res_dmx_ratio_lt. The residual signal encoding parameter res_cod_mode_flag is calculated according to

$$\text{res\_cod\_mode\_flag} = \begin{cases} 1, \text{res\_dmx\_ratio\_lt} > 0.01 \\ 0, else \end{cases}$$

when res_cod_mode_flag is equal to 1 means encode the residual signals, otherwise do not encode the residual signals. The res_dmx_ratio is a parameter described relationship between the downmixed signal energy and the residual signal energy of each of the M sub-bands. The res_dmx_ratio_lt is a parameter described a long-term smoothing parameter of the previous frame of the current frame. The long-term smoothing parameter res_dmx_ratio_lt is calculated according to

$$\text{res\_dmx\_ratio\_lt} = \text{res\_dmx\_ratio} * \alpha + \text{res\_dmx\_ratio\_lt\_prev} * (1 - \alpha)$$

wherein res_dmx_ratio_lt_prev represents the long-term smoothing parameter of the previous frame of the current frame, wherein

$$\alpha = \begin{cases} 0.2, & \text{frame\_nrg\_ratio} > 3.2 \text{ and } \text{res\_dmx\_ratio} < 0.1) \text{ or } (\text{frame\_nrg\_ratio} < 0.21 \text{ and } \text{res\_dmx\_ratio} > 0.4) \\ 0.05, & \text{else} \end{cases}$$

The res_dmx_ratio is calculated according to

$$\text{res\_dmx\_ratio[b]} = \text{res\_cod\_NRG\_S[b]}/(\text{res\_cod\_NRG\_S[b]} + (1 - g(b))\cdot(1 - g(b))\ \text{res\_cod\_NRG\_M[b]} + 1)$$

wherein res_dmx_ratio[b] represents the energy parameter of the sub-band whose sub-band index number is b, b is greater than or equal to 0 and is less than or equal to a preset maximum sub-band index number, res_cod_NRG_S[b] represents residual signal energy of the sub-band whose sub-band index number is b, res_cod_NRG_M[b] represents downmixed signal energy of the sub-band whose sub-band index number is b, and g(b) represents a function of a side gain side_gain[b] of the sub-band whose sub-band index number is b. The frame_nrg_ratio is a parameter described relationship between a sum of residual signal energy and downmixed signal energy of the M sub-bands, and a sum of residual signal energy and downmixed signal energy of M sub-bands in a frequency-domain signal of a previous frame of the current frame.

5.3.2.4.12.2.2 Adaptive downmix for stereo coding

In DFT stereo with WB 32 kbps, a corrected downmixed signal is calculated in a preset frequency band of the current frame when a previous frame of a current frame of a stereo signal is not a switching frame and a residual signal in the previous frame does not need to be encoded, or when a current frame is not a switching frame and a residual signal in the current frame does not need to be encoded. The corrected downmix signal is determined by a sum of the downmixed signal and the compensated downmixed signal in the current frame. The compensated downmixed signal in the sub-band b in the subframe i of the current frame is calculated according to:

$$\text{DMX\_comp}_{ib}(k) = \alpha_i(b) * L_{ib}''(k)$$

wherein $\text{DMX\_comp}_{ib}(k)$ represents the compensated downmixed signal in the sub-band b in the subframe i of the current frame, k represents a frequency bin index value, and $k \in [\text{band\_limits}(b), \text{band\_limits}(b+1) - 1]$. The downmix compensation factor $\alpha_i(b)$ in a sub-band b in the subframe i of the current frame is calculated according to

$$\alpha_i(b) = \frac{\sqrt{E\_L_i(b)} + \sqrt{E\_R_i(b)} - \sqrt{E\_LR_i(b)}}{2\sqrt{E\_L_i(b)}}$$

$$E\_L_i(b) = \sum_{k=band\_limits(b)}^{k=band\_limits(b+1)-1} L_{ib}''(k)^2$$

$$E\_R_i(b) = \sum_{k=band\_limits(b)}^{k=band\_limits(b+1)-1} R_{ib}''(k)^2$$

$$E\_LR_i(b) = \sum_{k=band\_limits(b)}^{k=band\_limits(b+1)-1} \left[L_{ib}''(k) + R_{ib}''(k)\right]^2$$

wherein $E\_L_i(b)$ represents an energy sum of a left channel frequency-domain signal in the sub-band b in the subframe i of the current frame; $E\_R_i(b)$ represents an energy sum of a right channel frequency-domain signal in the sub-band b in the subframe i of the current frame; $E\_LR_i(b)$ represents an energy sum of the energy of the left channel frequency-domain signal and the energy of the right channel frequency-domain signal in the sub-band b in the subframe i of the current frame; $band\_limits(b)$ represents a minimum frequency bin index value of the sub-band b in the subframe i of the current frame; $band\_limits(b+1)$ represents a minimum frequency bin index value of a sub-band b + 1 in the subframe i of the current frame; $L_{ib}''(k)$ represents a left channel frequency-domain signal that is in the sub-band b in the subframe i of the current frame and that is obtained after adjustment based on a stereo parameter; $R_{ib}''(k)$ represents a right channel frequency-domain signal that is in the subband b in the subframe i of the current frame and that is obtained after adjustment based on the stereo parameter; and k represents a frequency bin index value, wherein each subframe of the current frame comprises M sub-bands, the downmix compensation factor of the subframe i of the current frame comprises the downmix compensation factor of the subband b in the subframe i of the current frame, b is an integer, $b \in [0, M-1]$, and $M \geq 2$.

5.3.2.4.12.2.3 Calculation of downmixed signal and residual signal during transitional frame

In stereo coding, obtaining an initial downmixed signal and an initial residual signal of a sub-band corresponding to a preset frequency band in a current frame. If the previous frame is a switching frame, the downmixed signal and the residual signal of the sub-band corresponding to the preset frequency band in the current frame are calculated based on a switch fade-in/fade-out factor of a current frame, the initial downmixed signal, and the initial residual signal. The switch fade-in/fade-out factor of the current frame is determined based on a residual signal coding parameter and an inter-frame energy fluctuation parameter. The residual signal coding parameter is used to represent an energy relationship between a downmixed signal and a residual signal of the current frame, and the inter-frame energy fluctuation parameter is used to represent an energy or amplitude relationship between the current frame and a frame previous to the current frame. The switch fade-in/fade-out factor of the current frame is determined according to

when $frame\_nrg\_ratio > 3.2$ and $res\_dmx\_ratio < 0.1$, $switch\_fade\_factor = 0.75$; when $frame\_nrg\_ratio < 0.21$ and $res\_dmx\_ratio > 0.4$, $switch\_fade\_factor = 0.25$; in another case, $switch\_fade\_factor = 0.5$; wherein

$frame\_nrg\_ratio$ represents the inter-frame energy fluctuation parameter of the current frame which is defined as a ratio of total energy of the downmixed signal and the residual signal to total energy of a downmixed signal of a previous frame and a residual signal of the previous frame; res_dmx_ratio represents the residual signal coding parameter of the current frame; $switch\_fade\_factor$ represents the switch fade-in/fade-out factor of the current frame.

The initial downmixed signal and the initial residual signal are calculated according to

$$\overline{DMX_{i,b}}(k) = DMX_{i,b}(k) + (1 - switch\_fade\_factor) * DMX\_comp_{i,b}(k)$$

$$\overline{RES_{i,b}}(k) = switch\_fade\_factor * RES'_{i,b}(k)$$

wherein $\overline{DMX_{i,b}}(k)$ represents downmixed signal of a sub-band b in a subframe i in the current frame; $DMX_{i,b}(k)$ represents an initial downmixed signal of the sub-band b in the subframe i in the current frame; $DMX\_comp_{i,b}(k)$ represents a compensated downmixed signal of the sub-band b in the subframe i in the current frame; $RES'_{i,b}(k)$ represents an initial residual signal of the sub-band b in the subframe i in the current frame; $\overline{RES_{i,b}}(k)$ represents residual signal of the sub-band b in the subframe i in the current frame; the sub-band b in the subframe i in the current frame is a sub-band in the at least one sub-band corresponding to the preset frequency band; k represents a frequency bin index of the sub-band b in the subframe i in the current frame; and $0 \leq i \leq P-1$, wherein P represents a quantity of subframes comprised in the current frame.

when residual coding flag is unequal to residual coding flag value of previous frame, and a modification flag of the residual coding flag of the previous frame is 0 which indicates that the residual coding flag value of the previous frame

has not been modified, the residual coding switching flag set to 1 which indicates the frame is a switching frame and the residual signal should be encoded.

5.3.2.4.12.2.4 Adaptive downmix

The encoding mode indication information of the residual signal is obtained by at least one of the following information: residual signal encoding status of previous frame, updating manner flag for a long-term smooth parameter, status change parameter relative to previous frame. The encoding status of the previous frame is used to indicate at least one of the following cases: the quantity of consecutive frames whose residual signals are encoded before the current frame, a quantity of consecutive frames whose residual signals are not encoded before the current frame, and encoding modes of residual signals of previous frame. The status change parameter is a ratio of energy of the current frame to energy of previous frame. The encoding mode that used to indicate whether to encode the residual signal of the current frame is determined based on the encoding mode indication information and the initial encoding mode of the residual signal. The initial encoding mode of the residual signal of the current frame is determined based on the downmixed signal energy and the residual signal energy.

If the following conditions are met, the encoding mode of the current frame is the encoding mode of the previous frame. The conditions include that the initial encoding mode is different from the encoding mode of the previous frame which is closely adjacent to the current frame, and the encoding mode of the previous frame indicates to encode the residual signal of the previous frame, and an additional condition. The additional condition is the quantity of consecutive frames whose residual signals are encoded before the current frame is less than a threshold or the updating manner flag for the long-term smooth parameter is 0, and the encoding mode of the residual signal of the previous frame is not modified. If the additional condition is not met, the encoding mode of the current frame is the initial encoding mode. If the initial encoding mode is the same as the encoding mode of previous frame, the encoding mode of the residual signal of the current frame is the initial encoding mode.

If the following conditions are met, the encoding mode of the current frame is the encoding mode of the residual signal of the previous frame. The conditions include that the initial encoding mode is different from the encoding mode of previous frame, and the encoding mode of previous frame indicates not to encode the residual signal of the previous frame, and a second additional condition. The second additional condition is the quantity of consecutive frames whose residual signals are not encoded before the current frame is less than a threshold or the value of the status change parameter is not less than a second threshold, and not greater than a third threshold. If the second condition is not met, the encoding mode of the current frame is the initial encoding mode.

If the encoding mode of the residual signal of the current frame is different from the encoding mode of previous frame, and the encoding mode of previous frame is not modified, the encoding mode of the current frame is used to indicate the encoding mode of the current frame.

5.3.2.4.13 Reverberation gain parameter determination

The left channel signal and the right channel signal are treated as the first channel signal and the second channel signal. Encoder quantizes the first channel signal and the second channel signal based on the downmixed signal, the initial reverberation gain parameter, and the identification information, and writes the quantized first channel signal and a quantized second channel signal into the bitstream.

The reverberation gain parameters correspond to different sub-bands of the first channel signal and the second channel signal. The target reverberation gain parameter indicates those reverberation gain parameters that needs to be encoded. The target reverberation gain parameter is determined based on at least one of coherence between energy of the first channel signal and energy of the downmixed signal and coherence between energy of the second channel signal and the energy of the downmixed signal, wherein each of the first channel signal and the second channel signal comprises a plurality of frequency bins. The identification information is used to indicate a sub-band corresponding to the target reverberation gain parameter and whether the initial reverberation gain parameter needs to be adjusted. The identification information is determined based on the target difference value which is the larger difference value in the first difference value and the second difference value. The identification information uses 1bit to indicate the first frequency band. A target attenuation factor used to adjust initial reverberation gain parameter of a target channel signal is calculated based on the first difference value and the second difference value. Each of the plurality of attenuation factors corresponds to at least one sub-band of the target channel signal, and any sub-band corresponds to only one attenuation factor. The first difference value is a sum of absolute values of difference values between energy of the first channel signal and energy of the downmixed signal at a plurality of frequency bins, and the second difference value is a sum of absolute values of difference values between energy of the second channel signal and energy of the downmixed signal at the plurality of frequency bins. When the first difference value or the second difference value is greater than 120, the reverberation gain parameter corresponding to a sub-band of a first frequency band is the target reverberation gain parameter, wherein the

first frequency band is a part of all frequency bands of each of the first channel signal and the second channel signal, wherein a frequency of the first frequency band is less than a frequency of another frequency band different from the first frequency band in the first channel signal and the second channel signal. The plurality of frequency bins are in a second frequency band of each of the first channel signal and the second channel signal, and a frequency of the second frequency band is greater than a frequency of another frequency band, different from the second frequency band, in the first channel signal and the second channel signal.

5.3.2.5 Stereo classifier

5.3.2.5.1 General

The stereo classifier selects between the TD stereo mode and the DFT stereo mode. The selection is done in each frame based on stereo cues and stereo parameters. As a general rule, TD stereo mode is mostly selected for the encoding of stereo signals with uncorrelated left and right channels, especially signals with overlapping speech (cross-talk). The DFT stereo mode is mostly selected for stereo signals with correlated left and right channels with single-talk speech, music and mixed content.

The stereo classifier is composed of the following modules: 1) classifier of uncorrelated content, 2) cross-talk detector, and 3) stereo mode selector.

5.3.2.5.2 Classification of uncorrelated content

The classification of uncorrelated content is done in both, the TD stereo mode and the DFT stereo mode. The classification is based on the Logistic Regression (LogReg) model. The LogReg model is trained individually for the TD stereo mode and for the DFT stereo mode on a large database of stereo cues extracted from the IVAS coder.

The classification of uncorrelated content in the TD stereo mode uses the stereo cues:

- position of the maximum inter-channel cross-correlation function, k_{max}
- instantaneous target gain, g_t
- unnormalized cross-channel correlation at lag 0, p_{LR}
- side-to-mono energy ratio, r_{SM}
- difference between the maximum and the minimum cross-channel correlation, d_{mmLR}
- absolute difference between the Left-to-Mono and the Right-to-Mono correlation, d_{LRM}
- zero-lag cross-channel correlation value, R_0
- position of the maximum windowed inter-channel cross-correlation function, k_{wmax}
- maximum of the windowed inter-channel cross-correlation function, R_{wmax}
- evolution of the inter-channel cross-correlation function, RR

In total, there are $F_{TD}^{UNCLR} = 10$ stereo cues used by the classifier of uncorrelated content in the TD stereo mode forming a feature vector $\mathbf{f}_{raw,TD}^{UNCLR}$. For the simplicity of notation the subscript "TD" and the superscript "UNCLR" will be ommitted from the variables unless specifically stated. Thus, the input feature vector will be denoted as $\mathbf{f}_{raw}$.

The feature vector is normalized by removing its mean and scaling it to unit variance. This is done as follows

$$f_i = \frac{f_{i,raw} - \overline{f_i}}{\sigma_{f_i}}, \quad i = 1, \ldots, F \tag{5.3-183}$$

where $\overline{f_i}$ is the global mean of the *i*th feature across the training database and σ_{f_i} is the global variance of the *i*th feature across the training database. The global means $\overline{f_i}$ and the global variances σ_{f_i} for all $i = 1, \ldots, F$ are defined as constants in the IVAS coder.

The LogReg model takes the real-valued features as an input vector and makes a prediction as to the probability of the input belonging to the uncorrelated class (class 0). The output of the LogReg model is real-valued and takes the form of linear regression. That is

$$y_p = b_0 + \sum_{i=1}^{F} b_i f_i \quad (5.3\text{-}184)$$

where b_i are the constant coefficients of the LogReg model. The real-valued output y_p is then transformed into a probability using the logistic function. This is done as

$$\mathrm{p}(\mathrm{class} = 0) = \frac{1}{1+e^{-y_p}} \quad (5.3\text{-}185)$$

The probability $\mathrm{p}(\mathrm{class} = 0)$ is a real value between 0 and 1 where higher value means that the content in the current frame is more uncorrelated. The output of the LogReg model y_p is normalized with the function shown in Figure 5.3-29.

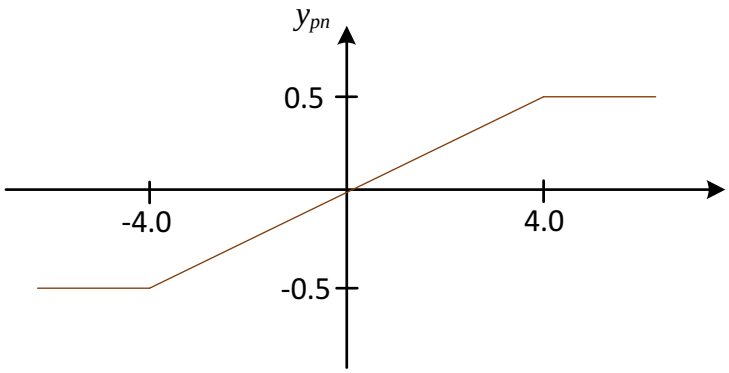


Figure 5.3-29: Normalization of the LogReg output of the classifier of uncorrelated content in the TD stereo mode

The normalization can be mathematically described as follows

$$y_{pn}(n) = \begin{cases} 0.5 & \text{if } y_p(n) \geq 4.0 \\ 0.125 y_p(n) & \text{if } -4.0 < y_p(n) < 4.0 \\ -0.5 & \text{if } y_p(n) \leq -4.0 \end{cases} \quad (5.3\text{-}186)$$

The normalized output of the LogReg model $y_{pn}(n)$ is then weighted with the relative frame energy as follows

$$scr_{UNCLR}(n) = y_{pn}(n) \cdot E_r(n) \quad (5.3\text{-}187)$$

where $E_r(n)$ is the relative frame energy defined in eq. (29) in clause 5.1.5.2 of [3]. The normalized weighted output of the LogReg model $scr_{UNCLR}(n)$ is called the "non-correlation score".

The non-correlation score $scr_{UNCLR}(n)$ contains occasional short-term "peaks" resulting from imperfect statistical model. The peaks can be filtered out by a simple averaging filter such as the first-order IIR filter. Unfortunately, the application of such averaging filter usually results in smearing of the rising edges representing transitions between the correlated and uncorrelated content. To preserve the rising edges it is necessary to reduce or even stop the smoothing process when a rising edge is detected in the input signal. The detection of rising edges in the input signal is done by analyzing the evolution of the relative frame energy.

The rising edges of the relative frame energy are found by filtering the relative frame energy with a cascade of $P = 20$ identical first-order RC filters each of which has the following form

$$F_p(z) = \frac{b_1 z^{-1}}{a_0 + a_1 z^{-1}}, \quad p = 1, \ldots, 20 \quad (5.3\text{-}188)$$

where a_0, a_1 and b_1 are the constants of the RC filter. The filtering of the relative frame energy with the cascade of $P = 20$ RC filters is done as follows

$$
\begin{aligned}
E_f^{[0]}(n) &= t_{edge} \cdot E_f^{[0]}(n-1) + \left(1 - t_{edge}\right) \cdot E_{rl}(n) \\
E_f^{[1]}(n) &= t_{edge} \cdot E_f^{[0]}(n-1) + \left(1 - t_{edge}\right) \cdot E_f^{[0]}(n) \\
&\dots \\
E_f^{[p]}(n) &= t_{edge} \cdot E_f^{[p]}(n-1) + \left(1 - t_{edge}\right) \cdot E_f^{[p-1]}(n)
\end{aligned}
\tag{5.3-189}
$$

where

$$
\frac{-a_1}{a_0} = \tau_{edge}, \quad \frac{b_1}{a_0} = 1 - \tau_{edge} \tag{5.3-190}
$$

The superscript $[p]$ has been added to denote the stage in the RC filter cascade. The output of the cascade of RC filters is equal to the output from the last stage, i.e.

$$
E_f(n) = E_f^{[P-1]}(n) = E_f^{[19]}(n) \tag{5.3-191}
$$

The cascade of first-order RC filters acts as a low-pass filter with a relatively sharp step function. When used on the relative frame energy it tends to smear out occasional short-term spikes while preserving slower but important transitions such as onsets and offsets. The slope of the rising edges is calculated as the difference between the relative frame energy and the filtered output. That is

$$
f_{edge}(n) = 0.95 - 0.05\big(E_{rl}(n) - E_f(n)\big) \tag{5.3-192}
$$

where $f_{edge}(n)$ is limited to the interval $\langle 0.95, \dots, 0.95 \rangle$. The score of the LogReg model scr is then smoothed with an IIR filter where $f_{edge}(n)$ is used as the forgetting factor. The smoothing is done as follows

$$
wscr_{UNCLR}(n) = f_{edge}(n) \cdot wscr_{UNCLR}(n-1) + \left(1 - f_{edge}(n)\right) \cdot scr_{UNCLR}(n) \tag{5.3-193}
$$

The classification of uncorrelated content in the DFT stereo mode is done similarly as the classification of uncorrelated content in the TD stereo mode described in equations (5.3-183) to (5.3-193). However, there are some notable differences.

In the DFT stereo mode the feature vector $\mathbf{f}_{raw}$ consists of the following stereo cues:

- ILD gain, g_{ILD}
- IPD gain, g_{IPD}
- IPD rotation angle, φ_{rot}
- prediction gain, g_{pred}
- mean energy of the inter-channel coherence, E_{coh}
- ratio of maximum and minimum intra-channel amplitude products, r_{PP}
- overall cross-channel spectral magnitude, f_X
- the maximum value of the GCC-PHAT function, G_{ITD}

In total, there are $F = 8$ features used by the classifier of uncorrelated content in the DFT stereo encoder. The normalization of the feature vector is done as per equation (5.3-183) where the global means $\overline{f_i}$ and the global variances σ_{f_i} for all $i = 1, \dots, F$ are defined specifically for the DFT stereo mode.

The LogReg model used in the DFT stereo mode is similar to the LogReg model in the TD stereo mode. The output of the LogReg model y_p is calculated with equation (5.3-184) and the probability that the current frame is uncorrelated (class 0) is given by equation (5.3-185). The raw output of the LogReg mode y_p is normalized similarly as in the TD stereo mode and according to the function in Figure 5.3-29. The normalization is mathematically described as follows

$$
y_{pn}(n) = \begin{cases} 0.5 & \text{if } y_p(n) \geq 4.0 \\ 0.125 y_p(n) & \text{if } -4.0 < y_p(n) < 4.0 \\ -0.5 & \text{if } y_p(n) \leq -4.0 \end{cases} \tag{5.3-194}
$$

The normalized output of the LogReg model $y_{pn}(n)$ is weighted with the relative frame energy as follows

$$scr_{UNCLR}(n) = y_{pn}(n) \cdot E_r(n) \quad (5.3\text{-}195)$$

and the output value is called the “non-correlation score”. The non-correlation score is reset to 0 when the VAD flag $f_{VAD}(n)$ is reset to 0. That is

$$scr_{UNCLR}(n) = 0, \text{ if } f_{VAD}(n) = 0 \quad (5.3\text{-}196)$$

Finally, the non-correlation score $scr_{UNCLR}(n)$ is smoothed with the IIR filter using the rising-edge detection mechanism described in equation (5.3-193). The output of the smoothing filter is denoted $wscr_{UNCLR}(n)$.

The classifier of uncorrelated content takes the smoothed non-correlation score $wscr_{UNCLR}(n)$ and makes a binary decision. Let $c_{UNCLR}(n)$ denote the binary output of the classifier with “1” representing the uncorrelated content and “0” representing the correlated content. The final output of the classifier is a state variable. It is initialized to 0. The classifier state is changed from the current class to the complementary class only in frames where certain conditions are met.

The mechanism for switching between the output states is depicted in Figure 5.3-30 in the form of a simple state machine.

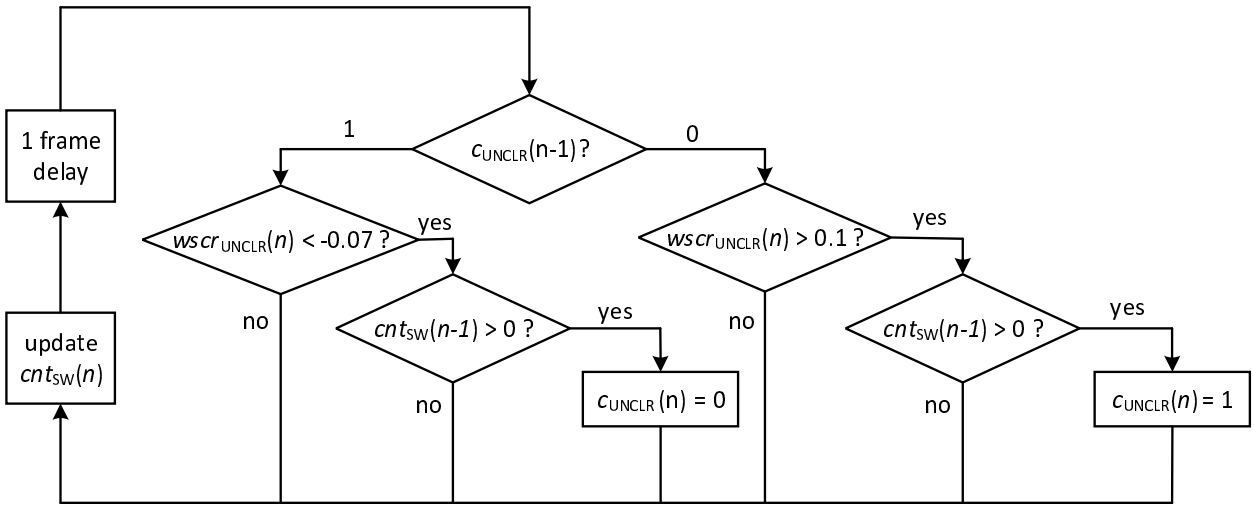


Figure 5.3-30: State-switching logic in the classifier of uncorrelated content

The counter $cnt_{sw}(n)$ indicates frames where it is possible to switch between the TD stereo mode and the DFT stereo mode. This counter is initialized to zero and updated in each frame with the following logic

$$cnt_{sw}(n) = \begin{cases} cnt_{sw}(n) + 1, & \text{if } c_{type} \in (\text{GENERIC, UNVOICED, INACTIVE}) \\ cnt_{sw}(n) + 1, & \text{if } c_{type} \notin (\text{GENERIC, UNVOICED, INACTIVE}) \wedge \text{VAD} = 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.3\text{-}197)$$

where c_{type} is the frame type of the current frame in the encoder defined in clause 5.1.13 of [3] and the VAD flag is defined in clause 5.1.12 of [3]. The counter $cnt_{sw}(n)$ is limited to the upper limit of 100.

5.3.2.5.3 Cross-talk detection using logistic regression

The cross-talk detector is based on the logistic regression statistical model trained individually for the TD stereo mode and for the DFT stereo mode. Both statistical models are trained on a large database of stereo cues collected by running the IVAS coder on real recordings as well as artificially-prepared stereo samples.

Cross-talk detection in the TD stereo mode is done similarly as the classification of uncorrelated content in the TD stereo mode described in clause 5.3.2.5.2. Due to the similarity between the classifiers the mathematical symbols from clause 5.3.2.5.2 will be re-used in the following text without specifically distinguishing them with the "XTALK" superscript.

The following features are used by the cross-talk detector in the TD stereo mode:

- L/R class difference, d_{clas}
- L/R difference of the maximum autocorrelation, d_v
- L/R difference of the average LSF, d_{LSF}
- L/R difference of the residual LP energy, d_{LPC13}
- L/R difference of the spectral correlation map, d_{cmap}
- L/R difference of noise characteristics, d_{nchar}
- L/R difference of the non-stationarity, d_{sta}
- L/R difference of the spectral diversity, d_{sdiv}
- un-normalized cross-channel correlation at lag 0, p_{LR}
- side-to-mono energy ratio, r_{SM}
- difference between the maximum and the minimum cross-channel correlation, d_{mmLR}
- zero-lag cross-channel correlation value, R_0
- evolution of the inter-channel cross-correlation function, RR
- position of the maximum inter-channel cross-correlation function, k_{max}
- maximum of the inter-channel correlation function, R_{max}
- difference between L/M and R/M correlation functions, Δ_{LRM}
- smoothed ratio of energies of the side signal and the mono signal, $\overline{r_{SM}}(n)$

In total, there are $F_{TD} = 17$ features used by the cross-talk detector in the TD stereo mode forming a feature vector $\mathbf{f}_{raw,TD}$. For the simplicity of notation the subscript "TD" will be dropped from the symbol names unless specifically stated.

Before the training process, the entire feature set is normalized by removing its global mean and scaling it to unit variance. This is done as follows

$$f_i = \frac{f_{i,raw} - \overline{f_i}}{\sigma_{f_i}}, \quad i = 1, \ldots, F \tag{5.3-198}$$

where $\overline{f_i}$ is the global mean of the *i*th feature across the training database and σ_{f_i} is the global variance of the *i*th feature across the training database. Please, note that the parameters $\overline{f_i}$ and σ_{f_i} used in the equation above have the same meaning but different value than in equation (5.3-183).

The output of the LogReg model is calculated with equation (5.3-184) and the probability that the current frame belongs to the cross-talk segment class (class 0) is given by equation (5.3-185). The output of the LogReg model y_p is then normalized with the function shown in Figure 5.3-31.

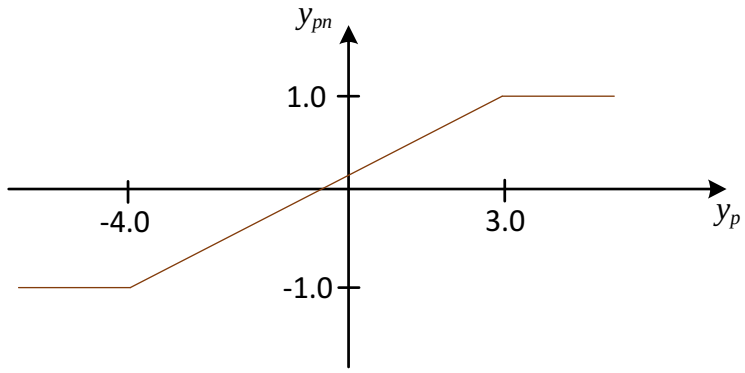


Figure 5.3-31: Normalization of the LogReg output of the cross-talk classifier in the TD stereo mode

The normalization can be mathematically described as follows

$$y_{pn}(n) = \begin{cases} 1.0 & \text{if } y_p(n) \geq 3.0 \\ 0.333 y_p(n) & \text{if } -3.0 < y_p(n) < 3.0 \\ -1.0 & \text{if } y_p(n) \leq -3.0 \end{cases} \quad (5.3\text{-}199)$$

The normalized output of the LogReg model $y_{pn}(n)$ is reset to 0 if the previous frame was encoded with the DFT stereo mode and the current frame is encoded with the TD stereo mode. This is done to prevent stereo mode switching artifacts.

The normalized output of the LogReg model $y_{pn}(n)$ is then weighted based on the relative frame energy $E_r(n)$ defined in eq. (29) in clause 5.1.5.2 of [3]. The weighting scheme applied in the cross-talk classifier in TD stereo mode is similar to the weighting scheme applied in the classifier of uncorrelated content in the TD stereo mode described in clause 5.3.2.5.2. The main difference is that the relative frame energy $E_r(n)$ is not used as the multiplicative factor (5.3-195). Instead, the relative frame energy $E_r(n)$ is linearly mapped in the interval $\langle 0, \ldots, 0.95 \rangle$ with inverse proportion. That is

$$w_{relE}(n) = -2.375 E_r(n) + 2.1375 \quad (5.3\text{-}200)$$

Thus, in frames with higher relative energy the weight is closer to 0 whereas in frames with low energy the weight is closer to the maximum value of 0.95. The weight $w_{relE}(n)$ is then used to filter the normalized output of the LogReg model $y_{pn}(n)$ as follows

$$scr_{XTALK}(n) = w_{relE} scr_{XTALK}(n-1) + (1 - w_{relE}) y_{pn}(n) \quad (5.3\text{-}201)$$

The normalized weighted output of the cross-talk detector is called the “cross-talk score”.

Similarly as in the classification of uncorrelated content in the TD stereo mode it is necessary to filter out occasional short-term “peaks” and “dips” from $scr_{XTALK}(n)$ as they could lead to clasification errors or false alarms. The filtering is done in such a way that rising edges of the LogReg output are preserved. The rising edges represent important transitions between the cross-talk segments and single-talk segments. The mechanism for rising-edge detection in the cross-talk classifier in the TD stereo mode is different to the mechanism of rising-edge detection described in clause 5.3.2.5.2.

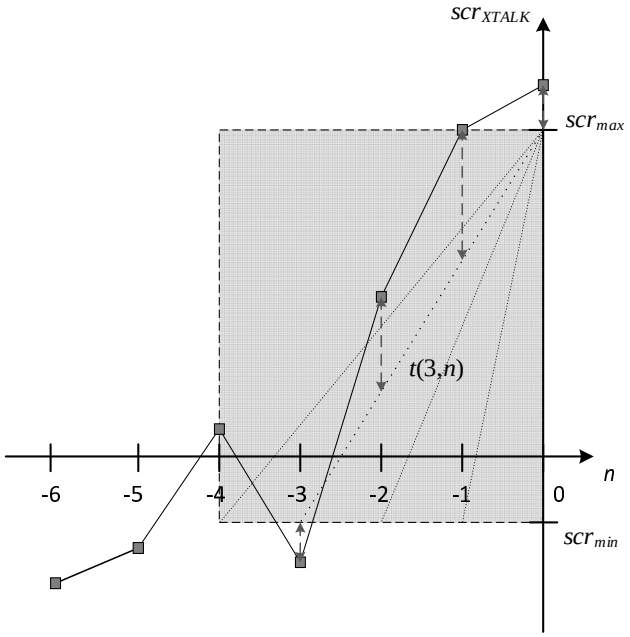


Figure 5.3-32: Rising-edge evaluation in the cross-talk detector in the TD stereo mode

The rising-edge detection algorithm analyzes the LogReg values from previous frames and compares them against a set of pre-calculated "ideal" edges with various slopes. The "ideal" edges are represented as linear functions of the frame index (n). The rising-edge detection mechanism is described by the schematic diagram in Figure 5.3-32. The *x* axis has the frame indices of frames preceding the current frame 0. The small grey rectangles represent exemplary values of the cross-talk score $scr_{XTALK}(n)$ in the last six frames preceding the current frame. It is clear from the graph that there is a rising edge starting at frame (-3). The dotted lines illustrate a set of four "ideal" rising edges that $scr_{XTALK}(n)$ is compared to.

For each "ideal" rising edge the algorithm calculates the mean-square error between the dotted line and the cross-talk score $scr_{XTALK}(n)$. The output of the rising-edge detector is the minimum value among the mean-square errors. The coefficients of the linear functions represented by the dotted lines are pre-calculated based on the minimum value scr_{min} and the maximum value scr_{max}.

The rising-edge detection is performed only in frames meeting the following criterion:

$$\begin{gathered}\min_{k=0,..,K}(scr_{XTALK}(n-k)) < 0 \text{ AND} \\ \min_{k=0,..,K}(scr_{XTALK}(n-k)) < 0 \text{ AND} \\ \max_{k=0,..,\mathrm{K}}(scr_{XTALK}(n-k)) - \min_{k=0,..,K}(scr_{XTALK}(n-k)) > 0.2\end{gathered} \tag{5.3-202}$$

where $K = 4$ is the maximum length of the tested edges.

Let the output value of the rising-edge detector be denoted ε_{0_1}. The "0_1" subscript underlines the fact that the output value of the rising-edge detector is limited in the interval $\langle 0, \dots, 1\rangle$. For frames not meeting the criterion (5.3-202) the output value of the rising-edge detector is reset to 0, i.e. $\varepsilon_{0\_1} = 0$.

The set of linear functions representing the tested rising edges can be mathematically expressed with the following formula

$$t(l, n-k) = scr_{\max} - k\frac{scr_{\max} - scr_{\min}}{l}, \quad l = 1, \dots, K; k = 1, \dots, l \tag{5.3-203}$$

where the index l denotes the length of the tested edge and $n-k$ is the frame index. The slope of each linear function is determined by three parameters, the length of the tested edge l, the minimum threshold $scr_{\min}$ and the maximum threshold $scr_{\max}$. For the purpose of the cross-talk detector in the TD stereo mode the thresholds are set as follows

$$\begin{gathered}scr_{max} = 1.0 \\ scr_{\min} = -0.2\end{gathered} \tag{5.3-204}$$

The values of the minimum and the maximum threshold were found experimentally. For each length of the tested edge the algorithm calculates the mean-square error between the linear function t and the true cross-talk score $\mathrm{scr}_{\mathrm{XTALK}}$. That is

$$\varepsilon(l) = \frac{1}{l}\varepsilon_0 + \frac{1}{l}\sum_{k=1}^{l}[scr_{XTALK}(n-k) - t(\mathrm{l}, n-k)]^2\,, \quad l = 1,..,K \tag{5.3-205}$$

where ε_0 is the initial error given by

$$\varepsilon_0 = [scr_{XTALK}(n) - scr_{max}[]^2] \tag{5.3-206}$$

The minimum mean-square error is calculated by

$$\varepsilon_{\min} = \min_{l=1,..,K}(\varepsilon(l)) \tag{5.3-207}$$

The lower the minimum mean-square error the stronger the detected rising edge. If the minimum mean-square error is higher than 0.3 then the output value of the rising-edge detector is reset to 0. In all other cases the minimum mean-square error is mapped linearly in the interval $\langle 0, \ldots, 1\rangle$. That is

$$\begin{aligned} \varepsilon_{0\_1} &= 0 && \text{if } \varepsilon_{min} > 0.3 \\ \varepsilon_{0\_1} &= 1 - 2.5\varepsilon_{\min} && \text{otherwise} \end{aligned} \tag{5.3-208}$$

Note, that the relationship between the output value of the rising-edge detector and the minimum mean-square error is inversely proportional.

The output value of the rising-edge detector is linearly mapped and limited to the interval of $\langle 0.5, \ldots, 0.9\rangle$. The resulting value is denoted as the edge sharpness $f_{edge}(n)$. The edge sharpness is calculated as follows

$$f_{edge}(n) = 0.9 - 0.4\varepsilon_{0\_1} \tag{5.3-209}$$

Finally, the normalized weighted output of the LogReg model $scr_{XTALK}(n)$ is smoothed with an IIR filter where $f_{edge}(n)$ is used as the smoothing factor. The smoothing with the IIR filter is done as follows

$$wscr_{XTALK}(n) = f_{edge}(n) \cdot wscr_{XTALK}(n-1) + \big(1 - f_{edge}(n)\big) \cdot scr_{XTALK}(n) \tag{5.3-210}$$

The smoothed output $wscr_{XTALK}(n)$ is reset to 0 in frames where the VAD flag $f_{VAD}(n)$ is zero. That is

$$wscr_{XTALK}(n) = 0, \text{ if } f_{VAD}(n) = 0 \tag{5.3-211}$$

Cross-talk detection in the DFT stereo mode is done similarly as cross-talk detection in the TD stereo mode which is described in equations (5.3-198) - (5.3-211). The Logistic Regression (LogReg) model is used as the basis of binary classification of the input feature vector. Please note, that to avoid the duplication of symbols the notation used in equations (5.3-198) - (5.3-211) will be reused in this section without specifically mentioning that it's related to the DFT stereo mode.

In the DFT stereo mode the feature vector $\mathbf{f}_{raw}$ of the cross-talk detector consists of the following stereo cues:

- ILD gain, g_{ILD}
- IPD gain, g_{IPD}
- IPD rotation angle, φ_{rot}
- prediction gain, g_{pred}
- mean energy of the inter-channel coherence, E_{coh}
- ratio of maximum and minimum intra-channel amplitude products, r_{PP}
- overall cross-channel spectral magnitude, f_X
- the maximum value of the GCC-PHAT function, G_{ITD}
- relationship between the amplitudes of the first and the second highest peak of the GCC-PHAT function, r_{GITD12}
- the amplitude of the second highest peak of the GCC-PHAT, m_{ITD2}

- the difference of the position of the second highest peak in the current frame with respect to the previous frame, Δ_{ITD2}

In total, there are $F = 11$ features used by the cross-talk detector in the DFT stereo mode.

The feature vector is normalized by removing its global mean and scaling it to unit variance. This is done as follows

$$f_i = \frac{f_{i,raw} - \bar{f}_i}{\sigma_{f_i}}, \quad i = 1, \ldots, F \tag{5.3-212}$$

where $\bar{f}_i$ is the global mean of the *i*th feature across the training database and σ_{f_i} is the global variance of the *i*th feature across the training database. Please, note that the parameters $\bar{f}_i$ and σ_{f_i} used in the equation above have the same meaning but different value than in equation (5.3-183).

The output of the LogReg model is described by equation (5.3-184) and the probability that the current frame is part of cross-talk segment (class 0) is given by equation (5.3-185). The raw output of the LogReg model y_p is normalized with the function shown in Figure 5.3-29. The normalized output of the LogReg model is denoted y_{pn}. In the DFT stereo mode, no weighting based on relative frame energy is performed. Therefore, the normalized weighted output of the LogReg model (the cross-talk score) $scr_{XTALK}(n)$ is simply set as follows

$$scr_{XTALK}(n) = y_{pn} \tag{5.3-213}$$

The cross-talk score $scr_{XTALK}(n)$ is reset to 0 when the VAD flag $f_{VAD}(n)$ is set to 0. That is

$$scr_{XTALK}(n) = 0, \text{ if } f_{VAD}(n) = 0 \tag{5.3-214}$$

Similarly as in the case of cross-talk detection the TD stereo mode it is necessary to smooth the cross-talk score $scr_{XTALK}(n)$ to remove short-term peaks. The smoothing is done by means of IIR filter using the rising-edge detection mechanism described in equations (5.3-202) - (5.3-209). The cross-talk score $scr_{XTALK}(n)$ is smoothed with the IIR filter as follows

$$wscr_{XTALK}(n) = f_{edge}(n) \cdot wscr_{XTALK}(n-1) + \left(1 - f_{edge}(n)\right) \cdot scr_{XTALK}(n) \tag{5.3-215}$$

where $f_{edge}(n)$ is the edge sharpness calculated in equation (5.3-209).

The classifier of uncorrelated content takes the smoothed score $wscr_{XTALK}(n)$ and makes a binary decision. Let $c_{UNCLR}(n)$ denote the binary output of the classifier with "1" representing the uncorrelated content and "0" representing the correlated content. The final output of the classifier is a state variable. It is initialized to 0. The classifier state is changed from the current class to the complementary class only in frames where certain conditions are met.

The cross-talk detector takes the smoothed cross-talk score $wscr_{XTALK}(n)$ and makes a binary decision. Let $c_{XTALK}(n)$ denote the binary output of the cross-talk detector with "1" representing the cross-talk class and "0" representing the single-talk class. The final output of the classifier is a state variable. It is initialized to 0. The classifier state is changed from the current class to the complementary class only in frames where certain conditions are met. The mechanism for cross-talk class switching is similar to the mechanism of class switching on uncorrelated content which has been described in detail in clause 5.3.2.5.2. However, there are some notable differences for both the TD stereo mode and the DFT stereo mode. These differences are discussed in more detail.

The state-switching mechanism in the cross-talk detector in the TD stereo mode is shown in Figure 5.3-33.

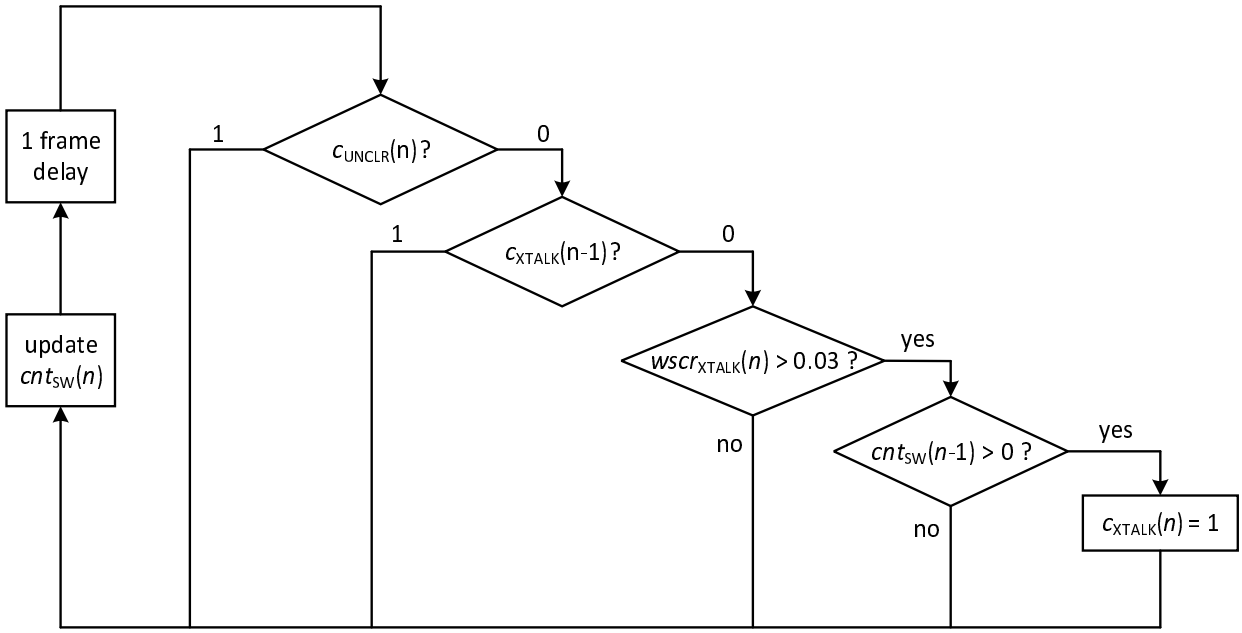


Figure 5.3-33: State-switching logic in the cross-talk detector in the TD stereo mode

The counter $cnt_{sw}(n)$ defined in equation (5.3-197) is shared between the classifier of uncorrelated content and the cross-talk detector. A positive value of the counter $cnt_{sw}(n)$ indicates that switching of the binary output of the cross-talk detector $c_{XTALK}(n)$ is allowed. Note, that the state switching logic in the cross-talk detector uses the binary output of the uncorrelated classifier $c_{UNCLR}(n)$. Note, that the state-switching logic in Figure 5.3-33 is unidirectional in the sense that the output of the cross-talk detector $c_{XTALK}(n)$ can only be changed from “0” (single-talk) to “1” (cross-talk). The state-switching logic in the opposite direction, i.e. from “1” (cross-talk) to “0” (single-talk), is part of the DFT/TD stereo mode selection logic which is described in detail in clause 5.3.2.5.5.

The state-switching mechanism of the cross-talk detector in the DFT stereo mode is shown in Figure 5.3-33.

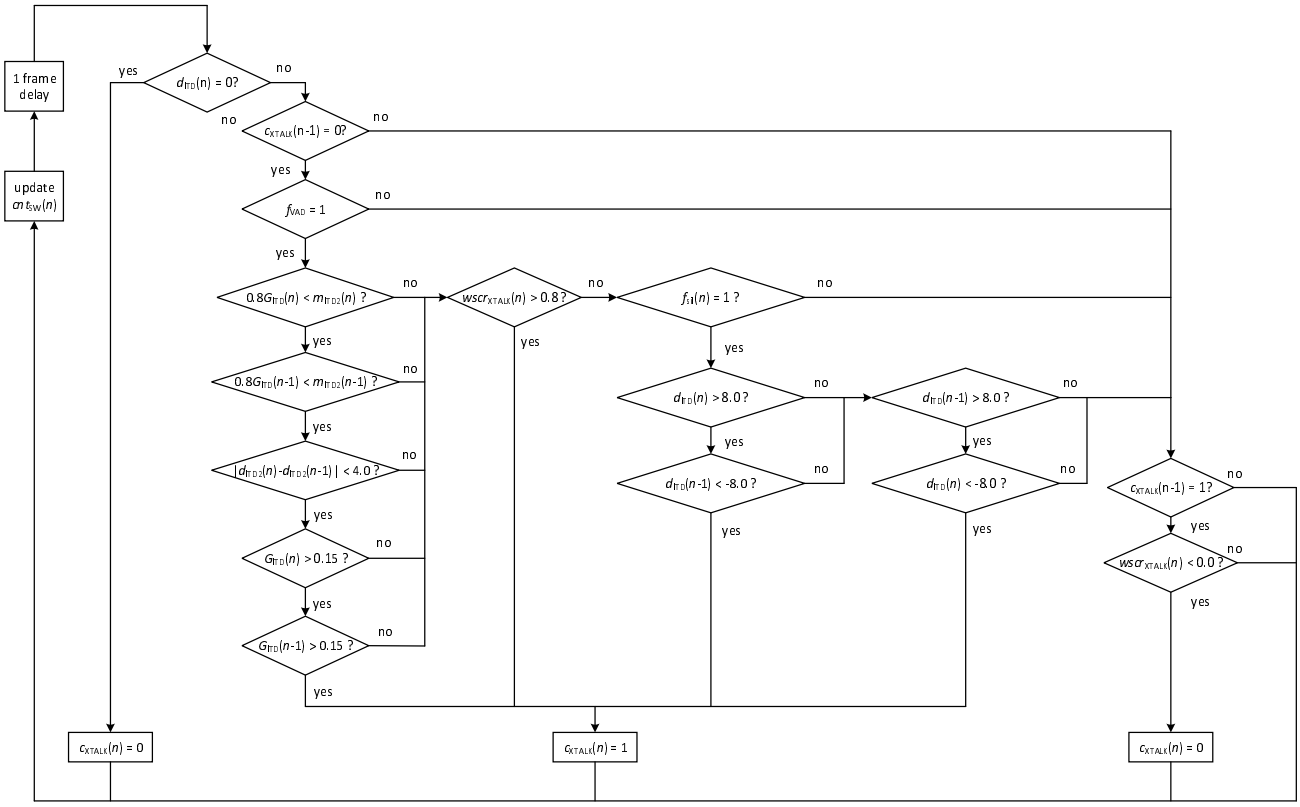


Figure 5.3-34: State-switching logic in the cross-talk detector in the DFT stereo mode

Similarly to the TD stereo mode, the counter $cnt_{sw}(n)$ is the counter of frames where it is possible to switch the binary output of the cross-talk detector. The counter $cnt_{sw}(n)$ is shared between the classifier of uncorrelated content and the cross-talk detector. The counter $cnt_{sw}(n)$ is initialized to zero and updated in each frame with equation (5.3-197).

5.3.2.5.4 Cross-talk detection based on the GCC-PHAT

For the transmission of stereo speech signals captured with two microphones with a certain distance between them when low bitrate is required, DFT stereo as a parametric stereo technique described in 5.3.2.4 is more suitable.

For the case where we have two or more talkers around this microphone setup and more than one talker is talking simultaneously during the same time period a parametric stereo system performs adequately for most situations. However, there are some special cases where the parametric model fails to reproduce the stereo image and deliver intelligible speech output for interfering talker scenarios. That happens when each of the two talkers are captured with a different ITD (Inter-channel Time Difference), the ITD values are large (large distance between the microphones) and the talkers are sitting in opposite positions around the microphone setup axis.

On the other hand, in a parametric stereo scheme like DFT stereo described in clause 5.3.2.4 some parameters are extracted to reproduce the spatial stereo scene and the stereo signal is deduced to a single-channel downmix that is further coded. In the case of interfering talkers, the downmix signal would be coded with ACELP as described in 5.2.2.3.2.1. However, such coding schemes are source-filter models of speech production, designed to represent single talker speech. For interfering talkers, it may be that the core coding model is being violated and perceptual quality is degraded.

For these particular cases discrete coding of the two stereo channels is preferred for best performance and therefore the stereo classifier switches to the TD stereo coding mode where discrete coding of the two channels is possible as described in 5.3.2.3.

In this clause the critical case is described when the two talkers have different ITDs and the difference between the two ITDs is large.

The stereo classifier utilizes the existing ITD estimator described in 5.3.2.4.4 which is based on the GCC-PHAT (Generalized Cross-Correlation Phase Transform). The basic principle of such an estimator is to detect a peak in the GCC-PHAT and this peak corresponds to the ITD of the stereo signal. However, when two talkers are speaking at the

same time and they have two different ITDs, there are typically two peaks in the GCC-PHAT as can be seen in an example in Figure 5.3-35. The logic is then to detect whether there is only one peak which would imply remaining in the parametric coding mode of the DFT stereo or two peaks far from each other in the GCC-PHAT which would imply switching to the TD-stereo mode.

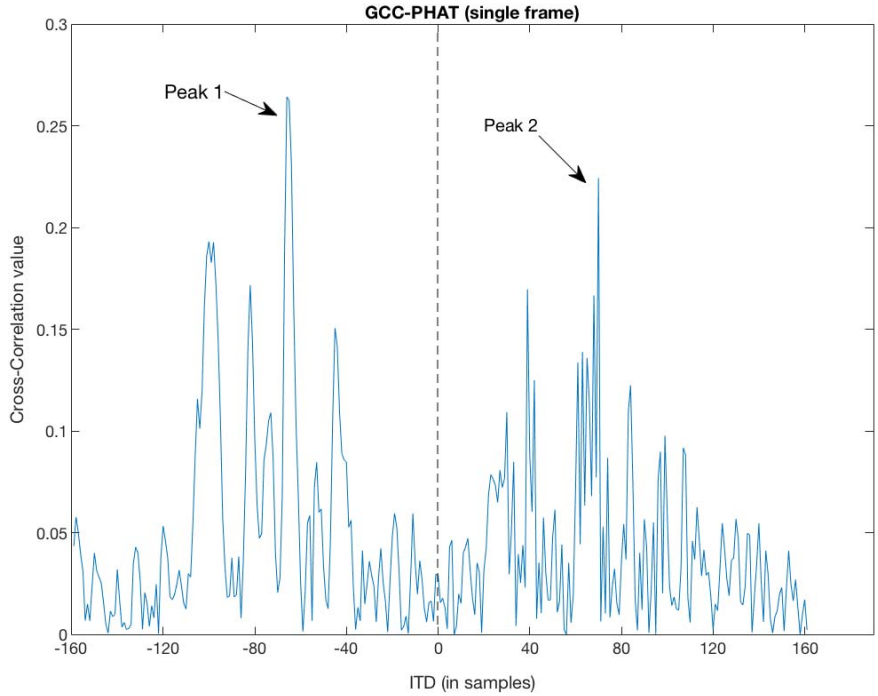


Figure 5.3-35: Example of the GCC-PHAT at a single frame with overlapping talkers

The algorithm is described in the following steps:

- Codec operates by default in DFT stereo mode
- Compute the GCC-PHAT of the stereo signal using a smoothed version of the cross-spectrum as described in 5.3.2.4.4 and equation (5.3-104).
- Estimate the main peak m_1 of the GCC-PHAT. This should correspond to the maximum of the absolute value of the GCC-PHAT. But it can also be different if some hysteresis is applied from the hangover mechanism described in clause 5.3.2.4.4.4to have a more stable ITD estimation.
- Select a portion of the GCC-PHAT which is far from the main peak: the distance between the main peak and the border of the portion is above a certain threshold. The threshold is bitrate-dependent due to the fact that starting from 32 kbps, DFT stereo operates with residual coding, which to some extent cans still accommodate coding of cross-talk segments. The threshold is defined as follows in Table 5.3-11.

Table 5.3-11: Threshold to enable switching per bitrate

Total bitrate	Threshold thr
32 kbps	16
24.4 kbps	8

- Find a second peak in the selected portion: this is for example the maximum of the absolute value of the GCC-PHAT. The steps to find the second peak are described in the following pseudocode:

```
m1 = 0.0f;
m2 = 0.0f;

itd2 = 0;

if ( itd > thr )
{
```

```
        m1 = fabsf( gcc_phat[itd + XTALK_PHAT_LEN] );
        m2 = fabsf( gcc_phat[0] );
        itd2 = -XTALK_PHAT_LEN;
        for ( i = 1; i < XTALK_PHAT_LEN - thr; i++ )
        {
            if ( fabsf( gcc_phat[i] ) > m2 )
            {
                itd2 = -XTALK_PHAT_LEN + i;
                m2 = fabsf( gcc_phat[i] );
            }
        }
    }
    else if ( itd < -thr )
    {
        m1 = fabsf( gcc_phat[itd + XTALK_PHAT_LEN] );
        m2 = fabsf( gcc_phat[XTALK_PHAT_LEN + thr + 1] );
        itd2 = thr + 1;
        for ( i = XTALK_PHAT_LEN + thr + 2; i < 2 * XTALK_PHAT_LEN + 1; i++ )
        {
            if ( fabsf( gcc_phat[i] ) > m2 )
            {
                itd2 = -XTALK_PHAT_LEN + i;
                m2 = fabsf( gcc_phat[i] );
            }
        }
    }
```

where XTALK_PHAT_LEN=200.

If the value of the second peak m_2 is higher of the threshold which is defined as the 80% of the value of the first peak m_1, i.e.:

$$m_2 > 0.8\, m_1 \tag{5.3-216}$$

then we can consider that the GCC-PHAT contains two significant peaks and we switch to mode TD stereo. Otherwise, there is no significant second peak, and we stay in DFT stereo.

Apart from the basic algorithm and the decision based on Equation (5.3-216) there are considerations to be made specific to the signal properties. The conditions that need to be additionally evaluated are summarized below:

Condition 1: check that $m_1 > 0.15$ to avoid switching on noisy frames, meaning the normalized cross-correlation is rather low and the detected peak may be rather random.

Condition 2: both conditions of Equation (5.3-216) and Condition 1 have to be verified on two consecutive frames. This is to avoid switching on unstable signals. This translates to:

$$m_2 > 0.8 m_1 \text{ and } m_2^{[-1]} > 0.8 m_1^{[-1]} \text{ and } m_1 > 0.15 \text{ and } m_1^{[-1]} > 0.15 \tag{5.3-217}$$

Condition 3: itd_2 of two consecutive frames have to be close to each other and their difference has to be below 4 samples. This is to avoid switching frequently on unstable signals, but to switch if the peak is toggling around a certain ITD value.

$$\left| itd_2 - itd_2^{[-1]} \right| < 4 \tag{5.3-218}$$

Condition 4: the SAD flag of the previous frame has to be 1 (meaning it is an active signal). This is to avoid switching at an onset of a scpeech segment.

Condition 5: itd_1 changes abruptly from one frame to the next by a big difference. In that case, we don't need to check for a second peak, we consider that a second speaker started talking and we switch to TD stereo:

$$itd_1 > thr \text{ and } itd_1^{[-1]} < -thr \tag{5.3-219}$$

or

$$itd_1 > thr \text{ and } itd_1 < -thr \tag{5.3-220}$$

where thr is the threshold defined in Table 5.3-11.

5.3.2.5.5 Stereo mode selection

The classifier of uncorrelated content and the cross-talk detector are essential parts of the DFT/TD stereo mode selection technology. The IVAS coder selects the appropriate stereo mode based on the binary outputs of both stereo classifiers. The stereo mode selection logic also takes into account some auxiliary parameters. These auxiliary parameters are used to prevent stereo mode switching in perceptually sensitive segments or to prevent frequent switching in segments where either the classifier of uncorrelated content or the cross-talk detector provide inaccurate outputs.

Stereo mode selection is done before down-mixing and encoding of the input stereo signal. Since both stereo classifiers are using parameters extracted from the IVAS encoder the stereo mode selector uses the outputs of the stereo classifiers from the previous frame. The stereo mode selection logic is described at the schematic diagram in Figure 5.3-36.

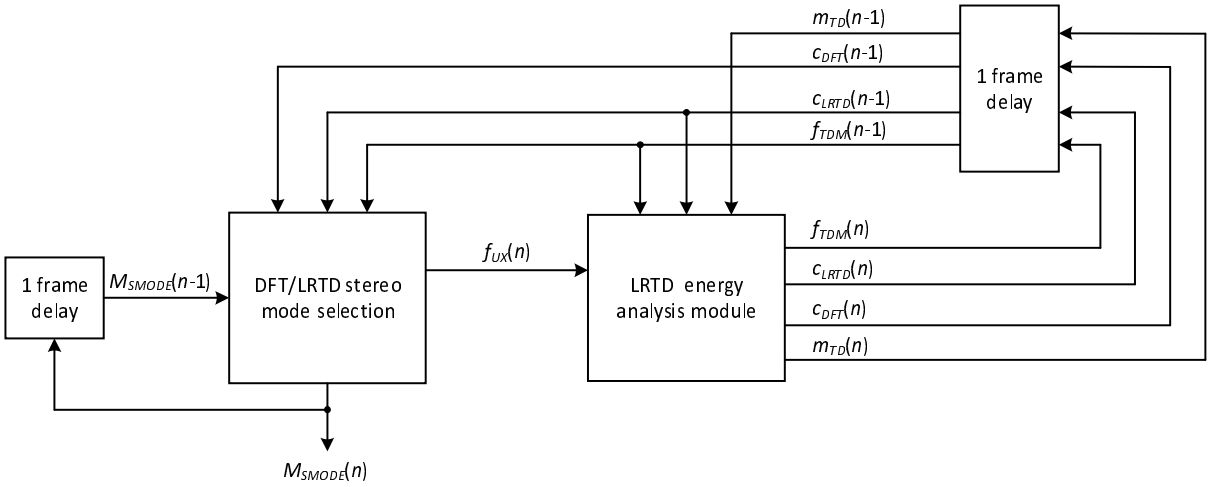


Figure 5.3-36: DFT/TD stereo mode selection

The DFT/TD stereo mode selection mechanism consists of the following functional blocks:

- Initial DFT/TD stereo mode selection
- TD to DFT stereo mode switching logic on cross-talk content

These functional blocks are described in detail in the following text.

The DFT stereo mode is the preferred mode for encoding single-talk speech with high inter-channel correlation between the left and the right channel of the input stereo signal. The initial selection of the stereo mode starts by checking whether the previous frame processed by the IVAS coder was “likely a speech frame”. This is done by examining the log-likelihood ratio between the “speech” class and the “music” class of the speech/music classifier, described in clause 5.2.2.2.11. The log-likelihood ratio is the difference between the log-likelihood of “music” and the log-likelihood of “speech”. That is

$$dL_{SM}(n) = L_M(n) - L_S(n) \tag{5.3-221}$$

where $L_S(n)$ is the log-likelihood of the “speech” class and $L_M(n)$ the log-likelihood of the “music” class. The details of log-likelihood ratio calculations are described in clause 5.1.13.6.3 of [3].

The log-likelihood ratio is smoothed with two IIR filters each of which has its own forgetting factor. The IIR filtering is done as follows

$$\begin{aligned} wdL_{SM}^{(1)}(n) &= 0.97 \cdot wdL_{SM}^{(1)}(n-1) + 0.03 \cdot dL_{SM}(n-1) \\ dL_{SM}^{(2)}(n) &= 0.995 \cdot wdL_{SM}^{(2)}(n-1) + 0.005 \cdot dL_{SM}(n-1) \end{aligned} \tag{5.3-222}$$

where the superscript (1) indicates the first IIR filter and the superscript (2) indicates the second IIR filter, respectively.

The smoothed outputs $wdL_{SM}^{(1)}(n)$ and $wdL_{SM}^{(2)}(n)$ are then compared with predefined thresholds and a binary flag $f_{SM}(n)$ is set to 1 if the following condition is met

$$f_{SM}(n) = \begin{cases} 1 & \text{if } wdL_{SM}^{(1)}(n) < 1.0 \text{ AND } wdL_{SM}^{(2)}(n) < 0.0 \\ 0 & \text{otherwise} \end{cases} \quad (5.3\text{-}223)$$

The binary flag $f_{SM}(n)$ is an indicator that the previous frame was likely a speech frame.

The initial stereo mode selection mechanism then sets another binary flag $f_{UX}(n)$ to 1 if the binary output of the uncorrelated classifier $c_{UNCLR}(n-1)$ or the binary output of the cross-talk detector $c_{XTALK}(n-1)$ are set to 1 and if the previous frame was likely a speech frame. That is

$$f_{UX}(n) = \begin{cases} 1 & \text{if } f_{SM}(n) = 1 \text{ AND } (c_{UNCLR}(n-1) = 1 \text{ OR } c_{XTALK}(n-1) = 1) \\ 0 & \text{otherwise} \end{cases} \quad (5.3\text{-}224)$$

Let $M_{SMODE}(n) \in (\text{TD}=3, \text{DFT}=2)$ be a discrete variable denoting the selected stereo mode in the current frame. The numeric constants "TD" and "DFT" are used to denote the TD stereo mode and the DFT stereo mode, respectively. The stereo mode is initialized in each frame with the value from the previous frame, i.e.

$$M_{SMODE}(n) = M_{SMODE}(n-1) \quad (5.3\text{-}225)$$

where $M_{SMODE}(0)$ is set to "TD". If the binary flag $f_{UX}(n)$ is set to 1, then the TD stereo mode is selected for the encoding of the input signal in the current frame. That is

$$M_{SMODE}(n) \leftarrow \text{TD if } f_{UX}(n) = 1 \quad (5.3\text{-}226)$$

If the binary flag $f_{UX}(n)$ is set to 0 and TD stereo mode was selected in the previous frame, then the auxiliary switching flag from the TD energy analysis module $f_{TDM}(n-1)$ (see clause 5.3.2.3.3 for the details of calculation) is analyzed to select the stereo mode in the current frame. This is done as follows

$$M_{SMODE}(n) \leftarrow \begin{cases} \text{TD} & \text{if } f_{SM}(n) = 1 \text{ AND } f_{TDM}(n-1) = 1 \\ \text{DFT} & \text{otherwise} \end{cases} \quad (5.3\text{-}227)$$

The parameter $f_{TDM}(n)$ is updated in every frame in the TD stereo mode only. The updating of this parameter is described in detail in clause 5.3.2.3.

If the binary flag $f_{UX}(n)$ is set to 0 and the stereo mode in the previous frame was the DFT sttereo mode, no stereo mode switching is done and the DFT stereo mode is selected in the current frame as well.

As can be seen from Figure 5.3-33 the binary output of the cross-talk detector in the TD stereo mode can only be set to 1 when cross-talk content is detected in the current frame. As a consequence, the initial stereo mode selection logic described above cannot select the DFT stereo mode when the cross-talk detector indicates single-talk content. An additional mechanism has been implemented for switching back from the TD stereo mode to the DFT stereo mode when single-talk content is detected. The mechanism is described in the following text.

If TD stereo mode has been selected in the previous frame and the initialization logic described above selected TD stereo mode in the current frame as well, then the stereo mode may be changed from the TD stereo mode to the DFT stereo mode. The change of stereo mode is conditioned on the binary output of the cross-talk detector $c_{XTALK}(n)$ as well as other auxiliary parameters. The stereo mode is changed if the following complex condition is fulfilled

$$M_{SMODE}(n) \leftarrow \text{DFT if } \left[\begin{array}{c} M_{SMODE}(n) = \text{TD AND} \\ M_{SMODE}(n-1) = \text{TD AND} \\ c_{XTALK}(n) = 1 \text{ AND} \\ f_{UX}(n-1) = 1 \text{ AND} \\ c_{TD}(n-1) > 15 \text{ AND} \\ c_{DFT}(n-1) > 3 \text{ AND} \\ clas(n-1) \in \left(\begin{array}{c} \text{UNVOICED\_CLAS} \\ \text{UNVOICED\_TRANSITION} \\ \text{VOICED\_TRANSITION} \end{array}\right) \text{ AND} \\ (brate_{total} \geq 16400 \text{ OR } wscr_{XTALK}(n-1) < 0.01) \end{array}\right] \quad (5.3\text{-}228)$$

where $brate_{total}$ is the total bitrate of the IVAS encoder and $clas(n)$ is the the frame type of the Frame Erasure Concealment (FEC) module of the primary channel in the TD stereo mode, described in clause 5.1.13.3 of [3]. The parameters $c_{TD}(n)$ and $c_{DFT}(n)$ are the counters of frames in the TD stereo mode and the DFT stereo mode, respectively. The counters $c_{TD}(n)$ and $c_{DFT}(n)$ are updated in each frame as part of the TD energy analysis module. The updating of these counters is described in detail in the following text.

When the IVAS encoder runs in the TD stereo mode several auxiliary parameters are calculated as part of the TD energy analysis module. These auxiliary parameters are used to improve the stability of the DFT/TD stereo mode selection mechanism.

For certain special types of frames the TD stereo mode runs in s so-called “regular submode”. The regular sub-ode is usually applied for short transition periods before switching from the TD stereo mode to the DFT stereo mode. Whether or not the TD stereo mode is runing in the regular submode is indicated by the binary flag $m_{TD}(n)$. The binary flag $m_{TD}(n)$ is first initialized in each frame as follows

$$m_{TD}(n) = f_{TDM}(n-1) \tag{5.3-229}$$

where $f_{TDM}(n)$ is the auxiliary switching flag described later in this clause in eq. (5.3-233). The binary flag $m_{TD}(n)$ is set to 0 or 1 in frames where the binary flag $f_{UX}(n) = 1$. The condition for setting $m_{TD}(n)$ is defined as follows

$$m_{TD}(n) \leftarrow \begin{cases} 1 & if\ f_{TDM}(n-1) = 1 \text{ OR } M_{SMODE}(n-1) \neq \text{TD OR } c_{TD}(n-1) < 5 \\ 0 & \text{otherwise} \end{cases} \tag{5.3-230}$$

If the binary flag $f_{UX}(n) = 0$ then the binary flag $m_{TD}(n)$ is not changed.

The TD energy analysis module contains two counters, $c_{TD}(n)$ and $c_{DFT}(n)$. The counter $c_{TD}(n)$ counts the number of consecutive frames when the IVAS coder runs in the TD stereo mode. Thus, the counter $c_{TD}(n)$ is set to 0 in frames where the IVAS coder runs in the DFT stereo mode. The counter $c_{TD}(n)$ is incremented by 1 in eeach frame where the IVAS coder runs in the TD stereo mode. That is

$$c_{TD}(n) = \begin{cases} c_{TD}(n-1) + 1 & \text{if } f_{UX}(n) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.3-231}$$

Essentially, $c_{TD}(n)$ contains the number of frames since the last DFT to TD stereo mode switching point. Note, that the counter $c_{TD}(n)$ is limited by the threshold of 100. The counter $c_{DFT}(n)$ counts the number of consecutive frames when the IVAS coder runs in the DFT stereo mode. Thus, the counter $c_{DFT}(n)$ is set to 0 in every frame where TD stereo mode has been selected in the encoder and is incremented by 1 in every frame where DFT stereo mode has been selected. That is

$$c_{DFT}(n) = \begin{cases} c_{DFT}(n-1) + 1 & \text{if } f_{TDM}(n) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.3-232}$$

Essentially, $c_{DFT}(n)$ contains the number of frames since the last TD to DFT stereo mode switching. Note, that the counter $c_{DFT}(n)$ is limited by the threshold of 100.

The last auxiliary parameter calculated in the TD energy analysis module is the auxiliary switching flag $f_{TDM}(n)$. This parameter is initialized with the binary flag $f_{UX}(n)$. That is

$$f_{TDM}(n) = f_{UX}(n) \tag{5.3-233}$$

The auxiliary switching flag is set to 0 when the left and the right channel of the input stereo signal are near out-of-phase (NOOP). The details of NOOP signal detection are described in detail in clasue 5.3.2.3.4. In case of NOOP it is desirable to invert the meaning of the primary and the secondary channel in the TD stereo mode. When NOOP is detected the binary flag $s2m(n)$ in the current frame is set to 1, otherwise it is set to zero. The auxiliary switching flag in the TD stereo mode is set to zero when the parameter $s2m(n)$ is set to 1. That is

$$f_{TDM}(n) \leftarrow 0 \quad \text{if } s2m(n) = 1 \tag{5.3-234}$$

If the parameter $s2m(n)$ is zero, then the auxiliary switching flag can be reset to zero based on the following set of conditions

$$f_{TDM}(n) \leftarrow 0 \text{ if } \left[\begin{array}{c} m_{TD}(n) = 1 \text{ AND} \\ c_{LRTD}(n) > 10 \text{ AND} \\ \left[\begin{array}{l} \text{VAD} = 0 \text{ OR} \\ c_{UNCLR}(n) = 0 \text{ AND } (scr_{XTALK}(n) < -0.8 \text{ OR } wscr_{XTALK}(n) < -0.13) \text{ OR} \\ c_{UNCLR}(n) = 1 \text{ AND } clas(n-1) = \text{UNVOICED\_CLAS AND } |wscr_{UNCLR}(n)| < 0.005 \end{array}\right] \end{array}\right] \tag{5.3-235}$$

Furthermore, the auxiliary switching flag $f_{TDM}(n)$ can also be reset to 0 based on the following set of conditions

$$f_{TDM}(n) \leftarrow 0 \text{ if } \begin{bmatrix} m_{TD}(n) = 0 \text{ AND} \\ \begin{bmatrix} \text{VAD} = 0 \text{ OR} \\ c_{UNCLR}(n) = 0 \text{ AND } (scr_{XTALK}(n) \leq 0 \text{ OR } wscr_{XTALK}(n) \leq 0.1) \text{ OR} \\ c_{UNCLR}(n) = 1 \text{ AND } clas(n-1) = \text{UNVOICED\_CLAS AND } |wscr_{UNCLR}(n)| < 0.025 \end{bmatrix} \end{bmatrix} \quad (5.3\text{-}236)$$

Note, that in the two sets of conditions defined above the condition

$$clas(n-1) = \text{UNVOICED\_CLAS}$$

refers to the $clas$ parameter calculated in the preprocessing module of the primary channel when the codec is in the DFT stereo mode. In case the codec is in the TD stereo mode, the condition is replaced with

$$clas_P(n-1) = \text{UNVOICED\_CLAS AND } clas_S(n-1) = \text{UNVOICED\_CLAS}$$

where the subscripts “P” and “S” have been added as a reference to the primary and the secondary channel in the TD stereo mode, respectively.

5.3.3 MDCT-based stereo

5.3.3.1 General Overview

For bitrates starting from 48 kbps up to 256 kbps, the MDCT-based stereo coding mode is used. MDCT stereo is based on the EVS high-rate MDCT-based TCX coder, described in clause 5.3.3 of [3]. This stereo mode offers discrete coding due to the higher bitrate compared to the unified stereo coding mode where the coding is parametric.

In Figure 5.3-37 a block diagram of the MDCT-based encoder is depicted. The input audio signal of each stereo channel is first transformed to the MCLT domain from which the MDCT-transform is further pre-processed to be perceptually flat (whitened) before any stereo processing is applied. The details of this pre-processing are described in 5.3.3.3. On the perceptually whitened signals the stereo processing is applied. The details of the stereo processing are described in clause 5.3.3.4. Finally, stereo IGF encoding is applied, and the obtained signals of each channel are quantized and encoded as described in clause 5.3.3.8.

5.3.3.2 ITD compensation

For the mid bitrates 48kbps and 64kbps the incoming signals are time aligned with respect to the inter-channel time difference (ITD) to achieve optimal energy compaction and larger coding gain. To achieve this, the same processing is done as for DFT-stereo, namely, the STFT analyis described in clause 5.3.2.4.3 and the ITD estimation using the Generalized Cross-Correlation Phase Transform (GCC-PHAT) as described in clause 5.3.2.4.4. The ITD compensation is done in the time-domain using the time-domain ITD extrapolation method described in clause 5.3.2.4.2.2 using the ITD estimation of the previous frame.

Different from unified stereo where ITDs up to 200 samples are compensated, the maximum is reduced to 80 samples in MDCT stereo.

Furthermore, time-domain ITD compensation as described for DFT-based stereo in clause 5.3.2.4.2 is also employed for MDCT-based stereo. The only difference is the number of samples over which equations (5.3-91) and (5.3-92) are computed. For MDCT stereo, $t = 0, \ldots, frame + frame - ITD$ with $frame$ the frame size in samples and ITD the ITD of the previous frame in samples.

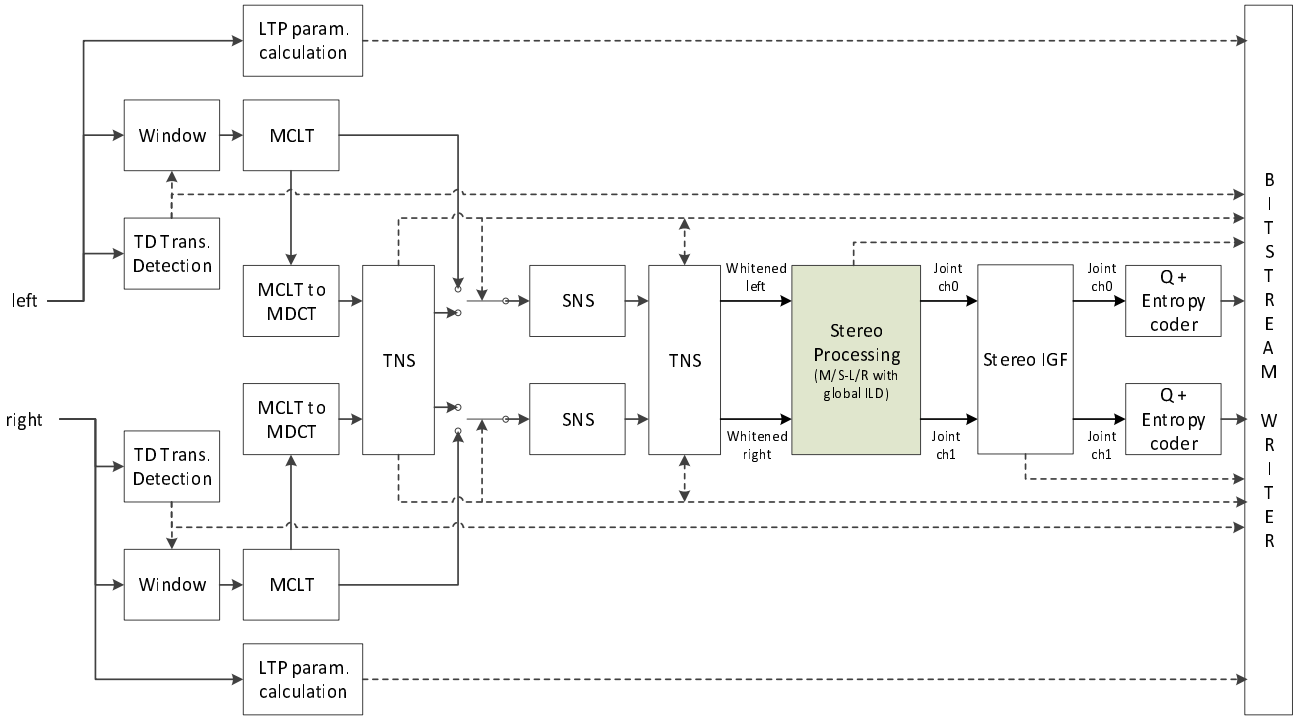


Figure 5.3-37: Block diagram of the MDCT-based stereo encoder

5.3.3.3 MDCT core pre-processing

5.3.3.3.1 General

The pre-processing of the input time-domain audio signal prior to any stereo processing comprises of the transform from time-domain to frequency-domain, more precisely to the MDCT-domain, and further processing of the first channel and second channel by applying Spectral Noise Shaping (SNS) operation on the transformed audio signal of each channel. SNS on the encoder side in fact performing flattening of the spectrum. We may refer to SNS also as Frequency-Domain Noise Shaping. Furthermore, depending on the input signal characteristics, temporal-noise shaping may be applied either prior or after SNS on the transformed signal.

5.3.3.3.2 TCX-LTP parameter estimation

The LTP filter parameters (pitch lag and gain) are first estimated and encoded into the bitstream such that the TCX LTP postfilter at the decoder side described in clause 6.9.2.2 of [3] can use them. The detailed description of the LTP filter parameter estimation can be found in clause 5.1.14.1.1.1 of [3].

5.3.3.3.3 Time-to-frequency transformations

5.3.3.3.3.1 Overview

The time-domain input audio signals of the first and second channel are then transformed to the frequency domain by performing an MDCT transform. The MDST transform is also performed, since that is needed to compute the power spectra used for SNS and IGF, as well as the modulated complex lapped transform (MCLT) spectra required for mid-high bitrate pre-processing. Note that, for each channel i, the MCLT spectrum is given by $X_i^{MDCT} + jX_I^{MDST}$, i.e., the MDCT represents the real(-valued) part of the MCLT while the MDST represents the imaginary part of the MCLT.

The time-to-frequency domain transform is the same as for the EVS TCX high rate MDCT-based coding mode. The details regarding the windowing, long and short block transforms, transient-depended overlap and transform length, transient detection, window transitions and the MDST transform can be found in clauses 5.3.2.1, 5.3.2.2, 5.3.2.3, 5.3.2.4.2 and 5.3.2.6 of [3]. Different to clause 5.3.2.5 of [3], transition between long and short windows is implemented as described in clause 5.3.3.3.3.2.

5.3.3.3.3.2 Transition between long and short blocks with double overlap

As in [3], the overlap mode FULL is used for transitions between long ALDO windows and short symmetric windows (HALF, MINIMAL) for short block configurations (TCX10, TCX5). However, longer and different overlap is used than in [3].

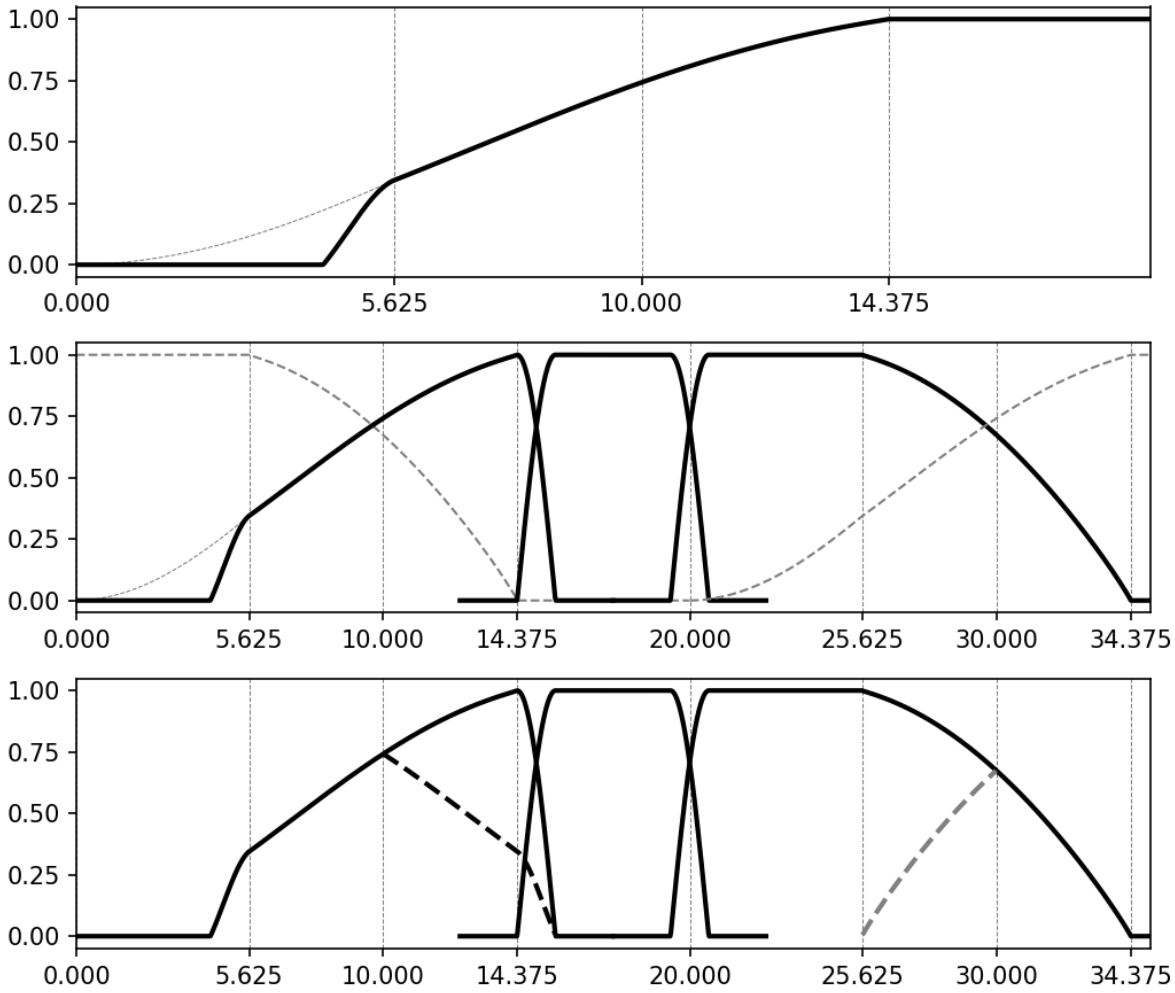


Figure 5.3-38: Window transition from ALDO to short blocks in the encoder

As described in tables 79 and 80 in 5.3.2.3 of [3], the overlap mode FULL is used for the left part of some short windows depending on the locations of the detected transients. Thanks to the multi-overlap where three window functions overlap with each other at the same time period (as can be seen in the bottom plot in Figure 5.3-38) there is no need to increase the look-ahead for the transient detection as is done in the traditional window switching and thus a delay due to a transient look-ahead is reduced.

The left overlap of the transition window (FULL) has a length of 8.75 ms (full overlap) + 1.25 ms (minimal overlap) = 10 ms. For deriving the left overlap of the transition window (FULL), the left slope of the ALDO analysis window (long slope, as shown with light grey dashed curve in the top plot of Figure 5.3-38) is shortened to 10 ms by zeroing out the first 4.375 ms. The first 1.25 ms of the remaining non-zero slope are multiplied with the 1.25 ms MINIMAL overlap slope. The resulting 10 ms slope is depicted with the black curve (4.375 ms-14.375 ms) in the top plot of Figure 5.3-38.

The window sequence FULL, TCX5, MINIMAL, TCX5, MINIMAL, TCX10, FULL corresponding to the overlap code 00-10 or 10-10 is shown in the middle plot of Figure 5.3-38. Details on the window sequences and the overlap codes are available in 5.3.2.3 of [3].

To correctly prepare the audio signal for generating spectral values using the MDCT, first the left overlap part of the first window and the right overlap part of the third window are folded-in (shown by dashed lines in the bottom plot of Figure 5.3-38), treating the three windows as a single window in this preprocessing step. After this preprocessing step each of the three windows is processed separately, by windowing and folding-in of the overlap portions between them.

For the right part of the transition (dashed grey line on the left in the middle plot of Figure 5.3-38), ALDO analysis window (short slope) of length 8.75 ms is used as in [3].

The 10 ms left overlap of the transition window (FULL) is also used for the sequences FULL, TCX10, HALF, TCX5, MINIMAL, TCX5, HALF and FULL, TCX10, MINIMAL, TCX5, MINIMAL, TCX5, MINIMAL.

Milliseconds are shown on the x axis and scalar values of the windows on the y axis in Figure 5.3-38.

5.3.3.3.3.3 Optimized TCX window overlap for long-term transient signals

For signals with longer-term transient behaviour, e.g. applause, the selected window overlap is optimized compared to [3]. For certain conditions – which will be described in the following – the used overlap is switched to HALF overlap, independent of the selected TCX mode and attack index (see table 78 in 5.3.2.3 of [3] for the general selection scheme).

For the detection of long-term transient behaviour, the current temporal flatness measure (TFM) value and a memory variable $tf_{mem}[i]$ derived from the previous frame are used. $TFM[i]$ is calculated according to equation 39 in clause 5.1.8. of [3] with $N_{past} = 0$. The detection is done independently for each channel with *i* signifying the channel index. The memory variable $tf_{mem}[i]$ is initially set to 0.75 and will be reset to 0.75 if the previous frame was not TCX, e.g. in case of bitrate switching.

If

$$\frac{1}{TFM[i]} > tf_{mem[i]}, \tag{5.3-237}$$

the long-term TFM value $TFM_{lt}[i]$ is calculated as

$$TFM_{lt}[i] = tf_{mem}[i](0.96875 - f) + \frac{1}{TFM[i]}(0.03125 - f) \tag{5.3-238}$$

with

$$f = \left(\frac{1}{TFM[i]} - tf_{mem}[i]\right)^2. \tag{5.3-239}$$

If

$$\frac{1}{TFM[i]} \leq tf_{mem}[i], \tag{5.3-240}$$

$TFM_{lt}[i]$ is calculated as

$$TFM_{lt}[i] = 0.96875 \cdot tf_{mem}[i] + 0.03125 \cdot \frac{1}{TFM[i]}. \tag{5.3-241}$$

If

$$TFM_{lt}[i] < 0.5625, \tag{5.3-242}$$

the current frame is classified as showing long-term transient characteristics. If the overlap of the current frame is not set to MINIMAL and if also

$$C_{LTP,norm}^{(prev)}[i] < 0.5625, \tag{5.3-243}$$

is true, where $C_{LTP,norm}^{(prev)}[i]$ is the normalized maximum correlation of the LTP parameter estimation of the previous frame (see clause 5.1.14.1.1.1 of [3]), the overlap is changed to HALF overlap.

Finally, both $tf_{mem}[i]$ and $C_{LTP,norm}^{(prev)}[i]$ are updated. First, $tf_{mem}[i]$ is set to the newly calculated value of $TFM_{lt}[i]$ and a lower bound is applied as

$$tf_{mem}[i] = \max(0.015625, TFM_{lt}[i]). \tag{5.3-244}$$

Later, when values for both channels are available, the value of tf_{mem} is synchronized between the channels by calculating the RMS of the individual values in each channel:

$$tf_{mem}[0] \leftarrow \sqrt{0.5(tf_{mem}[0]^2 + tf_{mem}[1]^2)} \tag{5.3-245}$$

$$tf_{mem}[1] \leftarrow \sqrt{0.5(tf_{mem}[0]^2 + tf_{mem}[1]^2)} \quad (5.3\text{-}246)$$

Similarly, $C_{LTP,norm}^{(prev)}$ is synchronized between both channels by taking the mean value:

$$C_{LTP,norm}^{(prev)}[0] \leftarrow 0.5\left(C_{LTP,norm}^{(prev)}[0] + C_{LTP,norm}^{(prev)}[1]\right) \quad (5.3\text{-}247)$$

$$C_{LTP,norm}^{(prev)}[1] \leftarrow 0.5\left(C_{LTP,norm}^{(prev)}[0] + C_{LTP,norm}^{(prev)}[1]\right) \quad (5.3\text{-}248)$$

5.3.3.3.4 Mid-high bitrate pre-processing

5.3.3.3.4.1 General

For the mid-high stereo bitrates, namely 48 and 64 kbps, some additional pre-processing steps are applied in order to improve energy compaction of the stereo processing and increase perceptual coding gain. The first step is kernel switching, allowing to switch between different MDCT and MDST based transform types depending on the frame-wise overall phase difference between the two input channel signals. The second step is stereo pre-processing, where the channel signals are phase aligned in the higher bands depending on frame-wise inter-channel correlation statistics. A third step, where the channel signals are additionally level fine-aligned in the higher bands after stereo pre-processing, is applied as part of the global inter-channel level difference (ILD) normalization during the stereo processing.

In the following, the term "mid-high stereo" shall indicate the operating mode defined in the previous clause, i.e., MDCT-based two-channel stereo coding at 48 and 64 kbps.

5.3.3.3.4.2 Kernel switching

5.3.3.3.4.2.1 Overview

The purpose of the kernel switching encoder is to select, signal adaptively on a subframe basis and by means of a kernel type controller, either the MDCT-IV or MDST-IV as primary transform (to be subjected to further processing in the following subclauses) in each channel. These MDCT-IV and MDST-IV transforms, which have been calculated (as part of the MCLT) for each subframe according to the time-spectrum conversion described in clause 5.3.3.3, represent a first group of transforms, with transform kernels having different time-domain aliasing (TDA) symmetries (even and odd, or vice versa) at both sides. To allow for TDA cancellation when switching, in a channel, from one transform to the other, the kernel switching encoder selects either the MDCT-II or MDST-II transform as a transitional transform, based on the transform chosen in the previous subframe. The MDCT-II and MDST-II transforms, which have not yet been calculated and are determined according to this subclause when necessary, belong to a second group of transforms, with transform kernels having the same TDA symmetries (both even or both odd) at both sides.

Further explanatory details are published in [20], and four kernel switching specific constants are defined as follows:

- MDCT_IV = 0
- MDST_II = 1
- MDCT_II = 2
- MDST_IV = 3

5.3.3.3.4.2.2 Subframe-wise control of transform kernel type

5.3.3.3.4.2.2.1 Initialization

At the beginning of an encoding run (i.e., when initializing the TCX encoder) or when encoding a TCX frame following an ACELP encoded frame (i.e., when last_core == ACELP_CORE) or an LFE frame (i.e., with cpe_id * CPE_CHANNELS + ch >= nChannels), reset for each channel:

- kernel_switch_corr_past = 0.0
- kernel_symmetry_past = 0
- kernel_type[0] = 0

When encoding a TCX subframe at index n in a mode which is not mid-high stereo, set for each channel:

- kernel_switch_corr_past = 0.0
- kernel_type[n] = (kernel_symmetry_past && element_mode == IVAS_CPE_MDCT ? 3 – mct_on : 0)

Otherwise (i.e., when encoding a mid-high stereo TCX subframe), if at subframe index n, the transform_type[n] values of both channels differ, set for each channel:

- kernel_switch_corr_past = 0.0
- kernel_type[n] = (kernel_symmetry_past ? 3 : 0)

Otherwise (i.e., if the two transform_type[n] values are equal), determine value *switchKernel* by means of the following kernel type controller.

Let, for the given subframe at n and depending on the value of transform_type[n],

- sigR0 be the MDCT samples of the first (i.e., left) channel, returned by the MCLT time-spectrum converter,
- sigR1 be the MDCT samples of the second (right) channel, returned by the MCLT time-spectrum converter,
- sigI0 be the MDST samples of the first (left) channel, returned by the MCLT time-spectrum converter,
- sigI1 be the MDST samples of the second (right) channel, returned by the MCLT time-spectrum converter,
- nSamplesCore be the number of core-coded MCLT-domain samples below the IGF start frequency, and
- nSamplesMax be the transform length (i.e., number of MCLT-domain samples including the IGF range),

with the MCLT time-spectrum converter described in subclause 5.2.3.4.3.3. Set in addition the following helper variables:

- bitRateMode = (element_brate · L_subframe) / nSubframes;
- anaLength = min(min(nSamplesCore, nSamplesMax), (bitRateMode >> 17) & 0xFFFE);
- cov00 = 0.0
- cov90 = 0.0
- sumR_0 = 0.0
- sumR_1 = 0.0
- sumI_0 = 0.0
- sumI_1 = 0.0

If transform_type[n] equals TCX_5, firstly set anaLength = anaLength>>1. Secondly, set:

- cov00 += $\sum_{s=1}^{anaLength-1}(\text{sigR0}[s] \cdot \text{sigR1}[s] + \text{sigI0}[s] \cdot \text{sigI1}[s])$
- cov90 += $\sum_{s=1}^{anaLength-1}(\text{sigR0}[s] \cdot \text{sigI1}[s] - \text{sigI0}[s] \cdot \text{sigR1}[s])$
- sumR_0 += $\sum_{s=1}^{anaLength-1} \text{sigR0}[s] \cdot \text{sigR0}[s]$
- sumR_1 += $\sum_{s=1}^{anaLength-1} \text{sigR1}[s] \cdot \text{sigR1}[s]$
- sumI_0 += $\sum_{s=1}^{anaLength-1} \text{sigI0}[s] \cdot \text{sigI0}[s]$
- sumI_1 += $\sum_{s=1}^{anaLength-1} \text{sigI1}[s] \cdot \text{sigI1}[s]$

Subsequently, if transform_type[n] equals TCX_5, set

- anaLength += (nSamplesMax >> 1)

- s = 1 + (nSamplesMax >> 1)

Otherwise (i.e., if transform_type[n] does not equal TCX_5), set

- s = (nSamplesMax < 512 ? 2 : 4)

Once the above has been completed, set for any transform type:

- cov00 += $\sum_{t=s}^{anaLength-1}(\text{sigR0}[t] \cdot \text{sigR1}[t] + \text{sigI0}[t] \cdot \text{sigI1}[t])$
- cov90 += $\sum_{t=s}^{anaLength-1}(\text{sigR0[t]} \cdot \text{sigI1[t]} - \text{sigI0[t]} \cdot \text{sigR1[t]})$
- sumR0 += $\sum_{t=s}^{anaLength-1} \text{sigR0[t]} \cdot \text{sigR0[t]}$
- sumR1 += $\sum_{t=s}^{anaLength-1} \text{sigR1[t]} \cdot \text{sigR1[t]}$
- sumI0 += $\sum_{t=s}^{anaLength-1} \text{sigI0[t]} \cdot \text{sigI0[t]}$
- sumI1 += $\sum_{t=s}^{anaLength-1} \text{sigI1[t]} \cdot \text{sigI1[t]}$

Then set:

- r00 = cov00 / ($\sqrt{(\text{sumR0} \cdot \text{sumR1})}$ + $\sqrt{(\text{sumI0} \cdot \text{sumI1})}$ + 1.0)
- r90 = cov90 / ($\sqrt{(\text{sumR0} \cdot \text{sumI1})}$ + $\sqrt{(\text{sumI0} \cdot \text{sumR1})}$ + 1.0)
- sumI_0 = max(0.0, | r90 | – | r00 |)
- sumI_1 = | kernel_switch_corr_past |

To conclude the operation of the kernel type controller, set:

- kernel_switch_corr_past = ($0.875 \cdot \text{sumI}_1 + 0.125 \cdot \text{sumI}_0$) · sign(r90)
- *switchKernel* = (($\text{sumI}_0 > 0.5 - 0.25 \cdot \text{sumI}_1$ || | r90 | > $0.75 - 0.5 \cdot \text{sumI}_1$) && ($\text{sumI}_1 > 0.0625$) ? sign(r90) : 0)

Having determined the value of *switchKernel* using the kernel type controller, if *switchKernel* does not equal zero, then set for the first channel:

- kernel_type[n] = (kernel_symmetry_past ? 3 : 1) – max(0, *switchKernel*)

and set for the second channel:

- kernel_type[n] = (kernel_symmetry_past ? 2 : 0) + max(0, *switchKernel*)

in order to apply signal-adaptive MDST-IV coding in one of the channels. Otherwise (i.e., if *switchKernel* equals zero), set for each channel:

- kernel_type[n] = (kernel_symmetry_past ? 2 : 0);

in order to switch back to MDCT-IV coding in all channels. Having selected the value of kernel_type[n] in all channels, set for each channel:

- kernel_symmetry_past = kernel_type[n] & 1;

5.3.3.3.4.2.2.2 Subframe-wise update of MDCT/MDST transforms

As noted in the overview clause 5.3.3.3.4.2.1, a certain sequence of subframe-wise transform kernel types must be enforced in order to allow for TDA cancellation during the decoder-side windowing and overlap-and-add operation. To this end, a remapping or recalculation of the channel transforms, by means of the following kernel type adaptive time-spectrum converter, may be required.

Let, for each channel and subframe at n and depending on the value of transform_type[n],

- sigR be the MDCT-IV samples of the given channel, returned by the MCLT time-spectrum converter
- sigI be the MDST-IV samples of the given channel, returned by the MCLT time-spectrum converter
- nSamplesMax be the transform length (i.e., number of MCLT-domain samples including the IGF range)

with the MCLT time-spectrum converter described in subclause 5.3.3.3.3. Set, further, the following helper variable:

- kernelType = kernel_type[n]; i.e., the above control information to guide the channel transform calculation

If kernelType equals MDCT_IV, no updates are required since sigR and sigI already contain the MDCT-IV and MDST-IV samples, respectively.

If kernelType equals MDST_IV, the real and imaginary MCLT parts need to be exchanged, i.e., the samples in sigR and sigI must be remapped:

- sigTemp = –sigR[*s*]
- sigR[*s*] = sigI[*s*]
- sigI[*s*] = sigTemp

with *s* = 0, 1, ... nSamplesMax – 1.

Otherwise (i.e., if kernelType equals MDCT_II or MDST_II), the chosen transform for the given channel and subframe belongs to the second group of transforms, as described above. Since these type-II transforms have not been calculated during MCLT-based time-spectrum conversion, recalculations of the transforms of the windowed time-domain samples are required. To this end, the MDCT transform, stored in sigR and calculated as the real part of the MCLT time-spectrum conversion according to clause 5.3.3.3.3 and the subclauses of clause 5.3.2 of [3] referenced therein, is recalculated, with the following changes.

If kernelType equals MDCT_II, an MDCT-II transform is obtained in the same way as an MDCT, with two exceptions:

- The TDAC equation is, instead of by equation (907) of [3], described by
  $\overline{x} = \begin{bmatrix} 0 & 0 & J_{L/2} & I_{L/2} \\ I_{L/2} & J_{L/2} & 0 & 0 \end{bmatrix} x_w$
- The DCT_{IV} given by equation (890) of [3] is replaced by a DCT_{II} kernel:
  $sigR[k] = X_{C^2}(k) = \sum_{n=0}^{L-1} \tilde{x}(n) \cos\left[\left(n + \frac{1}{2}\right)(k+0)\frac{\pi}{L}\right]$
  where the resulting value for the DC sample sigR[0] is additionally multiplied by 0.5.

If kernelType equals MDST_II, an MDST-II transform is obtained in the same way as the MDST, with two exceptions:

- The TDAC equation is, instead of by equation (907) of [3], described by
  $\overline{x} = \begin{bmatrix} 0 & 0 & J_{L/2} & -I_{L/2} \\ -I_{L/2} & J_{L/2} & 0 & 0 \end{bmatrix} x_w$
- The DSTIV given by equation (908) of [3] is replaced by a DSTII kernel:
  $sigR[k] = XS^2(k) = \sum_{n=0}^{L-1} \tilde{x}(n) \sin\left[\left(n + \frac{1}{2}\right)(k+1)\frac{\pi}{L}\right]$
  where the resulting value for sigR[nSamplesMax – 1] is additionally multiplied by 0.5.

In both cases, *k* = 0, 1, ... nSamplesMax – 1, and the transform follows the remaining MCLT related parameters such as transform size or windowing.

The MDST transform, stored in sigI and calculated as the imaginary part of the MCLT time-spectrum conversion, is not recalculated in this case to reduce the computational complexity during encoding. Note that, if transform_type[n] equals TCX_5, only the first TCX5 transform in the subframe is recalculated since the second TCX5 transform in the subframe is always either of type MDCT_IV or type MDST_IV.

Hence, if transform_type[n] equals TCX_5 and kernelType equals MDST_II, the samples in the upper half of sigR and sigI must be remapped:

- sigTemp = –sigR[*t*]
- sigR[*t*] = sigI[*t*]

- sigI[t] = sigTemp

with t = nSamplesMax / 2, nSamplesMax / 2 + 1, ... nSamplesMax – 1, since the second TCX5 transform is MDST_IV.

5.3.3.3.4.2.2.3 Encoding of frame-wise kernel switching parameters

For each channel belonging to a channel element with element_mode == IVAS_CPE_MDCT, the kernel type switching parameter value stored in kernel_type[n] for each subframe at n is encoded into the bitstream, as control information, to properly guide all inverse transform operations during the decoder-side kernel type adaptive spectrum-time conversion.

Having written into the bitstream the TCX transform windowing parameters (tcx_curr_overlap_mode and, if applicable, tcx_last_overlap_mode), the value of kernel_type[0], i.e., the transform kernel type for the first subframe of the given channel and frame, is written using i bits, with

i = (last_core ≠ ACELP_CORE ? 2 : 1);

If transform_type[n] equals TCX_5 or TCX_10 for any subframe at n in the given channel and frame (i.e., if the frame is not a TCX20 frame), the result of calculation kernel_type[1] & 1, i.e., the least significant bit of the kernel type for the second subframe, is additionally written using 1 bit.

5.3.3.3.4.3 Stereo pre-processing

5.3.3.3.4.3.1 Overview

The purpose of the encoder-side stereo pre-processing algorithm (Steppa) is to increase inter-channel correlation in the higher frequencies where the human auditory system is relatively insensitive to inter-aural phase differences. The increased cross-channel correlation can then be exploited by the ILD normalized mid/side stereo processing of clause 5.3.3.4.5, thereby increasing the perceptual coding efficiency. To maintain high audio quality, only the channels' higher-frequency subband phases are modified, while the channels' higher-frequency subband magnitudes are left unchanged.

5.3.3.3.4.3.2 Subframe-wise activation decision

At the beginning of an encoding run (i.e., when initializing the TCX encoder) or when encoding a TCX frame following an ACELP encoded frame (i.e., when last_core == ACELP_CORE) or an LFE frame (i.e., with cpe_id * CPE_CHANNELS + ch >= nChannels), reset for each channel: ste_corr_past = 0.

When encoding a subframe at index n in a mode which is not mid-high stereo, or if at n, the transform_type[n] values or the current frame's overlap modes or the previous frame's overlap modes of both channels differ, set for each channel: ste_corr_past = 0.

Otherwise (i.e., when encoding a mid-high stereo subframe and if all three parameters are equal in both channels), apply the following Steppa strength detection algorithm to each (possibly kernel switching modified) MCLT transform at n.

Let, for the given subframe at n and depending on the value of transform_type[n],

- sigR0 be the samples of the real part of the MCLT of the first (i.e., left) channel after kernel switching,
- sigR1 be the samples of the real part of the MCLT of the second (right) channel after kernel switching,
- sigI0 be the samples of the imaginary part of the MCLT of the first (left) channel after kernel switching,
- sigI1 be the samples of the imaginary part of the MCLT of the second (right) channel after kernel switching,
- nSamplesCore be the number of core-coded MCLT-domain samples below the IGF start frequency, and
- nSamplesMax be the transform length (i.e., number of MCLT-domain samples including the IGF range),

with the adaptive kernel switching algorithm described in clause 5.3.3.3.4.2.

Set, also, the following helper variables:

- bitRateMode = (element_brate · L_subframe) / (transform_type[n] == TCX_5 ? 4 : nSubframes);
- fadeInLen = (nSamplesMax < 512 ? (nSamplesMax < 256 ? 10 : 20) : 40);

- fadeInOff = ((bitRateMode · 3) >> 19) & (nSamplesMax < 512 ? 0xFFFF : 0xFFFE);

Calculate, and store in variable rLR, the Pearson correlation coefficient across the two channels' magnitude spectra, $\sqrt{(\,\mathrm{sigR0}[s]^2 + \mathrm{sigI0}[s]^2\,)}$ and $\sqrt{(\,\mathrm{sigR1}[s]^2 + \mathrm{sigI1}[s]^2\,)}$, in the frequency index range fadeInOff ≤ *s* < nSamplesCore. Also store, in variable *den*, the denominator of the rLR calculation, and assume that rLR = 0 when *den* ≤ 0.0.

Then set

- *chanCorrSign* = (–rLR > 2^{23} / *den* ? –1 : 1)
- *corr* = (rLR < 0.0 ? 0.0 : rLR^2)

If *corr* is less than or equal to 0.75 and ste_corr_past equals zero, set ste_corr_past = 0 and skip the application of the following subclause (deactivate Steppa).

Otherwise, apply the Steppa method as described in the following subclause, using the above variables, and afterwards set ste_corr_past = (*corr* > 0.75 ? 1 : 0).

5.3.3.3.4.3.3 Subframe-wise application of Steppa

Set limitWeight = (*nrg* > 0.875 ? 1 : 0), with *nrg* being the average of the current and previous frames' normalized correlation values $C_{LTP,norm}$ calculated for both channels during the LTP filtering described in clause 5.1.14.1.1 of [3], referenced in clause 5.3.3.3.4.2 herein. Set, also, the following helper variables:

- stepWeightI = (limitWeight || (*corr* > 0.75 && !ste_corr_past) || (*corr* ≤ 0.75 && ste_corr_past) ? 1 : 2)
- stepWeightD = 4 – stepWeightI
- xTalkI = stepWeightI
- xTalkD = stepWeightD · (2 · fadeInLen – 1)

To smoothly transition between unchanged and Steppa modified subband samples, perform the three operations

- *applyStereoPreProcessingCplx*(sigR0[*s*], sigR1[*s*], sigI0[*s*], sigI1[*s*], xTalkI, xTalkD, *chanCorrSign*)
- xTalkI += stepWeightI
- xTalkD –= stepWeightD

for each subband index *s* = fadeInOff + 1, fadeInOff + 2, ... fadeInOff + fadeInLen – 1.

Then perform operation

applyStereoPreProcessingCplx(sigR0[*s*], sigR1[*s*], sigI0[*s*], sigI1[*s*], xTalkI, xTalkD, *chanCorrSign*);

for each *s* = fadeInOff + fadeInLen, fadeInOff + fadeInLen + 1, ... min(nSamplesCore, nSamplesMax) – 1.

If nSamplesCore < nSamplesMax (i.e., if IGF is used), smoothly transition at the high frequencies by performing

- xTalkI –= stepWeightI
- xTalkD += stepWeightD
- *applyStereoPreProcessingCplx*(sigR0[*s*], sigR1[*s*], sigI0[*s*], sigI1[*s*], xTalkI, xTalkD, *chanCorrSign*)

for each *s* = min(nSamplesCore, nSamplesMax), ... min(nSamplesCore, nSamplesMax) + fadeInLen – 1, i.e., in the first fadeInLen subbands of the IGF range, assuming the IGF range comprises at least fadeInLen samples.

In the above description, fadeInLen represents a spectral range of 1kHz, and function *applyStereoPreProcessingCplx*( ) defines the following MCLT sample-wise channel phase alignment operation, using the six additional helper variables

- absR[0] = | sigR0[*s*] |
- absR[1] = | sigR1[*s*] |

- dmxR[0] = sigR0[*s*] · xTalkD + *sign* · sigR1[*s*] · xTalkI
- dmxR[1] = sigR1[*s*] · xTalkD + *sign* · sigR0[*s*] · xTalkI
- dmxI[0] = sigI0[*s*] · xTalkD + *sign* · sigI1[*s*] · xTalkD
- dmxI[1] = sigI1[*s*] · xTalkD + *sign* · sigI0[*s*] · xTalkD

taking $c = 0, 1$ as a channel iterator. If $|\text{dmxR}[c]| \geq \text{absR}[0] + \text{absR}[1]$, set *sign* = *chanCorrSign*. Otherwise, set *sign* = –*chanCorrSign* and if $\text{absR}[c] \cdot \text{xTalkD} < \text{absR}[1-c] \cdot \text{xTalkI}$, recalculate dmxR and dmxI to avoid signal cancellation:

- $\text{dmxR}[c] = \text{sigR}_{1-c}[s] \cdot \text{xTalkI} + sign \cdot \text{sigR}_{c}[s] \cdot \text{xTalkD}$
- $\text{dmxI}[c] = \text{sigI}_{1-c}[s] \cdot \text{xTalkI} + sign \cdot \text{sigI}_{c}[s] \cdot \text{xTalkD}$

To conclude the application of the phase alignment, calculate, for subband index *s* and each channel *c*, correction factor

$f[c] = \sqrt{(n / \max(\text{EPSILON}, d))}$ with EPSILON = 1 being a lower limit to avoid divisions by zero,

where numerator *n* is the power spectral value of MCLT sample sig*c*[*s*] (i.e., squared real part + squared imaginary part) and denominator *d* is the power spectral value of the channel's downmix instance (i.e., $\text{dmxR}[c]^2 + \text{dmxI}[c]^2$). Then, set:

- sigR0[*s*] = dmxR[0] · *f*[0]
- sigR1[*s*] = dmxR[1] · *f*[1]
- sigI0[*s*] = dmxI[0] · *f*[0]
- sigI1[*s*] = dmxI[1] · *f*[1]

These pre-processed MCLT samples for both channels, separated as previously into a real-valued MDCT-like part and an imaginary-valued MDST-like part, can now be subjected to temporal noise shaping encoding, as described in clause 5.3.3.3.5.

5.3.3.3.5 Temporal noise shaping (TNS)

5.3.3.3.5.1 General overview of algorithm

Temporal Noise Shaping (TNS) is used to control the temporal shape of the quantization noise within each window of the transform. For each channel, if TNS is set to be active, up to two filters per MDCT-spectrum will be applied. TNS is always calculated on a per sub-window basis, so in case of a 4 TCX5 window sequence the filtering steps must be applied once for each of the 4 sub-windows. The algorithm for the TNS detection, filtering and coding of the parameters are described in detail in clause 5.3.3.2.2 of [3].

For the stereo audio input case there are some stereo specific considerations that apply. The first aspect concerns the case where the TNS algorithm applies similar filters to both channels. In that case, a common filter is applied in order to save on bitrate since only one set of parameters would need to be encoded. Furthermore, it has been found that in certain cases it is favorable, in terms of audio quality performance, to apply the temporal noise shaping prior to the spectral noise shaping (SNS) and for other cases it is better to apply it afterwards. The stereo-specific aspects of the TNS analysis are described in detail in the following clause 5.3.3.3.5.2.

5.3.3.3.5.2 TNS for stereo

At first the TNS analysis is applied to each channel individually to determine the filter configuration per channel. The TNS filtering can be synchronized between the channels only when the spectral resolution is the same, that means that the transform type must be the same for both channels. Specifically for the short blocks, the transform type for each subframe should be also in sync. If that is not the case than the TNS decision is applied for each channel separately as done in EVS and described in clause 5.3.3.2.2 of [3].

If the transform types are in sync for both channels than the similarity of the filters are examined and if they are similar than the TNS decision is forced to be the same for both channels. Depending on the similarity it may be that the same filters are used to save on TNS parameters that need to be transmitted to the decoder.

At first the mean prediction gain $\tilde{\eta}_{tns}$ is calculated as in:

$$\tilde{\eta}_{tns} = \frac{\eta_{tns_1} + \eta_{tns_2}}{2}$$

where η_{tns_1} and η_{tns_2} are the TNS prediction gains for the first and the second channel respectively.

For the mid-high bitrates of the MDCT-based stereo, namely 48 and 64 kbps the prediction gains n_{tns_1} and n_{tns_2} are set to the mean prediction gain $\tilde{\eta}_{tns}$, if the individual gains are larger than a minimum threshold $n_{tns_1} > \eta_{min}$ and $n_{tns_2} > \eta_{min}$ and if the number of TNS filters detected for both channels are the same.

To determine whether the TNS decision can be synced, given that the number of filters detected are the same, the difference of the prediction gains to a specific threshold are examined as follows:

$$\left|\eta_{\mathrm{tns}_1} - \eta_{\mathrm{tns}_2}\right| < 0.04 \cdot \tilde{\eta}_{tns}$$

Then if one of the conditions described in clause 5.3.3.2.2.1 of [3] for one of the channels is met then both channels are synced to enable TNS filtering, otherwise they are both kept to the same state as for the previous frame - keep it turned off or on.

Furthermore, for the mid-high bitrates 48 and 64 kbps where TNS is detected to be applied, the filters applied are synchronized to save on bitrate for transmitting the TNS parameters. This is described in the steps below:

- Determine the maximum absolute difference between the respective filter coefficients of the two channels.
- If the maximum absolute difference equals 1, then the TNS coefficients are sufficiently similar to equalize the two filters.
- The common filter coefficients to be used for each tap by both channels are the coefficients with the smallest absolute value between the two channels.

If the above conditions do not apply, then the TNS detection and decision is applied individually for each channel as described in clause 5.3.3.2.2.1 of [3].

Additionally, if the maximum prediction gain of both channels is lower than a threshold of 3 and the MDCT transform type is either TCX20 or TCX10, then the steps described above are repeated on the frequency-domain signal after the spectral noise shaping (whitened signal). In this case, all filters are reset to zero and TNS is turned off and is not applied on the MDCT spectrum of the original input signal.

Finally, after the procedure of determining whether TNS will be applied (before or after the whitening of the spectrum) and the filters to be applied, the actual TNS filtering occurs either on the MDCT spectrum of the original audio input or the whitened MDCT spectrum after the SNS. The TNS filtering is described in detail in clause 5.3.3.2.2.2 of [3].

5.3.3.3.6 Spectral Noise Shaping (SNS)

5.3.3.3.6.1 General

Spectral Noise Shaping (SNS) applies a set of scale parameters as factors to the signal after being transformed to a spectral representation (MDCT transform, see clause 5.3.3.3.3). These scale parameters shape the quantization noise introduced in the frequency domain by the spectral quantization. The noise shaping is performed in such a way that the quantization noise is minimally perceived by the human ear, maximizing the perceptual quality of the decoded output.

The SNS encoder consists of a scale parameter calculator, a downsampler and a scale parameter encoder. After encoding the SNS scale parameters, a signal processor interpolates the downsampled scale parameters to obtain scale factors and process the signal spectrum by applying these factors to both channels of the signal in the frequency domain.

5.3.3.3.6.2 SNS Frequency bands

The SNS scale parameters are calculated and applied in 64 frequency bands which approximately follow the Bark scale. These psychoacoustic bands are used in every frame except in transition frames when the previous frame was coded using ACELP. For such frames, equally sized bands are used and the length of each band is calculated by dividing the length of the spectrum by the number of bands (64). The bands are defined in Table 5.3-12 with *b* being the band index, L_b the length of the band in frequency bins, $i_{low,b}$ indicating the first index of the band and $i_{high,b}$ indicating the last index of the band.

Table 5.3-12: Definition of frequency bands for SNS

	Core Fs = 16 kHz						Core Fs = 25.6 kHz						Core Fs = 32 kHz					
	TCX20			TCX10			TCX20			TCX10			TCX20			TCX10		
b	L_b	$I_{low,b}$	$I_{high,b}$	L_b	$I_{low,b}$	$I_{high,b}$	L_b	$I_{low,b}$	$I_{high,b}$	L_b	$I_{low,b}$	$I_{high,b}$	L_b	$I_{low,b}$	$I_{high,b}$	L_b	$I_{low,b}$	$I_{high,b}$
0	2	0	1	1	0	0	2	0	1	1	0	0	2	0	1	1	0	0
1	2	2	3	1	1	1	2	2	3	1	1	1	2	2	3	1	1	1
2	2	4	5	1	2	2	2	4	5	1	2	2	2	4	5	1	2	2
3	2	6	7	1	3	3	2	6	7	1	3	3	2	6	7	1	3	3
4	2	8	9	1	4	4	2	8	9	1	4	4	2	8	9	1	4	4
5	2	10	11	1	5	5	2	10	11	1	5	5	2	10	11	1	5	5
6	2	12	13	1	6	6	2	12	13	1	6	6	2	12	13	1	6	6
7	2	14	15	1	7	7	2	14	15	1	7	7	2	14	15	1	7	7
8	2	16	17	1	8	8	2	16	17	1	8	8	2	16	17	1	8	8
9	2	18	19	1	9	9	2	18	19	1	9	9	2	18	19	1	9	9
10	2	20	21	1	10	10	2	20	21	2	10	11	2	20	21	2	10	11
11	2	22	23	1	11	11	4	22	25	2	12	13	4	22	25	2	12	13
12	2	24	25	1	12	12	4	26	29	2	14	15	4	26	29	2	14	15
13	2	26	27	1	13	13	4	30	33	2	16	17	4	30	33	2	16	17
14	2	28	29	1	14	14	4	34	37	2	18	19	4	34	37	2	18	19
15	2	30	31	1	15	15	4	38	41	2	20	21	4	38	41	2	20	21
16	4	32	35	2	16	17	4	42	45	2	22	23	4	42	45	2	22	23
17	4	36	39	2	18	19	4	46	49	2	24	25	4	46	49	2	24	25
18	4	40	43	2	20	21	4	50	53	3	26	28	4	50	53	3	26	28
19	4	44	47	2	22	23	4	54	57	3	29	31	4	54	57	3	29	31
20	4	48	51	2	24	25	4	58	61	3	32	34	4	58	61	3	32	34
21	4	52	55	2	26	27	4	62	65	3	35	37	4	62	65	3	35	37
22	4	56	59	2	28	29	4	66	69	3	38	40	4	66	69	3	38	40
23	4	60	63	2	30	31	4	70	73	3	41	43	4	70	73	3	41	43
24	4	64	67	2	32	33	4	74	77	3	44	46	4	74	77	4	44	47
25	4	68	71	2	34	35	4	78	81	3	47	49	4	78	81	4	48	51
26	4	72	75	2	36	37	4	82	85	4	50	53	4	82	85	4	52	55
27	4	76	79	2	38	39	6	86	91	4	54	57	6	86	91	4	56	59
28	4	80	83	2	40	41	6	92	97	4	58	61	6	92	97	4	60	63
29	4	84	87	2	42	43	6	98	103	4	62	65	6	98	103	4	64	67
30	6	88	93	2	44	45	6	104	109	4	66	69	6	104	109	5	68	72
31	6	94	99	2	46	47	6	110	115	4	70	73	6	110	115	5	73	77
32	6	100	105	3	48	50	6	116	121	4	74	77	6	116	121	5	78	82
33	6	106	111	3	51	53	6	122	127	4	78	81	8	122	129	5	83	87
34	6	112	117	3	54	56	6	128	133	4	82	85	8	130	137	5	88	92
35	6	118	123	3	57	59	8	134	141	4	86	89	8	138	145	5	93	97
36	6	124	129	3	60	62	8	142	149	5	90	94	8	146	153	6	98	103
37	6	130	135	3	63	65	8	150	157	5	95	99	8	154	161	6	104	109
38	6	136	141	3	66	68	8	158	165	5	100	104	10	162	171	6	110	115
39	6	142	147	3	69	71	8	166	173	5	105	109	10	172	181	6	116	121
40	6	148	153	3	72	74	8	174	181	5	110	114	10	182	191	6	122	127
41	6	154	159	3	75	77	8	182	189	5	115	119	10	192	201	6	128	133
42	6	160	165	3	78	80	8	190	197	5	120	124	12	202	213	7	134	140
43	6	166	171	3	81	83	10	198	207	5	125	129	12	214	225	7	141	147
44	6	172	177	3	84	86	12	208	219	5	130	134	12	226	237	7	148	154
45	6	178	183	3	87	89	12	220	231	5	135	139	12	238	249	7	155	161
46	6	184	189	3	90	92	12	232	243	6	140	145	14	250	263	7	162	168
47	6	190	195	3	93	95	12	244	255	6	146	151	14	264	277	7	169	175
48	6	196	201	4	96	99	12	256	267	6	152	157	14	278	291	8	176	183
49	6	202	207	4	100	103	12	268	279	6	158	163	16	292	307	8	184	191
50	8	208	215	4	104	107	14	280	293	6	164	169	18	308	325	8	192	199
51	8	216	223	4	108	111	14	294	307	6	170	175	18	326	343	8	200	207
52	8	224	231	4	112	115	14	308	321	6	176	181	18	344	361	8	208	215
53	8	232	239	4	116	119	14	322	335	6	182	187	20	362	381	8	216	223
54	8	240	247	4	120	123	14	336	349	6	188	193	22	382	403	9	224	232
55	8	248	255	4	124	127	16	350	365	6	194	199	22	404	425	9	233	241
56	8	256	263	4	128	131	16	366	381	7	200	206	22	426	447	9	242	250
57	8	264	271	4	132	135	18	382	399	7	207	213	24	448	471	9	251	259
58	8	272	279	4	136	139	18	400	417	7	214	220	26	472	497	10	260	269
59	8	280	287	4	140	143	18	418	435	7	221	227	26	498	523	10	270	279
60	8	288	295	4	144	147	18	436	453	7	228	234	26	524	549	10	280	289
61	8	296	303	4	148	151	18	454	471	7	235	241	28	550	577	10	290	299

62	8	304	311	4	152	155	20	472	491	7	242	248	30	578	607	10	300	309
63	8	312	319	4	156	159	20	492	511	7	249	255	32	608	639	10	310	319

5.3.3.3.6.3 SNS scale factor calculation

The scale parameter calculator first calculates the energy in each of the non-uniform bands as given in Table 5.3-12. The energy is calculated as the squared amplitude of the complex signal spectrum. To calculate the complex amplitude of the signal spectrum, both the MDCT and the MDST spectral values are needed. If TNS has already been applied on the MDCT spectrum, the signal energy is only estimated from the MDCT spectrum as the TNS processing is not applied on the MDST spectrum. The energy per band is normalized by dividing it by the respective band length.

$$E(b) = \begin{cases} \sum_{i=i_{low,b}}^{i_{high,b}} \frac{(X^{MDCT}(i))^2}{L_b}, if\ TNS\ has\ been\ applied \\ \sum_{i=i_{low,b}}^{i_{high,b}} \frac{(X^{MDCT}(i))^2 + (X^{MDST}(i))^2}{L_b}, otherwise \end{cases}, b = 0, \dots, 63. \quad (5.3\text{-}249)$$

The energy per band is smoothed according to

$$E_s(b) = \begin{cases} 0.75\ E(0) + 0.25\ E(1), if\ b = 0 \\ 0.75\ E(63) + 0.25\ E(62), if\ b = 63 \\ 0.25\ E(b-1) + 0.5\ E(b) + 0.25\ E(b+1), otherwise \end{cases}, b = 0, \dots, 63. \quad (5.3\text{-}250)$$

A pre-emphasis operation is performed on the smoothed energy values so that low frequency amplitudes are emphasized with respect to high frequency amplitudes:

$$E_p(b) = E_s(b) \cdot 10^{\frac{b \cdot g_{tilt}}{(63/10)}}, b = 0, \dots, 63 \quad (5.3\text{-}251)$$

where g_{tilt} controls the pre-emphasis tilt and depends on the core sampling rate as specified in Table 5.3-13.

Table 5.3-13: Pre-emphasis tilt values for SNS parameter calculation

	Fs = 16 kHz	Fs = 25.6 kHz	Fs = 32 kHz
g_{tilt}	19	22	23.5

Next, a noise floor at -40 dB is added to the pre-emphasized energy values:

$$E_n(b) = max\left(E_p(b), nf\right), b = 0,1, \dots, 63, where\ nf = max\left(\frac{\sum_{b=0}^{63} E_p(b)}{63} \cdot 10^{-4}, 2^{-32}\right). \quad (5.3\text{-}252)$$

The energy values are then transformed into a logarithmic domain according to:

$$E_L(b) = \frac{\log_2(E_n(b))}{2}. \quad (5.3\text{-}253)$$

After this first set of scale parameters in the logarithmic domain is computed, the downsampler calculates a second set of scale parameters by downsampling the first set.

The vector $E_L(b)$ is downsampled by a factor of 4 according to:

$$E_4(b) = \begin{cases} w(0)E_L(0) + \sum_{k=1}^{5} w(k)E_L(4b+k-1), if\ b = 0 \\ \sum_{k=0}^{4} w(k)E_L(4b+k-1) + w(5)E_L(63), if\ b = 15,\ b = 0, \dots, 15 \\ \sum_{k=0}^{5} w(k)E_L(4b+k-1), otherwise \end{cases} \quad (5.3\text{-}254)$$

with

$$w(k) = \left\{\frac{1}{12}, \frac{2}{12}, \frac{3}{12}, \frac{3}{12}, \frac{2}{12}, \frac{1}{12}\right\} \quad (5.3\text{-}255)$$

This step applies a low pass filter before decimation which is equivalent to a weighted average operation performed on the values of the parameter vector weighting closer scale parameters stronger than more distant ones. Subsequently, a mean removal is performed so that the second set of scale parameters calculated by the downsampler is mean free and the parameter vector is scaled by a constant:

$$scf(n) = 0.85\left(E_4(n) - \frac{\sum_{b=0}^{15} E_4(b)}{16}\right), n = 0, \ldots, 15 \quad (5.3\text{-}256)$$

The resulting set of scale parameters is then passed to the scale parameter encoder that quantizes the set using one of two different vector quantizers depending on the operating point.

5.3.3.3.6.4 SNS scale factor quantization

5.3.3.3.6.4.1 General

The quantization mechanism for the SNS scale parameters depends on the codec bitrate and the sampling rate at which the core-coder operates. For mid bitrates at a core sampling rate above 16 kHz, a multi-stage stochastic VQ with a variable number of stages is used. At higher bitrates, a 2-stage vector quantizer structure is used consisting of a stochastic split VQ and an Algebraic Vector Quantizer (AVQ). Both SNS quantization modes employ a signal-adaptive joint coding mechanism. Based on a similarity measure between the channels, either joint (mid/side) or separate coding of the scale parameters is used. In the joint case, the side representation of the SNS scale parameters is treated differently than the other representations by quantizing it with a reduced number of stages compared to the number of stages used for mid, left and right representations of the SNS scale parameters. Also, dedicated signalling is used to communicate to the decoder if the side representation is very close to a zero vector, in which case it is encoded in the bitstream by just the single signalling bit.

Selection of the quantizer type is done according to Table 5.3-14 below.

Table 5.3-14: Assignment of SNS VQ types to operating points

Core F_s	≤ 64 kbps	> 80 kbps
16 kHz	Split/AVQ	Split/AVQ
25.6 kHz	MSVQ	Split/AVQ
32 kHz	N/A	Split/AVQ

5.3.3.3.6.4.2 Determining the encoding mode

Using the joint encoding mode is only possible if both channels of a frame use the same transformation block length. If the transformation block length differs between the channels, the separate encoding mode is used. If both channels use short blocks, joint encoding is only possible at operating points where the MSVQ is used.

To determine the encoding mode, a similarity measure E is calculated as the squared difference between the SNS scale parameters for the left and the SNS scale parameters of the right channel:

$$E = \sum_{i=0}^{15}\left(SNS_{left}(i) - SNS_{right}(i)\right)^2. \quad (5.3\text{-}257)$$

E is then compared to a threshold value $\varepsilon = 12$. If E is bigger than ε, the separate encoding mode is chosen, otherwise the joint encoding mode is chosen. In the joint encoding mode, the SNS scale parameters are transformed into a mid/side representation before quantization:

$$SNS_{mid}(i) = \frac{SNS_{left}(i) + SNS_{right}(i)}{\gamma_{enc}} \quad (5.3\text{-}258)$$

$$SNS_{side}(i) = \frac{SNS_{left}(i) - SNS_{right}(i)}{\gamma_{enc}}, \quad (5.3\text{-}259)$$

where $\gamma_{enc} = 1$ when the Split/AVQ quantizer is used and $\gamma_{enc} = 2$ when the MSVQ is used.

In operating points that use the MSVQ, the process is done once for each subframe (1 for long blocks, 2 for short blocks). Thus, it can be the case that the encoding mode is different in each subframe and the SNS scale parameters are encoded in different representation between the subframes.

The encoding mode is transmitted in the bitstream using a single bit (if the MSVQ is used, one bit per subframe is used). This bit is only written if both channels use the same transform block length in the current frame. If the block lengths differ between channels, this is used as an implicit signaling for the separate encoding mode as using the joint encoding mode is not possible in that case.

5.3.3.3.6.4.3 2-Stage Split/AVQ Quantizer

The 2-stage Split/AVQ quantizer is used at bitrates above 64 kbps and in any bitrate if the core sampling rate is 16 kHz. It consists of two stages with separate vector quantizers. The first stage vector quantizer is a stochastic split VQ which performs a fixed rate quantization. After the first stage, a residual vector is calculated, and the residual vector is further quantized using an Algebraic Vector Quantizer (AVQ) in the second stage which performs a variable rate quantization. In the joint encoding mode, the first stage is skipped for the side SNS parameters, and the residual is set to zero which results in SNS_{side} being directly quantized by the second stage AVQ only, as is depicted in Figure 5.3-39.

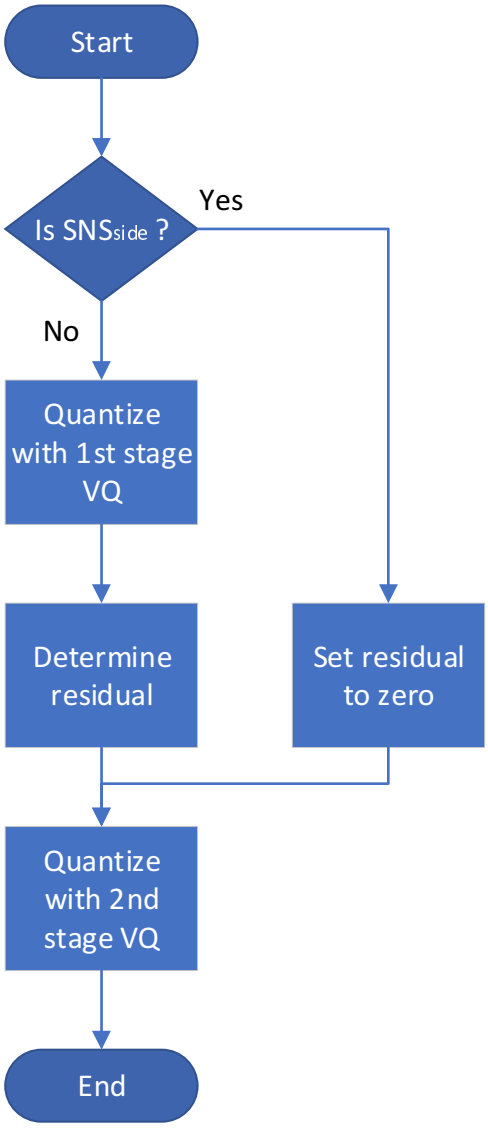


Figure 5.3-39: SNS parameter VQ encoding for long blocks in both channels

The 1st stage split VQ uses off-line trained stochastic codebooks of dimension 8 with 32 code vectors per codebook. Codebook selection depends on the split number (first split, second split) and the transform block size, see [7] for the definition of the codebooks. To reduce the overall size of the needed codebooks, dependency on the core sampling rate is removed by normalizing the input vectors to the first stage VQ. Normalization is done by subtracting a mean value from each scale parameter in the target vector:

$$\overline{SNS}(i) = SNS(i) - \mu(i)\ \kappa_{means},\ i \in \{0, \dots, 15\}. \quad (5.3\text{-}260)$$

As the first stage VQ is skipped for side SNS scale parameter quantization, SNS can be one of SNS_{left}, SNS_{right} or SNS_{mid}. The values for $\mu(i)$ were computed as the means of each SNS scale parameter in the training data used for creating the codebooks for different core sampling rates and are denote in Table 5.3-15 below. The values are stored in a 16-bit Q format with 14 bits used for the fractional part. $\kappa_{means} = \frac{1}{2^{14}}$ is a conversion constant to convert the values back from the fixed-point number format.

Table 5.3-15: SNS parameter mean normalization values for 1st stage split VQ

Core sampling rate	16 kHz	16 kHz	25.6 kHz	25.6 kHz	32 kHz	32 kHz
block size	**Long**	**Short**	**Long**	**Short**	**Long**	**short**
μ(0)	14210	12018	14973	15560	15041	16510
μ(1)	19017	15915	20323	19489	20603	20660
μ(2)	14362	11089	16461	14623	16969	16025
μ(3)	9309	6015	9554	5595	10289	7224
μ(4)	5385	847	4017	2084	4973	3921
μ(5)	2674	-705	3103	1699	4283	3868
μ(6)	1055	-539	1602	775	3003	2623
μ(7)	-211	-1548	1694	-1312	3316	742
μ(8)	-1407	-893	-221	-2195	1684	-1316
μ(9)	-3059	-2163	-1401	-6101	-259	-6269
μ(10)	-4393	-1806	-6817	-9078	-6614	-8284
μ(11)	-8597	-4189	-10071	-9465	-9535	-7288
μ(12)	-11180	-7017	-11503	-7825	-10363	-6380
μ(13)	-11756	-8670	-11805	-6603	-11834	-8410
μ(14)	-12131	-8874	-13158	-7281	-16625	-13351
μ(15)	-13281	-9480	-16749	-9960	-24930	-20277

For both splits, a full search of the respective codebook is carried out. The code vector that minimizes the Mean Squared Difference to the target vector is chosen, i.e the codebook index for split $n \in [0,\ 1]$ is chosen according to:

$$ind_n = argmin_{i=0,\dots,31}\left\{\sum_{j=0}^{7}\left(\overline{SNS}(j+8\mathrm{n}) - V_{n,m}(i)(j)\kappa_{codebook}\right)\right\}, for\ j = 0,\dots,31 \qquad (5.3\text{-}261)$$

where m is 0 for long transform blocks and 1 for short transform blocks and $V_{n,m}(i)$ denotes the i-th code vector for split n and the respective transform length. The codebooks are stored in 16-bit Q format using 12 bits for the fractional part with $\kappa_{codebook} = \frac{1}{2^{12}}$ being the conversion constant. The found two optimal indices are represented as two 5 bit values and get combined into a single 10-bit index as $ind_{SNS,1} = ind_0 + 2^5 \times ind_1$.

The output of the 1st stage VQ is then an intermediate quantized vector according to:

$$\widehat{SNS}_{VQ,1}(i) = \begin{cases} V_{0,m}(ind_0)\ (i)\kappa_{codebook} + \mu(i)\ \kappa_{means}, for\ i \in \{0,\dots,7\} \\ V_{1,m}(ind_1)(i-8)\kappa_{codebook} + \mu(i)\ \kappa_{means}, for\ i \in \{8,\dots,15\} \end{cases}. \qquad (5.3\text{-}262)$$

For side SNS scale parameter quantization, the first stage is skipped and $\widehat{SNS}_{VQ,1}(i)$ is set to zero instead.

The residual vector is obtained by calculating the difference between each SNS scale parameter and the corresponding quantized parameter in the intermediate quantized vector and weighting the difference with the factor $1/_{0.4}$.

$$SNS_{res}(i) = \frac{SNS(i) - \widehat{SNS}_{VQ,1}(i)}{0.4}. \qquad (5.3\text{-}263)$$

The residual vector is further quantized by the second stage VQ which consists of the AVQ as described in clause 5.2.3.1.6.9 of [3]. The output of the second stage consists of a base codebook index and a Voronoi extension index.

After the quantization indices have been calculated, the quantized SNS scale parameter vector is calculated as

$$\widehat{SNS}(i) = \widehat{SNS}_{VQ,1}(i) + 0.4\ \widehat{SNS}_{VQ,2}(i), \qquad (5.3\text{-}264)$$

where $\widehat{SNS}_{VQ,2}$ denotes the quantized residual vector as calculated by the AVQ.

For scale parameters in side representation, a *zero_side* flag is set according to

$$zero\_side = \begin{cases} 1, if\ \sum_{i=0}^{15}\widehat{SNS}_{side}(i) = 0 \\ 0, otherwise \end{cases}. \qquad (5.3\text{-}265)$$

The flag is encoded in the bitstream using a single bit. If the *zero_side* flag is 1, the side SNS scale parameter vector is not encoded in the bitstream.

If at least one of the channels uses short transform blocks, no joint coding of the SNS scale parameters with the split/AVQ quantizer is supported. Both channel's SNS scale parameters are then separately encoded using both stages.

At 48 kbps only, a special low-bitrate mode can be used additionally. The low-bitrate mode is activated globally for a frame (not on a per-channel basis) and is signaled by a single low-bitrate-mode indicator bit which is written only at 48 kbps if at least one channel uses short blocks. The indicator bit is set to one if the speech/music classifier (clause 5.3.2.5) classified the first channel as speech, otherwise it is set to zero. If it is one, only the first stage split VQ is used to quantize all SNS scale parameter vectors in this frame and the second stage VQ is skipped.

If the low-bitrate mode is not used (i.e. at 48 kbps if the frame is not classified as speech and at all other bitrates – regardless of signal classification), an inter-subframe coding mechanism is used for channels with short transform block lengths. For short transform block lengths, two sets of SNS scale parameters need to be quantized and encoded – one for each subframe. The first set is always encoded as described before, using both VQ stages. For the second set, one of two encoding strategies is chosen, based on which one needs less bits to encode the second parameter vector. The first strategy is simply to encode the second subframe's SNS vector with both stages of the VQ as done for the first subframe's vector. The number of bits needed for that is $nbits_1 = 10 + nbits_{AVQ,1}$, where 10 is the fixed number of bits used by stage 1 and $nbits_{AVQ,1}$is the number of bits needed by stage 2. The second strategy employs the 2nd stage only, using the difference between the quantized first subframe's vector and the unquantized second subframe's vector as the residual to be quantized. The number of bits needed for this strategy is only the number of bits needed by the AVQ to quantize this residual, denoted as $nbits_{AVQ,2}$. If $nbits_{AVQ,2} < nbits_1$, the second encoding strategy is chosen and only the AVQ indices for quantizing the difference vector are transmitted. Otherwise, 1st and 2nd stage indices are transmitted for the second subframe, too. This encoding process is depicted in Figure 5.3-40. The encoding strategy is signaled in the bitstream using a single bit.

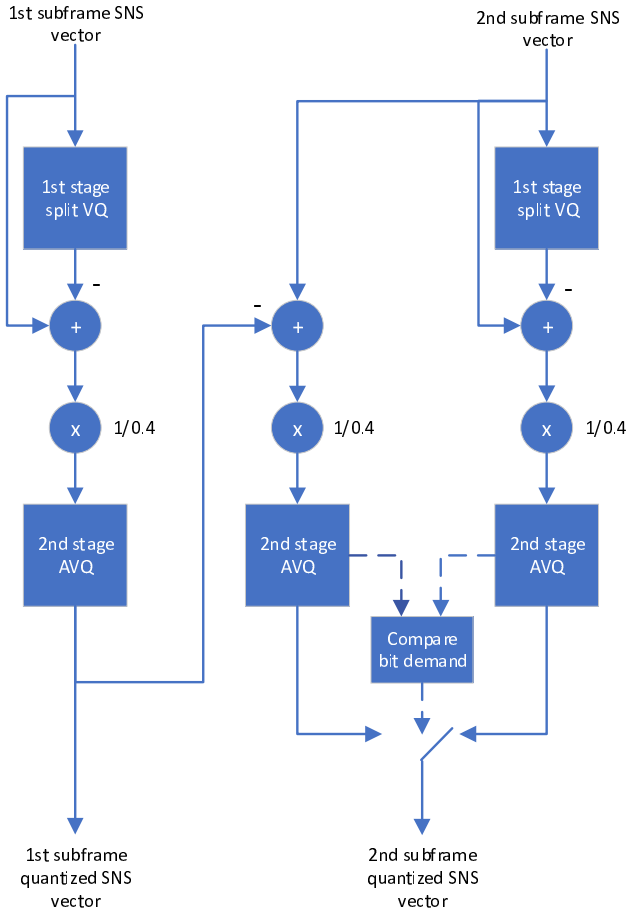


Figure 5.3-40: Inter-subframe encoding of SNS scale parameter vectors

5.3.3.3.6.4.4 MSVQ

At bitrates 48 and 64 kbps if the core sampling rate is higher than 16 kHz, the SNS scale parameters are quantized using a stochastic MSVQ with offline-trained codebooks. Codebook selection differentiates between the kind of representation the respective SNS parameter vector is in. SNS_{left}, $SNS_{right}$and SNS_{mid}share codebooks while there is a separate codebook used for SNS_{side}vectors. In between these two groups, there are separate codebooks used depending

on the transform block length of the current frame (long or short). Therefore, there are four codebooks in total. The number of stages and numbers of bits per stage for each codebook are shown in Table 5.3-16.

Table 5.3-16: Codebook parameters for SNS MSVQ

Codebook	Used for		Number of stages	Bits per stage
V_A	SNS_{left}, SNS_{right} , SNS_{mid}	Long blocks	4	7, 6, 5, 5
V_B	SNS_{left}, SNS_{right} , SNS_{mid}	Short blocks	3	7, 5, 3
V_C	SNS_{side}	Long blocks	2	5, 5
V_D	SNS_{side}	Short blocks	2	5, 3

Encoding of an SNS scale parameter vector is done as described in clause 5.6.3.5 of [3] but using the respective codebooks and parameters. For scale parameters in side representation, a reduced number of stages is used compared to what is used for the other representations and the quantization is skipped completely if the side representation is determined to be near zero. This is done by examining the similarity measure E as defined in Equation (5.3-257) and setting a *zero_side* flag according to

$$zero\_side = \begin{cases} 1, if\ E < 1 \\ 0, otherwise \end{cases}. \quad (5.3\text{-}266)$$

The flag is encoded in the bitstream using a single bit. If *zero_side* is 1, the values in the side SNS scale parameter vector are set to zero and quantization is skipped. Otherwise, the parameter vector is quantized using the respective codebook and MSVQ parameters as given in Table 5.3-16.

5.3.3.3.6.4.5 Restoring left/right representation of jointly coded SNS scale parameters

If the joint encoding mode was chosen, the SNS scale parameters were encoded in a mid/side representation and need to be decoded back to a left/right representation so they can be used to shape the channel signals. The restored left/right SNS scale parameter vectors are calculated according to:

$$\widehat{SNS}_{left}(i) = \begin{cases} \widehat{SNS}_{mid}(i), if\ zero\_side\ is\ 1 \\ \widehat{SNS}_{mid}(i) + \frac{\widehat{SNS}_{side}(i)}{\gamma_{dec}}, otherwise \end{cases} \quad (5.3\text{-}267)$$

$$\widehat{SNS}_{right}(i) = \begin{cases} \widehat{SNS}_{mid}(i), if\ zero\_side\ is\ 1 \\ \widehat{SNS}_{mid}(i) - \frac{\widehat{SNS}_{side}(i)}{\gamma_{dec}}, otherwise \end{cases} \quad (5.3\text{-}268)$$

where $\gamma_{dec} = 2$ if the Split/AVQ quantizer was used and $\gamma_{dec} = 1$ if the MSVQ was used.

5.3.3.3.6.5 Spectral Shaping

5.3.3.3.6.5.1 General

After quantizing the SNS scale parameters, the MDCT spectrum is scaled using scale factors derived from the quantized scale parameters. The scale factors are obtained by interpolation of the scale parameters. Shaping of the MDCT spectrum is done separately for both channels.

5.3.3.3.6.5.2 SNS scale factor interpolation

The 16 quantized SNS scale parameters are interpolated to obtain 64 scale factors according to

$$\begin{aligned} scf(0) &= \widehat{SNS}(0) \\ scf(1) &= \widehat{SNS}(0) \\ scf(4n+2) &= \widehat{SNS}(n) + \tfrac{1}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \ldots, 14 \\ scf(4n+3) &= \widehat{SNS}(n) + \tfrac{3}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \ldots, 14 \\ scf(4n+4) &= \widehat{SNS}(n) + \tfrac{5}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \ldots, 14 \end{aligned} \quad (5.3\text{-}269)$$

$$scf(4n+5) = \widehat{SNS}(n) + \frac{7}{8}\left(\widehat{SNS}(n+1) - \widehat{SNS}(n)\right), n = 0, \ldots, 14$$

$$scf(62) = \widehat{SNS}(15) + \frac{1}{8}\left(\widehat{SNS}(15) - \widehat{SNS}(14)\right)$$

$$scf(63) = \widehat{SNS}(15) + \frac{3}{8}\left(\widehat{SNS}(15) - \widehat{SNS}(14)\right).$$

The scale factors are then transformed back into the linear domain according to

$$g_{SNS}(b) = 2^{-scf(b)}, b = 0, \ldots, 63. \quad (5.3\text{-}270)$$

5.3.3.3.6.5.3 Scaling the MDCT spectrum

The interpolated scale factors are applied to the MDCT spectrum in the same bands as used for calculating the SNS scale parameters as described in clause 5.3.3.3.6.2 according to the following pseudocode:

```
For b = 0 to 63 do
    For k = I_low(b) to I_high(b) do
        X_s(k) = X(k)*g_SNS(b)
```

5.3.3.4 Stereo processing

5.3.3.4.1 General

The stereo encoding processing is applied band-wise on the frequency-domain noise-shaped signals. First, a global ILD normalization value is determined, based on the energy values of the first channel and the second channel inputs. Depending on the normalization value, at least one of the two channels is normalized to achieve similar levels for both channels. Then, the encoding of the normalized audio signals follows, by determining for each spectral band whether the output spectral band remains the same for the first and second channel (dual-mono representation) or whether the output spectral band of the first channels is a mid signal of the first and second channel and the output spectral band of the second channel is a side signal of the first and second channel (mid-side representation), depending on the spectral band-wise criteria described in detail in clause 5.3.3.4.5. For many signals significant coding gain is achieved by having some bands in L/R (dual-mono) and some bands in M/S (mid-side) representation. Finally, for each spectral band the decided encoding is performed outputting the encoded audio signal for each channel.

5.3.3.4.2 Stereo spectral bands

Most of the stereo processing is done in spectral bands configured specifically for the stereo processing. The stereo spectral bands are configured following mainly the ERB scale with some modifications to accommodate the long frames for the TCX20 coding mode and the short frames for the TCX10 coding mode and to match the IGF spectral bands for the bitrates where IGF is active.

In Table 5.3-17 the grouping of frequency bins to spectral bands and the number of bands that are assigned to each bandwidth for TCX20 and TCX10 coding mode is shown. If the total spectral offset exceeds the maximum available bandwidth length, then the last band is corrected to the maximum bandwidth length.

$$b_{offset[b_{last}]} = \min\left(b_{offset}, L_{frameTCX}\right) \quad (5.3\text{-}271)$$

Table 5.3-17: Core stereo spectral bands per coding mode

	TCX20			TCX10		
Band	b_{offset}	N_{bins}	**Bwidth**	b_{offset}	N_{bins}	**Bwidth**
0	0	0	WB	0	0	WB
1	4	4	WB	4	4	WB
2	8	4	WB	8	4	WB
3	12	4	WB	12	4	WB
4	16	4	WB	16	4	WB
5	20	4	WB	20	4	WB
6	24	4	WB	28	8	WB
7	28	4	WB	36	8	WB
8	32	4	WB	44	8	WB
9	36	4	WB	52	8	WB
10	40	4	WB	64	12	WB
11	48	8	WB	76	12	WB
12	56	8	WB	88	12	WB
13	64	8	WB	100	12	WB
14	72	8	WB	112	12	WB
15	80	8	WB	124	12	WB
16	88	8	WB	136	12	WB
17	96	8	WB	148	12	WB
18	108	8	WB	160	12	WB
19	120	12	WB	172	12	SWB
20	132	12	WB	184	12	SWB
21	144	12	WB	196	12	SWB
22	164	12	WB	208	12	SWB
23	184	20	WB	220	12	SWB
24	204	20	WB	232	12	SWB
25	226	20	WB	244	12	SWB
26	248	22	WB	256	12	SWB
27	270	22	WB	268	12	SWB
28	292	22	WB	288	20	SWB
29	314	22	WB	320	32	SWB
30	336	22	SWB	352	32	FB
31	358	22	SWB	384	32	FB
32	360	22	SWB	432	48	FB
33	392	22	SWB	512	80	FB
34	424	22	SWB			
35	446	22	SWB			
36	468	22	SWB			
37	490	22	SWB			
38	512	22	SWB			
39	534	22	SWB			
40	576	22	SWB			
41	640	42	SWB			
42	704	64	FB			
43	800	96	FB			
44	960	160	FB			

The aforementioned stereo bands are applied for the configurations shown in Table 5.3-18.

Table 5.3-18: Configurations where the core stereo bands are used

core sampling rate [kHz]	bitrates [kbps]
48	all
32	up to 96
25.6	up to 96
16	up to 96

For all other configurations i.e. 16 – 32 kHz core internal sampling rate and bitrate > 96 kbps the spectral bands defined in Table 5.3-12 are used.

For the bitrates where IGF is enabled the cross-over bands are adjusted to represent the cross-over frequency between the core encoding and the IGF encoding. Also, the bands over the IGF cross-over frequency are adjusted to be identical to the IGF spectral bands, defined depending on the configuration, as depicted in Table 5.2-19. For this case, the spectral bands are divided to core spectral bands and IGF spectral bands.

5.3.3.4.3 Global ILD normalization

The single global ILD normalization value is determined based on the energies of the first and second channel as follows:

$$\mathrm{E}_1 = \sqrt{\sum_{k=0}^{L} X_1^{MDCT}[k]^2} \tag{5.3-272}$$

$$E_2 = \sqrt{\sum_{k=0}^{L} X_2^{MDCT}[k]^2} \tag{5.3-273}$$

$$ILD = \frac{E_1}{E_1 + E_2} \tag{5.3-274}$$

where: $\mathrm{X}_1^{\mathrm{MDCT}}[k]$ is the k-th coefficient of the MDCT spectrum of the first channel, $X_2^{\mathrm{MDCT}}[\mathrm{k}]$ is the k-th coefficient of the MDCT spectrum of the second channel respectively, and L is the frame length as in number of MDCT coefficients.

The final normalization value is calculated using the quantized ILD value, calculated as:

$$\widehat{\mathrm{ILD}} = max\left(1, \min\left(2^{\mathrm{b_{ILD}}} - 1, \left\lfloor 2^{\mathrm{b_{ILD}}} * \mathrm{ILD} + 0.5 \right\rfloor\right)\right) \tag{5.3-275}$$

$$g_{ILD} = \frac{2^{b_{ILD}}}{\widehat{ILD}} - 1 \tag{5.3-276}$$

where $\widehat{ILD}$ is the quantized ILD, b_{ILD} is the number of bits selected for quantization, and g_{ILD} is the normalization value.

Then the spectrum of either the first channel or second channel is normalized, depending on the normalization value, by applying g_{ILD} to the spectral coefficients, to achieve similar energy levels as follows:

$$\begin{aligned} \bar{\mathrm{X}}_1^{\mathrm{MDCT}}[\mathrm{k}] &= g_{\mathrm{ILD}} \cdot \mathrm{X}_1^{\mathrm{MDCT}}[\mathrm{k}], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} < 1 \\ \bar{X}_2^{MDCT}[k] &= 1/g_{ILD} \cdot X_2^{MDCT}[k], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} > 1 \end{aligned} \tag{5.3-277}$$

where $L_{frameTCX}$ represents the full bandwidth frame length in bins and $\bar{X}_1^{MDCT}$, $\bar{X}_2^{MDCT}$ represent the respective normalized signals.

The MDST spectrum coefficients are also normalized with the same normalization value:

$$\begin{aligned} \bar{X}_1^{MDST}[k] &= g_{ILD} \cdot X_1^{MDST}[k], \quad \text{for } \mathrm{k} = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} < 1 \\ \bar{X}_2^{MDST}[k] &= 1/g_{ILD} \cdot X_2^{MDST}[k], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } g_{ILD} > 1 \end{aligned} \tag{5.3-278}$$

5.3.3.4.4 Conditional ILD correction

When encoding a TCX subframe at index n in mid-high stereo mode, with both channels being coded with an identical number of subframes in the given frame, and band-wise encoding mode (band-wise M/S) is selected for n as described in clause 5.3.3.4.5, the following additional high-frequency inter-channel magnitude alignment is performed as a counterpart to the high-frequency inter-channel phase alignment described in clause 5.3.3.3.4.3.

Let $sfb < n_{Bands,core}$ denote the index of the highest-frequency stereo spectral band with a width of less than 12 samples (see also Table 5.3-13), where $n_{Bands,core}$ is defined as in clause 5.3.3.4.5 below. If TNS coding is disabled in both channels for n, calculate, for each channel c and each band $i = sfb, sfb + 1, \ldots n_{Bands,core} - 1$, the sum $S_c[i]$ of the power spectral values of the MCLT coefficients $X_c^{\mathrm{MDCT}} + j \cdot X_c^{\mathrm{MDST}}$ associated with c and i (i.e., squared real parts + squared imaginary parts). Otherwise (i.e., if TNS coding is used in at least one channel for n), approximate, for c and i, the sum $S_c[i]$ of the power spectral values by accumulating only the squared values of X_c^{MDCT} (i.e, only the squared real parts).

If, for both of the two channels, $S_c[i]$ is greater than zero, calculate, using the average $S_{avg}[i]$ of the two $S_c[i]$ values,

$nrgRatio_c[i] = (\max(0.25, \min(4.0, \sqrt{(Savg[i] / Sc[i])})))^{\circ}$

where attenuation exponent o = 0.25 when *i* == *sfb* or element_brate == 64kbps, otherwise o = 0.5. The channel-wise energy ratios $nrgRatio_c$ are then used to perform band-wise corrective ILD reduction for each *i*:

- $X_c^{MDCT}[b_{offset}(i) + k] = nrgRatio_c[i] \cdot \overline{X}_c^{MDCT}[b_{offset}(i) + k]$;
- $X_c^{MDST}[b_{offset}(i) + k] = nrgRatio_c[i] \cdot \overline{X}_c^{MDST}[b_{offset}(i) + k]$;

where $k = 0, 1, \ldots N_{bins}(i) - 1$ and $b_{offset}(i)$ as well as $N_{bins}(i)$ are defined as in clause 5.3.3.4.5 below.

5.3.3.4.5 Band-wise M/S decision

The stereo encoder selects between three stereo coding modes namely full mid-side encoding (full M/S), full dual mono encoding (dual-mono or L/R) and the band-wise encoding mode (band-wise M/S or L/R). To determine which mode should be selected for encoding, the total number of bits needed for each of the stereo coding modes is estimated, given the available number of bits for encoding using perceptual entropy coding. The stereo encoding mode with the smallest estimated number of bits is selected.

Stereo encoding may only be applied when the coding mode for both channels is identical (either TCX20 or TCX10) and therefore, spectral resolution is the same. Otherwise, the stereo encoding mode is forced to dual-mono and the steps described below are skipped.

The detailed steps for the M/S decision and determination of the stereo encoding mode are described below.

The SQ gain G is first estimated given the normalized spectra of the first channel and the second channel and the number of bits available for encoding.

The full M/S transform of the first channel and second channel spectra is calculated as:

$$X_M[k] = \frac{\sqrt{(2)}}{2}(X_1[k] + X_2[k]), \quad for\ k = 0, \ldots, L_{frame} - 1 \qquad (5.3\text{-}279)$$

$$X_s[k] = \frac{\sqrt{(2)}}{2}(X_1[k] - X_2[k]), \quad for\ k = 0, \ldots, L_{frame} - 1$$

Then the respective number of bits, b_{LR} for the dual-mono mode and b_{MS} for the full M/S mode, are estimated by quantizing the respective spectra and applying a perceptual entropy coder described in clause 5.2.2.3.3.3 given the SQ gain G and the number of bits available for the encoding.

The number of bits b_{BW} needed for the band-wise M/S encoding mode are estimated using the following formula:

$$b_{BW} = n_{Bands} + \sum_{i=0}^{n_{Bands,core}-1} \min\left(b_{bwLR}^i, b_{bwMS}^i\right) \qquad (5.3\text{-}280)$$

where: $n_{Bands,core}$ is the number of the core spectral bands of the normalized audio signal, one bit is needed for each band to signal the mode for the particular band;

b_{bwMS}^i is an estimation for a number of bits that are needed for encoding an i-th spectral band of the mid signal and for encoding the i-th spectral band of the side signal;

b_{bwLR}^i is an estimation for a number of bits that are needed for encoding an i-th spectral band of the first signal and for encoding the i-th spectral band of the second signal.

The stereo encoding mode decision for each spectral band is stored in a $n_{Bands,core}$ long array, the so-called M/S mask, where for encoding the i-th spectral band M/S is signalled with 1 and dual-mono is signalled with 0.

If $b_{LR} < b_{BW}$ then the stereo mode is DUAL_MONO and the M/S mask is an n_{Bands} array of 0s.

If $b_{MS} < b_{BW}$ then the stereo mode is MS_FULL and the M/S mask is an n_{Bands} array of 1s.

Otherwise, the stereo encoding mode is BW_MS and the M/S mask is populated accordingly with decision m_i depending on the respective estimated number of bits for M/S b_{bwMS}^i and dual-mono b_{bwLR}^i for the i-th spectral band as follows:

$$\mathrm{m_i} = \begin{cases} 1, & if\ b^i_{bwMS} < b^i_{bwLR} \\ 0, & if\ b^i_{bwLR} < b^i_{bwMS} \end{cases} \quad i = 0 \ldots n_{Bands,core} - 1 \tag{5.3-281}$$

If IGF is enabled for the given configuration, the M/S decision for the IGF spectral bands is determined by the algorithm described in clause 5.3.3.4.6.

5.3.3.4.6 M/S decision for IGF

For the IGF spectral bands the M/S decision is based on the correlation estimate of the IGF bands and the correlation estimate and the chosen stereo mode for the bands below the IFG region. For the decision a number of thresholds is defined, a target coherence threshold $thresh_{coh,target} = 0.6$, a source coherence threshold $thresh_{coh,source} = 0.6$, and a panning threshold $thresh_{panning}$=0.07.

If the core stereo mode is set to MS_FULL, the final target coherence is obtained by multiplying the target coherence threshold by 0.7: $thresh_{coh,target} = 0.7 thresh_{coh,target}$.

For each IGF scale factor band $k = 0, \ldots, nB - 1$ the coherence and panning are calculated for the IGF band itself and the source band using the mapping from 5.2.2.3.3.4.7:

$$E_{1,t}(\mathrm{k}) = \sum_{tb=t_k}^{t(k+1)-1} X_1(tb) X_1(tb) \tag{5.3-282}$$

$$E_{2,t}(\mathrm{k}) = \sum_{tb=t_k}^{t(k+1)-1} X_2(tb) X_2(tb)$$

$$E_{1,2,t}(\mathrm{k}) = \sqrt{E_{1,t} E_{2,t}}$$

$$R_t(\mathrm{k}) = \sum_{tb=t_k}^{t(k+1)-1} X_1(tb) X_2(tb)$$

$$C_t(\mathrm{k}) = \begin{cases} \frac{R_t(\mathrm{k})}{E_{1,2,t}(\mathrm{k})}, & if\ E_{1,2,t}(\mathrm{k}) > 0 \\ 0, & else \end{cases}$$

$$p_t(\mathrm{k}) = \begin{cases} \frac{R_t(\mathrm{k})}{E_{1,t}(\mathrm{k})}, & if\ E_{1,t}(\mathrm{k}) > 0 \\ 0, & else \end{cases}$$

$$E_{1,s}(\mathrm{k}) = \sum_{tb=t_k}^{t(k+1)-1} X_1\big(m(tb)\big) X_1(m(tb)) \tag{5.3-283}$$

$$E_{2,s}(\mathrm{k}) = \sum_{tb=t_k}^{t(k+1)-1} X_2\big(m(tb)\big) X_2(m(tb))$$

$$E_{1,2,s}(\mathrm{k}) = \sqrt{E_{1,s} E_{2,s}}$$

$$R_s(\mathrm{k}) = \sum_{tb=t_k}^{t(k+1)-1} X_1\big(m(tb)\big) X_2(m(tb))$$

$$C_s(\mathrm{k}) = \begin{cases} \frac{R_s(\mathrm{k})}{E_{1,2,s}(\mathrm{k})}, & if\ E_{1,2,s}(\mathrm{k}) > 0 \\ 0, & else \end{cases}$$

$$p_s(\mathrm{k}) = \begin{cases} \frac{R_s(\mathrm{k})}{E_{1,s}(\mathrm{k})}, & if\ E_{1,s}(\mathrm{k}) > 0 \\ 0, & else \end{cases}$$

The first stereo decision for each IGF band is made according to the following pseudo code:

```
if (|C_t(k)| > thres_coh,target)
{
    if ( |C_s(k)| > thresh_coh,source )
    {
        if ( |p_t(k) − p_s(k)| < thresh_panning ∧ |p_t − 1| > 2thresh_panning)
        {
            m_igf(k)=0
        }
        else
        {
            m_igf(k) = 1
        }
```

```
        }
        else
        {
            m_igf(k) = 1
        }
}
else
{
    m_igf(k) = 0
}
```

where the two possible two-channel representations are a M/S representation when $m_{IGF} = 1$ and a L/R representation when $m_{IGF} = 0$.

If for no IGF band the MS map is set to 1, the IGF stereo mode is set to DUAL_MONO.

If for no IGF band the MS map is set to 0, the IGF stereo mode is set to MS_FULL.

If at least for one IGF band the MS map is set to 1 but the total number of IGF bands with the MS map set to 1 is less than a quarter of the number of IGF bands, the IGF stereo mode is set to DUAL_MONO the IGF bands MS map entries are all set to 0.

If at least for one IGF band MS map is set to 0 but the total number of IGF bands where the MS map is set to 0 is less than a quarter of the number of IGF bands, the IGF stereo mode is set to MS_FULL the bands MS map entries are all set to 1.

Otherwise, the IGF stereo mode is set to BW_MS.

5.3.3.4.7 M/S Transformation

Depending on the stereo encoding mode decision described in clause 5.3.3.4.5 either full-mid-side (MS_FULL), full dual-mono (DUAL_MONO) or band-wise encoding (BW_MS) is applied.

If the selected mode is full mid-side (MS_FULL) and either no IGF is active or IGF is active and the selected IGF stereo mode is also full mid-side (MS_FULL): a mid signal from the first channel and from the second channel of the normalized audio signal is obtained as a first channel as shown in Equation (5.3-284) and a side signal from the first channel and from the second channel of the normalized audio signal as a second channel as shown in Equation (5.3-285).

$$\tilde{X}_1[k] = \frac{\sqrt{(2)}}{2}(\bar{X}_1[k] + \bar{X}_2[k]), \quad for\ \mathrm{k} = 0, \ldots, \mathrm{L_{frame}} - 1 \tag{5.3-284}$$

$$\tilde{X}_2[k] = \frac{\sqrt{(2)}}{2}(\bar{X}_1[k] - \bar{X}_2[k]), \quad for\ k = 0, \ldots, L_{frame} - 1 \tag{5.3-285}$$

where $\bar{X}_1, \bar{X}_2$ are the normalized inputs and $\widetilde{\mathrm{X}}_1, \widetilde{\mathrm{X}}_2$ represent the processed outputs of the first and second channel. The factor $\frac{\sqrt{(2)}}{2}$ is employed to preserve the energy level of the signal after decoding and inverse transformation.

If the full dual-mono (DUAL_MONO) mode is selected and either no IGF is active or IGF is active and the selected IGF stereo mode is also full dual-mono (DUAL_MONO) then no processing occurs and the normalized audio signals of the two channels are encoded as is.

If the band-wise stereo encoding mode (BW_MS) is selected, then the M/S mask described in clause 5.3.3.4.5 which represents the decision for each spectral band determines whether mid-side encoding or whether dual-mono encoding is employed in a given spectral band.

If mid-side encoding is employed, for a given spectral band the respective processed output spectral band of the first channel is a mid-signal based on the respective spectral bands of the normalized audio signals of the first channel and the second channel. Accordingly, the respective processed output spectral band of the second channel is a side signal based on the respective spectral bands of the normalized audio signals of the first channel and the second channel.

If dual-mono encoding is employed for a given spectral band, then no processing occurs, and the output spectral band for the first channel is the respective spectral band of the input normalized audio signal and the output spectral band of the second channel is the respective spectral band of the second channel normalized audio signal.

This is depicted in Equations (5.3-286) and (5.3-287).

For each spectral band $i = 0 \ldots n_{Bands,core} - 1$:

$$\tilde{X}_1\left[b_{offset}(i)+k\right] = \begin{cases} \frac{\sqrt{2}}{2}\left(\bar{X}_1\left[b_{offset}(i)+k\right] + \bar{X}_2\left[b_{offset}(i)+k\right]\right), & \text{if } m_i = 1 \\ \bar{X}_1\left[b_{offset}(i)+k\right], & \text{otherwise} \end{cases} \quad (5.3\text{-}286)$$

$$\tilde{X}_2\left[b_{offset}(i)+k\right] = \begin{cases} \frac{\sqrt{2}}{2}\left(\bar{X}_1\left[b_{offset}(i)+k\right] - \bar{X}_2\left[b_{offset}(i)+k\right]\right), & \text{if } m_i = 1 \\ \bar{X}_2\left[b_{offset}(i)+k\right], & \text{otherwise} \end{cases} \quad (5.3\text{-}287)$$

where $k = 0 \ldots N_{bins}(i) - 1$, $\bar{X}_1\left[b_{offset}(i)+k\right]$, $\bar{X}_2\left[b_{offset}(i)+k\right]$ is the k-th bin of the *i*-th spectral band of the normalized first and second channel respectively, $\tilde{X}_1\left[b_{offset}(i)+k\right]$, $\tilde{X}_2\left[b_{offset}(i)+k\right]$ is the k-th bin of the *i*-th spectral band of the processed first and second channel output, $b_{offset}(i)$ is the spectral offset of the *i*-th band from Table 5.3-17, m_i is the MS mask decision of the the *i*-th spectral band set in equation (5.3-281), $N_{bins}(i)$ is the width of the *i*-th spectral band as in number of spectral bins in Table 5.3-17, $n_{Bands,core}$ is the total number of core spectral bands.

If IGF is active the same processing is applied to the IGF bands using the IGF stereo mode and the IGF MS mask.

If IGF active and either core stereo mode or IGF stereo mode are not DUAL_MONO, also the spectra $\tilde{X}_{1,inv}$, $\tilde{X}_{2,inv}$ using the inverse core MS masks, i.e. MS masks where 0 and 1 are replaced by each other compared to the MS masks defined above, are created. If the MDST spectra are needed for the power spectra calculation, the M/S and inverse M/S transformation are also applied to the MDST spectra.

5.3.3.4.8 Encoding of the stereo parameters

The stereo parameters that are encoded to the bitstream to revert the stereo processing on the decoder side are the following:

- The stereo encoding mode
- The quantized ILD $\widehat{ILD}$
- If the stereo encoding mode is band-wise M/S the ms mask
- The stereo mode for the IGF spectral bands (if applicable)
- If the IGF stereo encoding mode is band-wise M/S the ms mask for the IGF spectral bands

5.3.3.5 Power spectrum calculation and spectrum noise measure

After the modified spectra from the stereo processing for the first and second channel are derived, the power spectrum is newly calculated. That is needed for the IGF analysis and the calculation of the spectrum noise measure for each channel. The procedure is identical as in the EVS MDCT based TCX mode and is described in detail in clause 5.3.3.2.5 of [3]. As mentioned in clause 5.3.3.3.6.3 if TNS is applied, then the MDST is not used for the calculation of the power spectrum, instead an estimate from the MDCT spectrum is used. The power spectra are also calculated for the spectra with inverse M/S processing. If IGF is active, also the inverse MS mask MDCT and if applicable MDST spectra are used to calculate the power spectra P_{inv} with the inverse MS mask:

5.3.3.6 Intelligent gap filling (IGF)

5.3.3.6.1 General overview

The intelligent gap filling bandwidth extension is carried out in general for each channel separately on the modified spectra from the stereo processing. The details of the IGF algorithm are described in clause 5.3.3.2.11 of [3]. The modifications that apply in the IVAS codec are described in clause 5.2.2.3.3.4. For the IGF portion parameters for the IGF reconstruction bands are encoded, giving a smaller spectral resolution in the IGF reconstruction bands compare to the core-coder encoding.

However, if certain conditions apply, special considerations for the stereo IGF encoding takes place, the details of which are described in clause 5.3.3.6.2. Stereo IGF encoding is used when the spectral resolution for both channels is

the same, meaning the transform type is identical and either when the stereo mode is band-wise M/S or when the stereo mode for the core spectral bands is different from the stereo mode for the IGF spectral bands.

5.3.3.6.2 Stereo IGF encoding

5.3.3.6.2.1 Overview

In case of stereo IGF encoding the first and second stereo processed channels are processed according to clause 5.3.3.2.11 of [3] with the modifications that apply in the IVAS codec are described in clause 5.2.2.3.3.4, and some steps in the scale factor calculation described in 5.3.3.2.11.4 are replaced by the following steps that describes how the IGF scale factor vectors for the first and second channel $g_1(k), g_2(k), k = 0,1, \dots, nB-1$ are calculated on transmission (TX) side. The SFB band wise MS masks for both core and IGF are converted to MDCT a bin wise MS masks $m_b(l)$ for the processing:

For each spectral band $i = 0 \dots n_{Bands,core} - 1$:

$$m_b(b_{offset}(i) + k) = m_{core}(i) \tag{5.3-288}$$

where: $k = 0 \dots N_{bins}(i) - 1$;

$m_b(b_{offset}(i) + k)$ is the k-th bin of the *i*-th spectral band of the bin wise M/S mask

m_{core}(i) is the i-th entry of the core M/S mask

$b_{offset}(i)$ is the spectral offset of the i -th band from Table 5.3-17

For each IGF scale factor band $i = 0, \dots, nB - 1$:

$$m_b(k) = m_{IGF}(i) \tag{5.3-289}$$

where: $k = t(k) \dots t(k+1) - 1$;

m_b(k) is the k-th bin of the bin wise M/S mask

m_{IGF}(i) is the i-th entry of the IGF M/S mask

$t(k)$ are the band borders of the IGF bands

5.3.3.6.2.2 Stereo IGF scale factor calculation

5.3.3.6.2.2.1 Complex valued calculation

In case the TCX power spectra P_n and $P_{n_{inv}}$ are available the IGF scale factor values g_n are calculated using P, P_{inv}:

$$E_{\substack{n(k) \\ cplx,target}} = \sqrt{\frac{1}{t(k+1)-t(k)} \sum_{tb=t(k)}^{t(k+1)-1} P_n(tb)}, k = 0,1, \dots, nB-1; n = 1,2 \tag{5.3-290}$$

And let $m: N \rightarrow N$ be the mapping function which maps the IGF target range into the IGF source range described in subclause 5.2.2.3.3.4.7, calculate:

$$E_{\substack{n(k) \\ cplx,source}} = \sqrt{\frac{1}{t(k+1)-t(k)} \sum_{tb=t(k)}^{t(k+1)-1} \begin{cases} P_n\big(m(tb)\big) \; if \; m_b\big(m(tb)\big) = m_b(tb) \\ P_{n_{inv}}\big(m(tb)\big) \; if \; m_b\big(m(tb)\big) \neq m_b(tb) \end{cases}}, k = 0,1, \dots, nB-1; n = 1,2 \tag{5.3-291}$$

$$E_{\substack{n(k) \\ real,source}} = \sqrt{\frac{1}{t(k+1)-t(k)} \sum_{tb=t(k)}^{t(k+1)-1} \begin{cases} \tilde{X}_n\big(m(tb)\big) \; if \; m_b\big(m(tb)\big) = m_b(tb) \\ \tilde{X}_{n_{inv}}\big(m(tb)\big) \; if \; m_b\big(m(tb)\big) \neq m_b(tb) \end{cases}}, k = 0,1, \dots, nB-1; n = 1,2 \tag{5.3-292}$$

where $t(0), t(1), \dots, t(nB)$ shall be already mapped with the function tF, see subclause 5.3.3.2.11.1.1 of [3], and nB are the number of IGF bands.

With these energies, the gains are calculated according to equations (1004) and (1005) of [3] with IGF damping according to 5.2.2.3.3.4.2. For the SFM estimate of the source spectrum SFM_{src} used both in the damping and the whitening a modified source spectrum is used:

$$P_{n,SFM}(m(tb)) = \begin{cases} P_n(m(tb)) \; if \; m_b(m(tb)) = m_b(tb) \\ P_{n_{inv}}(m(tb)) \; if \; m_b(m(tb)) \neq m_b(tb) \end{cases} \quad (5.3\text{-}293)$$

Where $tb = t(k), \dots, t(k+1) - 1, k = 0,1, \dots, nB - 1; n = 1,2.$

5.3.3.6.2.2.2 Real valued calculation

If the TCX power spectra are not available calculate:

$$\underset{real}{E_n(k)} = \sqrt{\frac{1}{t(k+1)-t(k)} \sum_{tb=t(k)}^{t(k+1)-1} \tilde{X}_n(tb)}, k = 0,1, \dots, nB - 1; n = 1,2 \quad (5.3\text{-}294)$$

With these energies, the gains are calculated according to equations (1007) and (1008) of [3] and the IGF damping in 5.2.2.3.3.4.2..

5.3.3.7 Bitrate distribution

The numbers of bits for quantization and encoding are assigned adaptively to each channel based on the channel energies. First the ratio of the energy of the first channel to total energy of both channels is computed in the same fashion as is done for the initial ILD calculation described with equations (5.3-272)-(5.3-274). The quantized ratio is signalled in the bitstream and is then used for splitting the bits to be assigned to each channel in order to be in sync with the decoder. The quantized ratio is calculated:

$$\hat{r}_{split} = \left\lfloor 2^{b_{\hat{r}_{split}}} * r_{split} + 0.5 \right\rfloor \quad (5.3\text{-}295)$$

where r_{split} is the split ratio derived from equation (5.3-274) applied on the stereo processed channels and $b_{\hat{r}_{split}}$ is the number of bits to code the split ratio.

The bits assigned to each channel for encoding the final spectra are divided among the total number of bits that remain after encoding the parameters of the processing steps namely TCX-LTP, TNS, SNS, IGF, the stereo processing parameters and the split ratio $\hat{r}_{split}$. Additionally, the number of bits reserved for the noise filling levels after quantization are also subtracted:

$$b_{available} = b_{total} - b_{tcxltp} - b_{TNS} - b_{SNS} - b_{IGF} - b_{stereo} - b_{NF} - b_{\hat{r}_{split}}$$

Then the bitrate assigned to each channel for the encoding of the spectrum is derived as follows:

$$b_{ch0} = \frac{\hat{r}_{split}}{2^{b_{\hat{r}_{split}}}} \cdot b_{available} \quad (5.3\text{-}296)$$

and

$$b_{ch1} = b_{available} - b_{ch0} \quad (5.3\text{-}297)$$

5.3.3.8 Quantization and encoding

5.3.3.8.1 Spectrum quantization and coding

In general, quantization and coding of the MDCT coefficients follows the procedure described for EVS in clause 5.3.3.2.7 to 5.3.3.2.13 of [3]. For changes with respect to clause 5.3.3.2.11 (Intelligent Gap Filling) see clause 5.2.2.3.3.4 above. The Arithmetic coder described in [3], clause 5.3.3.2.8, has been replaced by a range coder for all MDCT-based encoding modes, see 5.2.2.3.3.3. The remaining changes to the MDCT spectrum quantization are described in the following.

Generally, in MDCT-based stereo coding, the context based arithmetic coder with no Harmonic Model is used. Residual quantization as described in 5.3.3.2.9.3 of [3] is used only for bitrates <= 64000kbps. Also, no Adaptive Low-Frequency Emphasis (ALFE) is applied to the spectrum before quantization, hence clause 5.3.3.2.4 of [3] is not applied.

5.3.3.8.2 Global gain estimator

The procedure for getting a first estimate for the global gain as described in 5.3.3.2.8.1.1 of [3] is modified for MDCT-Stereo encoding. When computing the energy of a 4-tuple (equation (929) of [3]), the lowest possibly quantized gain value $minGainInv = 0.5 \log_{10}(\frac{L}{160})$ and the expected value of the quantization noise energy per 4-tuple $\log_{10}\left(\frac{4}{12}\right) \cong 0.94$ are added:

$$E[k] = \sum_{i=0}^{4} \hat{X}^2[4k+i] + minGainInv + 0.94 \quad (5.3\text{-}298)$$

Then, then same bisection search as in 5.3.3.2.8.1.1 of [3] is performed, with the following modifications:

fac=18.4.

offset = fac – fac/512.

Only 8 iterations are performed.

The first gain estimate is then calculated according to:

$$g_{TCX} = 10^{\frac{offset}{2} - minGainInv} \quad (5.3\text{-}299)$$

5.3.3.8.3 Rate-loop for constant bit rate and global gain

The rate-loop described in 5.3.3.2.8.1.2 of [3] is performed similar, but the maximum number of iterations is lowered to 2.

5.3.3.8.4 Stereo noise level estimation

The noise factor computation is done separately for each channel according to 5.3.3.2.10 of [3]. In equation (977), section 5.3.3.2.10.3 the used LTP gain g_{LTP} is replace by a smoothed LTP gain $\bar{g}_{LTP}$:

$$\bar{g}_{LTP} = 0.4 g_{LTP} + 0.2 \sum_{f=1}^{3} g_{LTP}^{[-f]} \quad (5.3\text{-}300)$$

5.3.4 Switching between stereo modes

5.3.4.1 Overview

The switching between stereo modes can happen as a consequence of the IVAS total bitrate change (see Table 5.3-1) or of the stereo mode selector output (see clause 5.3.2.5). This clause then deals with mechanical aspects of the stereo mode switching, e.g. memory handling, past buffers and parameters updates, or signal continuity mechanism that ensure seamless switching without subjective artifacts.

The stereo encoder (in general the CPE coding tool) consists of the TD stereo, DFT stereo and MDCT stereo coding modes as shown in Figure 5.3-41. Each of them represents a different technology in terms of downmixing, coding, delay, or number of transport channels. However, each of them uses one or two core-coder(s) while the transport channel(s) entering the core-coder(s) always belong to the same time instances.

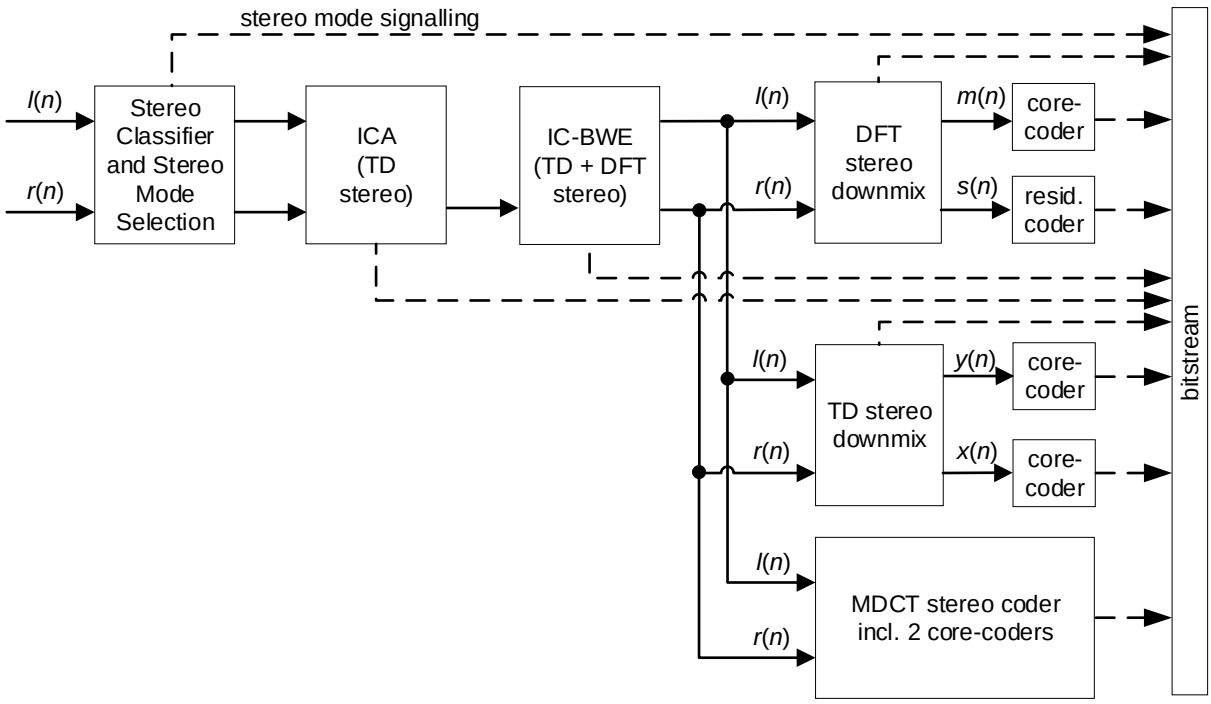


Figure 5.3-41: High-level block diagram illustrating stereo coding modes

5.3.4.2 General

The stereo encoder performs buffering of one 20-ms frame of stereo input signal (left and right channels $l(n)$ and right $r(n)$ where the sample index $n = 0, \ldots, L-1$, L being the frame length in samples), few classification steps, down-mixing, pre-processing and actual coding. A 8.75 ms look-ahead is available and used mainly for analysis, classification and Overlap-Add (OLA) operations used in transform-domain based technologies like as in the TCX core, the HQ core, and the FD-BWE. Note that there are differences between stereo modes wrt. the resampling operation as described in clause 5.2.2.2.2: the resampling is performed either in the time domain (TD stereo and MDCT stereo modes) while the length of the redressing segment is 8.4375 ms in TD stereo or 0.9375 ms in MDCT stereo, respectively, or in the DFT domain (DFT stereo mode) while the length of the redressing segment is 3.125 ms. These differences need to be carefully taken into account during the stereo switching.

Switching between stereo modes involves the use of the stereo mode switching mechanism to maintain continuity of the following input signals 1) to 5) to enable adequate processing of these signals in the stereo encoder:

1) the input stereo signal including the left $l(n)$ and right $r(n)$ channels, used for the time-domain transient detection and IC-BWE;

2) The stereo down-processed signal (down-mixed signal for TD and DFT stereo modes) at the input stereo signal sampling rate:

- DFT stereo (described in clause 5.3.2.4): downmixed (mid-)channel $m(n)$ defined in clause 5.3.2.4.11;

- TD stereo (described in clause 5.3.2.3): primary channel $y(n)$ and secondary channel $x(n)$ defined in clause 5.3.2.3.3;

- MDCT stereo (described in clause 5.3.3): original (no down-mix) left channel $l(n)$ and right channel $r(n)$.

3) Down-processed signal (down-mixed signal for TD and DFT stereo modes) at 12.8 kHz sampling rate – used in pre-processing;

4) Down-processed signal (down-mixed signal for TD and DFT stereo modes) at internal sampling rate – used in core-coder;

5) High-band (HB) input signal – used in core-coder BWEs.

While it is straightforward to maintain the continuity for signal 1) above, it is challenging for signals 2) to 5) due to several aspects, for example a different down-mixing mechanism, a different length of the re-computed part of the look-ahead, use of ICA in the TD stereo mode only, etc.

The following Table 5.3-19 then lists in a sequential order processing operations for each frame depending on the current stereo coding mode. Note that Table 5.3-19 corresponds to operations shown in Figure 5.3-41 but with more details.

Table 5.3-19: Processing operations in IVAS stereo encoder

DFT stereo mode	TD stereo mode	MDCT stereo mode
Stereo classification and stereo mode selection		
Memory allocation/deallocation		
	Set TD stereo sub-mode	
Stereo mode switching updates		
	ICA	
IC-BWE buffering		
TD transient detectors		
Stereo encoder configuration		
DFT analysis	TD analysis	
Stereo processing and down-mixing in DFT domain	Weighted down-mixing in TD domain	
DFT synthesis		
Front pre-processing		
Core-coder configuration		
	TD stereo configuration	
DFT stereo residual coding		
Further pre-processing		
Core-coding		
Common stereo updates		

In the following text, only operations responsible for the stereo mode switching are described in detail.

Stereo classification and stereo mode selection

See clause 5.3.2.5.

Memory allocation/handling

See details later in clause 5.3.4.3.

Set TD stereo sub-mode (TD stereo mode only)

Selects between regular TD sub-mode and LRTD sub-mode within the TD stereo.

Stereo mode switching updates

The stereo mode switching mechanism updates long-term parameters and updates or resets past buffer memories.

When switching from the DFT stereo mode to the TD stereo mode, it resets TD stereo and ICA static memory data structures. These data structures store the parameters and memories of the TD stereo analysis, weighted down-mixing, and ICA. Then the stereo mode switching mechanism sets a TD stereo past frame mixing ratio index according to the TD stereo mode being regular TD sub-mode or LRTD sub-mode. Specifically,

- The previous frame mixing ratio index (defined in clause 5.3.2.3.3) is set to 15, indicating that the down-mixed mid-channel $m(n)$ is coded as the primary channel $y(n)$, where the mixing ratio is $\beta = 0.5$, in the regular TD stereo sub-mode; or

- The previous frame mixing ratio index is set to 31, indicating that the left channel $l(n)$ is coded as the primary channel $y(n)$, in the LRTD sub-mode of the TD stereo.

When switching from the TD stereo mode to the DFT stereo mode, the stereo mode switching mechanism resets the DFT stereo data structure. This DFT stereo data structure stores parameters and memories related to the DFT stereo processing and down-mixing module.

Also, the stereo mode switching mechanism transfers some stereo-related parameters between data structures. As an example, parameters related to time shift and energy between the channels $l(n)$ and $r(n)$, namely a side gain (or ILD parameter) and ITD parameter of the DFT stereo mode are used to update a target gain and correlation lags (ICA parameters) of the TD stereo mode and vice versa. These target gain and correlation lags are described in clause 5.3.2.3.1.

ICA encoder (TD stereo only)

ICA is performed using ITD synchronization between the two input channels $l(n)$ and $r(n)$ in the time-domain. This is achieved by delaying one of the input channels ($l(n)$ or $r(n)$) and by extrapolating a missing part of the down-mixed signal corresponding to the length of the ITD delay while a maximum value of the ITD delay is 7.5 ms. The time alignment, i.e. the ICA time shift, is applied first and alters the most part of the current TD stereo frame. The extrapolated part of the look-ahead down-mixed signal is recomputed and thus temporally adjusted in the next frame based on the ITD estimated in that next frame.

When no stereo mode switching is anticipated, the 7.5 ms long extrapolated signal is re-computed in the ICA encoder. However, when stereo mode switching may happen, namely switching from the DFT stereo mode to the TD stereo mode, a longer signal is subject to re-computation. The length then corresponds to the length of the DFT stereo redressed signal plus the FIR resampling delay, i.e. 8.75 ms + 0.9375 ms = 9.6875 ms. Clause 5.3.4.4.2 describes these features in more detail.

Another purpose of the ICA encoder from Figure 5.3-41 is the scaling of the input channel $r(n)$. The scaling gain, i.e. the gain as described in clause 5.3.2.3.1, is estimated as a logarithm ratio of the $l(n)$ and $r(n)$ channels energies smoothed with the previous frame target gain at every frame regardless of the DFT or TD stereo mode being used. The target gain estimated in the current frame (20 ms) is applied to the last 15 ms of the current input channel $r(n)$ while the first 5 ms of the current channel $r(n)$ is scaled by a combination of the previous and current frame target gains in a fade-in / fade-out manner.

5.3.4.3 Memory handling

The switching between stereo modes comprises an operation of memory handling (i.e. memory allocation and deallocation). To perform the operation of memory handling, the mechanism of switching between the DFT, TD and MDCT stereo modes dynamically allocates/deallocates inter-frame static memory data structures to/from the DFT, TD and MDCT stereo modes depending on the previous and current stereo mode. Such memory allocation keeps the static memory impact of the CPE coding tool as low as possible by maintaining only those data structures that are employed in the current frame.

For example, in a first DFT stereo frame following a TD stereo frame, the data structures related to the TD stereo mode (for example TD stereo data handling, second core-coder data structure) are freed (deallocated) and the data structures related to the DFT stereo mode (for example DFT stereo data structure) are instead allocated and initialized. It is noted that the deallocation of the further unused data structures is done first, followed by the allocation of newly used data structures. This order of operations is important to not increase the static memory impact at any point of the encoding.

A summary of main inter-frame static memory data structures as used in various stereo modes is shown in Table 5.3-20.

Table 5.3-20: Allocation of encoder data structures in different stereo modes

Data structure	DFT stereo mode	TD stereo mode		MDCT stereo mode
		regular sub-mode	LRTD sub-mode	
CPE main structure	X	X	X	X
Stereo classifier	X	X	X	X
DFT stereo	X	–	–	–
TD stereo	–	X	X	–
MDCT stereo	–	–	–	X
ICA	X	X	X	–
IC-BWE	X	X	–	–
Core-coder	X	X X	X X	X X
ACELP core	X	X X	X X	– –
TCX core + IGF	X	X –	X –	X X
HQ core	X	X –	X –	– –
TD-BWE	X	X –	X X	– –
FD-BWE	X	X –	X X	– –

Note: “X” means allocated, “X X” means twice allocated.

5.3.4.4 Switching between DFT and TD stereo modes

5.3.4.4.1 Switching from TD stereo to DFT stereo

Switching from the TD stereo mode to the DFT stereo mode is illustrated in a flow chart of Figure 5.3-42.

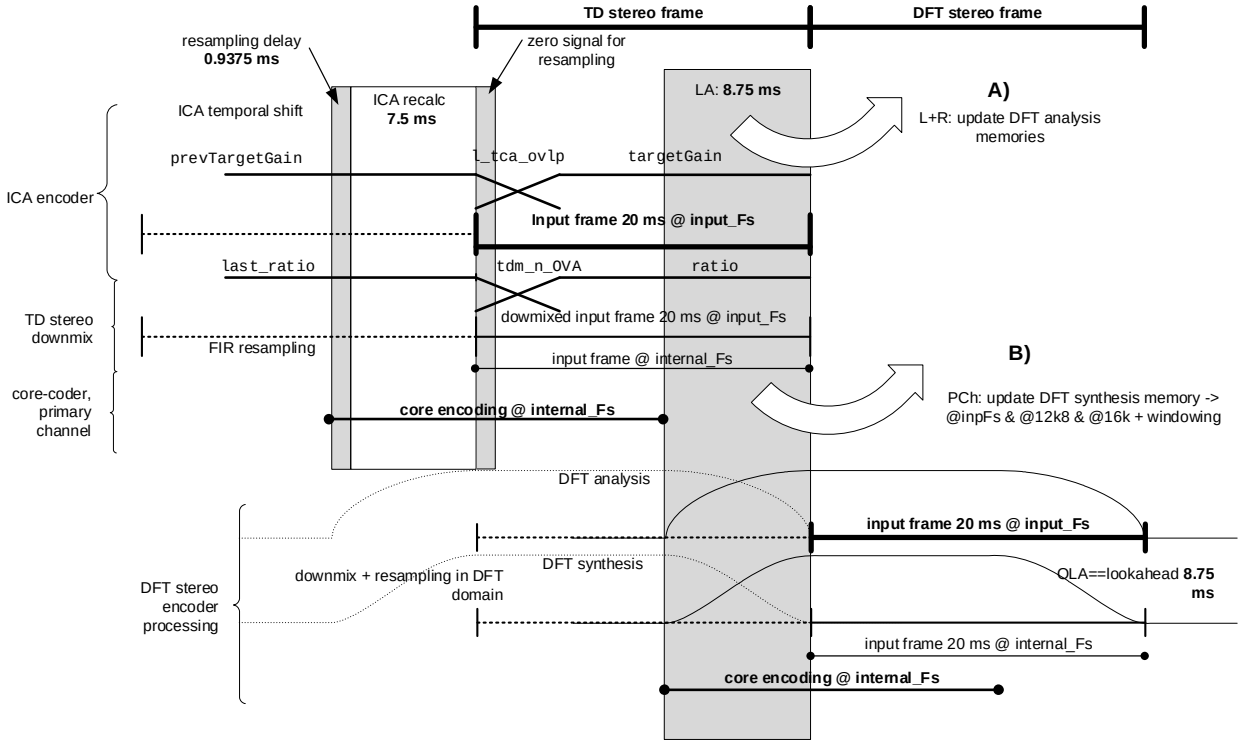


Figure 5.3-42: Flow chart illustrating operations upon switching from TD stereo to DFT stereo

Figure 5.3-42 shows two frames of stereo input signal, i.e. a TD stereo frame followed by a DFT stereo frame, with different processing operations and related time instances when switching from the TD stereo mode to the DFT stereo mode. In this scenario, a sufficiently long look-ahead is available, resampling is done in the DFT domain (thus no FIR decimation filter memory handling is needed), and there is a transition from two core-coders in the last TD stereo frame to one core-coder in the first DFT stereo frame. The following operations are then performed in response to the stereo mode selection.

The instance A) of Figure 5.3-42 refers to an update of the DFT analysis memory, specifically the DFT stereo OLA analysis memory as part of the DFT stereo data structure which is subject to windowing prior to the DFT calculating

operations. This update is done by the stereo mode switching mechanism before the ICA (see in Figure 5.3-41) and comprises storing samples related to the last 8.75 ms of the current TD stereo frame of the channels $l(n)$ and $r(n)$ of the input stereo signal. This update is done every TD stereo frame in both channels $l(n)$ and $r(n)$. Details about the DFT analysis can be found in clause 5.3.2.4.3.

The instance B) of Figure 5.3-42 then refers to an update of the DFT synthesis memory, specifically the OLA synthesis memory as part of the DFT stereo data structure which results from windowing after the IDFT calculating operations, upon switching from the TD stereo mode to the DFT stereo mode. The stereo mode switching mechanism performs this update in the first DFT stereo frame following the TD stereo frame and uses, for this update, the TD stereo memories as part of the TD stereo data structure and used for the TD stereo processing corresponding to the down-mixed primary channel $y(n)$. Details about the DFT synthesis memory can be found in clause 5.3.2.4.11 while details about the TD stereo operations can be found in clause 5.3.2.3.3.

Starting with the first DFT stereo frame, certain TD stereo related data structures, for example the TD stereo data structure and a data structure of the core-coder related to the secondary channel $x(n)$, are no longer needed and, therefore, are de-allocated, i.e. freed by the stereo mode switching mechanism.

In the DFT stereo frame following the TD stereo frame, the stereo mode switching mechanism continues the core-coding operation in one core-coder of the DFT stereo encoder with memories of the primary channel $y(n)$ core-coder (e.g. synthesis memory, pre-emphasis memory, past signals and parameters, etc.) in the preceding TD stereo frame while controlling time instance differences between the TD and DFT stereo modes to ensure continuity of several core-coder buffers, e.g. pre-emphasized input signal buffers, HB input buffers, etc. which are later used in the low-band encoder, resp. the FD-BWE high-band encoder.

5.3.4.4.2 Switching from DFT stereo to TD stereo

Switching from the DFT stereo mode to the TD stereo mode is more complicated than switching from the TD stereo mode to the DFT stereo mode, due to the more complex structure of the TD stereo encoder. The following operations performed upon switching from the DFT stereo mode to the TD stereo mode are then performed by the stereo mode switching mechanism in response to the stereo mode selection.

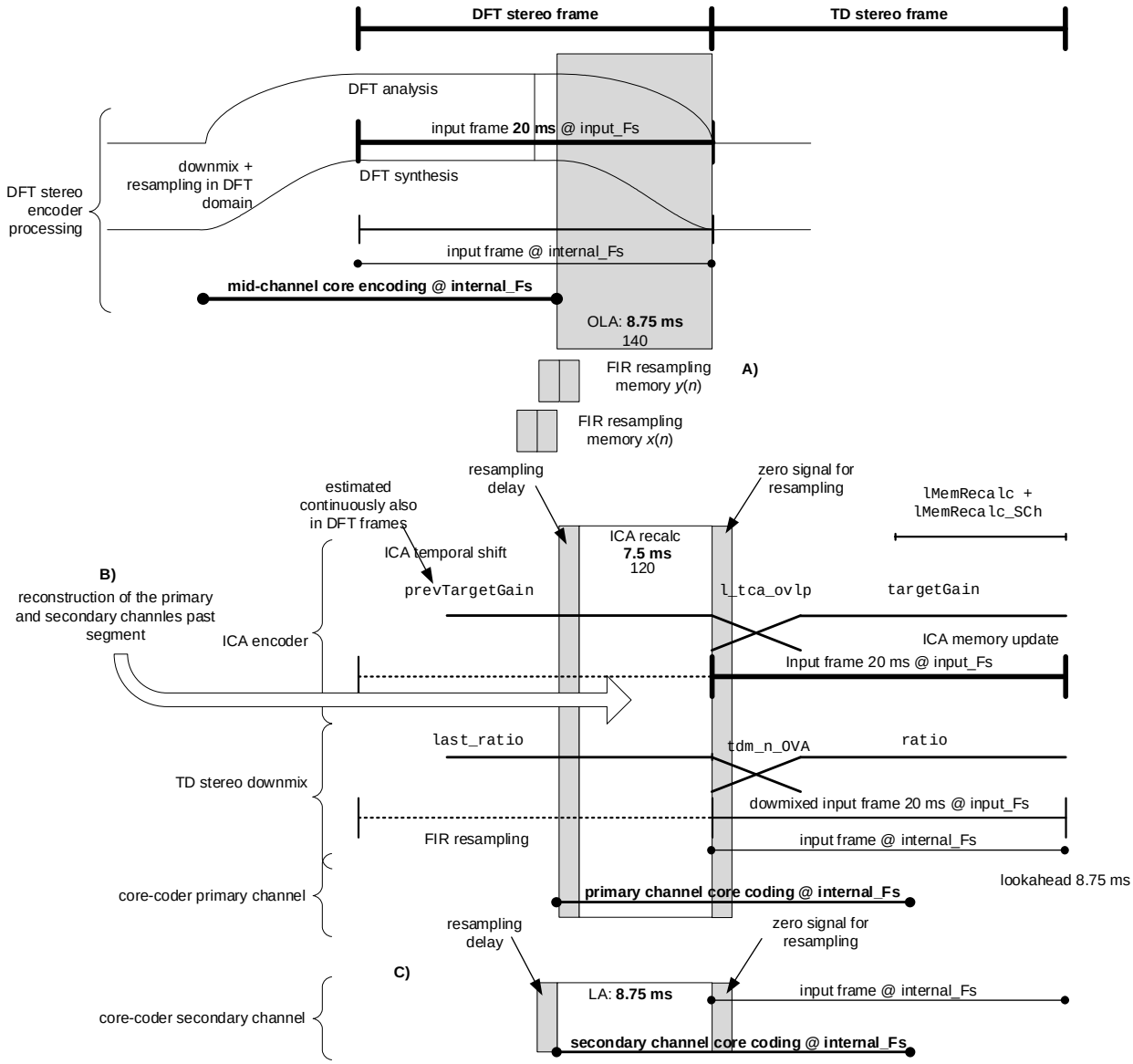


Figure 5.3-43: Flow chart illustrating operations upon switching from DFT stereo to TD stereo

Figure 5.3-43 is a flow chart illustrating processing operations in the stereo encoder upon switching from the DFT stereo mode to the TD stereo mode. In particular, Figure 5.3-43 shows two frames of the stereo input signal, i.e. a DFT stereo frame followed by a TD stereo frame, at different processing operations with related time instances when switching from the DFT stereo mode to the TD stereo mode.

The instance A) of Figure 5.3-43 refers to the update of the FIR resampling filter memory (as employed in the FIR resampling from the input stereo signal sampling rate to the 12.8 kHz sampling rate and to the internal core-coder sampling rate) used in the primary channel $y(n)$ of the TD stereo coding mode. The stereo mode switching mechanism performs this update in every DFT stereo frame using the down-mixed mid-channel $m(n)$ and corresponds to a 2 x 0.9375 ms long segment before the last 7.5 ms long segment in the DFT stereo frame, thus ensuring continuity of the FIR resampling memory for the primary channel $y(n)$.

Since the side channel $s(n)$ of the DFT stereo encoder is not available though it is used at, for example, the 12.8 kHz sampling rate, at the input stereo signal sampling rate and at the internal sampling rate, the stereo mode switching mechanism populates the FIR resampling filter memory of the down-mixed secondary channel $x(n)$ differently. In order to reconstruct the full length of the down-mixed signal at the internal sampling rate for the core-coder, a 8.75 ms segment (see the top grey box in Figure 5.3-43) of the down-mixed signal of the previous frame is recomputed in the TD stereo frame. Thus, the update of the down-mixed secondary channel $x(n)$ FIR resampling filter memory corresponds to a 2 x 0.9375 ms long segment of the down-mixed mid-channel $m(n)$ before the last 8.75 ms long segment (the top grey box in Figure 5.3-43); this is done in the first TD stereo frame after switching from the preceding DFT stereo frame. The secondary channel $x(n)$ FIR resampling filter memory update is referred to by instance C) in

Figure 5.3-43. As can be seen, the stereo mode switching mechanism re-computes in the TD stereo frame a length (see the bottom grey box in Figure 5.3-43) of the down-mixed signal which is longer in the secondary channel $x(n)$ with respect to the recomputed length of the down-mixed signal in the primary channel $y(n)$ (see the middle grey box in Figure 5.3-43).

Instance B) in Figure 5.3-43 relates to updating (re-computation) of the primary $y(n)$ and secondary $x(n)$ channels in the first TD stereo frame following the DFT stereo frame. The operations of instance B) as performed by the stereo mode switching mechanism are illustrated in more detail in Figure 5.3-44.

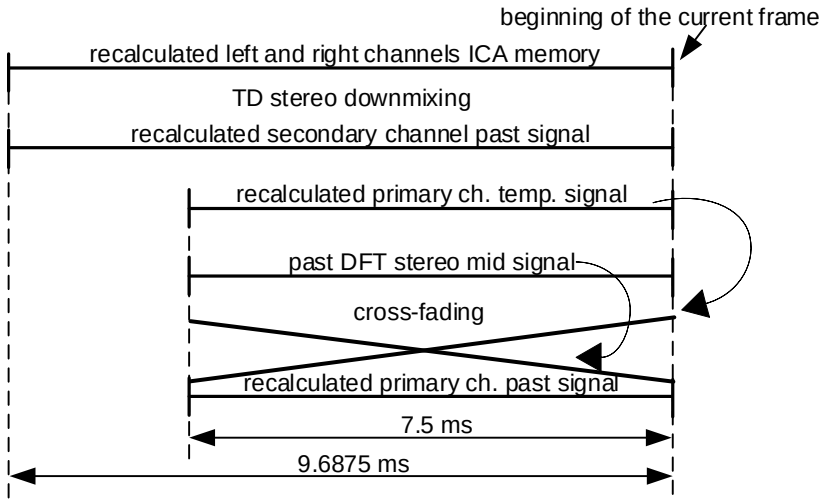


Figure 5.3-44: Flow chart illustrating operations related to TD stereo past signals upon switching from DFT stereo to TD stereo

Referring to Figure 5.3-44, the stereo mode switching mechanism recalculates the ICA memory as used in the ICA analysis and computation (see ICA operation in Figure 5.3-41) and later as input signal for the pre-processing and core-coders of length of 9.6875 ms of the channels $l(n)$ and $r(n)$ corresponding to the previous DFT stereo frame. Thus, the stereo mode switching mechanism recalculates the primary $y(n)$ and secondary $x(n)$ channels of the DFT stereo frame by down-mixing the ICA-processed channels $l(n)$ and $r(n)$ using a stereo mixing ratio β of that frame.

For the secondary channel $x(n)$, the length of the past segment to be recalculated by the stereo mode switching mechanism is 9.6875 ms (see Figure 5.3-44 although a segment of length of only 7.5 ms is recalculated when there is no stereo coding mode switching. For the primary channel $y(n)$, the length of the segment to be recalculated by the stereo mode switching mechanism using the TD stereo mixing ratio of the past frame $\beta^{[-1]}$ is always 7.5 ms. This ensures continuity of the primary $y(n)$ and secondary $x(n)$ channels.

A continuous down-mixed signal is employed when switching from mid-channel $m(n)$ of the DFT stereo frame to the primary channel $y(n)$ of the TD stereo frame. For that purpose, the stereo mode switching mechanism cross-fades the 7.5 ms long segment of the DFT mid-channel $m(n)$ with the recalculated primary channel $y(n)$ of the DFT stereo frame in order to smooth the transition and to equalize for different down-mix signal energy between the DFT stereo mode and the TD stereo mode as shown in Figure 5.3-44. The reconstruction of the secondary channel $x(n)$ uses the mixing ratio β of the DFT stereo frame while no further smoothing is applied because the secondary channel $x(n)$ from the DFT stereo frame is not available.

The core-coding in the first TD stereo frame following the DFT stereo frame then continues with resampling of the down-mixed signals using the FIR filters, pre-emphasizing these signals, computation of HB signals, etc.

With respect to the pre-emphasis filter (see clause 5.2.2.2.3), the stereo mode switching mechanism stores two values of the pre-emphasis filter memory in every DFT stereo frame. These memory values correspond to time instances based on different re-computation length of the DFT and TD stereo modes. This mechanism ensures an optimal re-computation of the pre-emphasis signal in the mid channel $m(n)$ of the DFT stereo or the primary channel $y(n)$ of the TD stereo, respectively, with a minimal signal length. For the secondary channel $x(n)$ of the TD stereo mode, the pre-emphasis filter memory is set to zero before the first TD stereo frame is processed.

Starting with the first TD stereo frame following the DFT stereo frame, certain DFT stereo related data structures (e.g. DFT stereo data structure) are not needed, so they are deallocated/freed by the stereo mode switching mechanism (see clause 5.3.4.3). On the other hand, a second instance of the core-coder data structure is allocated and initialized for the

core-coding of the secondary channel $x(n)$. The majority of the secondary channel $x(n)$ core-coder data structures are reset though some of them are estimated for smoother switching transitions. Specifically, the previous excitation buffer (adaptive codebook of the ACELP core), previous LSF parameters and LSP parameters of the secondary channel $x(n)$ are populated from their counterparts in the primary channel $y(n)$.

5.3.4.5 Switching between DFT and MDCT stereo modes

5.3.4.5.1 Switching from DFT stereo to MDCT stereo

A mechanism similar to the switching from the DFT stereo mode to the TD stereo mode as described above is used in this scenario, where the primary $y(n)$ and secondary $x(n)$ channels of the TD stereo mode are replaced by the left $l(n)$ and right $r(n)$ channels of the MDCT stereo mode.

The original stereo input in the MDCT overlap region, i.e. the last part of the current input frame, is stored during DFT stereo frames. In a following MDCT stereo transition frame the stored stereo input is linearly cross-faded with the previous downmix channel. The cross-fade is done over the full MDCT overlap region with a corresponding length of 8.75 ms. By fading out the downmix channel and fading in the original stereo channel a smooth MDCT overlap memory is created, thereby avoiding possible discontinuities in the transition frame.

Additionally, all required core coder state variables and buffers are copied over to the second channel in the transition frame.

5.3.4.5.2 Switching from MDCT stereo to DFT stereo

A mechanism similar to the switching from the TD stereo mode to the DFT stereo mode as described above is used in this scenario, where the primary $y(n)$ and secondary $y(n)$ channels of the TD stereo mode are replaced by the left $l(n)$ and right $r(n)$ channels of the MDCT stereo mode.

The memory for the DFT synthesis (as described in clause 5.3.2.4.11) is filled by copying the according signal part from the first channel of the previous MDCT stereo frame and applying the DFT window to it.

All DFT stereo related variables and buffers are reset.

5.3.4.6 Switching between TD and MDCT stereo modes

5.3.4.6.1 Switching from TD stereo to MDCT stereo

Switching from the TD stereo mode to the MDCT stereo mode is relatively straightforward because both these stereo modes handle two input channels and employ two core-coder instances. The main obstacle is to maintain the correct phase of the input left and right audio channels.

In order to maintain the correct phase of the input left and right channels of the stereo sound signal, the stereo mode switching mechanism alters the TD stereo down-mixing from clause 5.3.2.3.3. In the last TD stereo frame before the first MDCT stereo frame, the TD stereo mixing ratio is set to $\beta = 1.0$ and an opposite-phase down-mixing of the left and right channels of the stereo sound signal is implemented using the following formula for the TD stereo down-mixing:

$$y(n) = r(n) \cdot (1 - \beta) + l(n) \cdot \beta \quad (5.3\text{-}301)$$

$$x(n) = l(n) \cdot (1 - \beta) + r(n) \cdot \beta \quad (5.3\text{-}302)$$

where $y(n)$ is the TD stereo primary channel, $x(n)$ is the TD stereo secondary channel, $l(n)$ is the left channel, $r(n)$ is the right channel, β is the TD stereo mixing ratio, and $n = 0, \ldots, L - 1$ is the discrete time index while L is the frame length in samples. In turn, this means that the TD stereo primary channel $y(n)$ is identical to the MDCT stereo past left channel $l^{[-1]}(n)$ and the TD stereo secondary channel $x(n)$ is identical to the MDCT stereo past right channel $r^{[-1]}(n)$. For completeness, it is noted that the default TD stereo down-mixing uses the following formula from equation (5.3-70), i.e.

$$y(n) = r(n) \cdot (1 - \beta) + l(n) \cdot \beta \quad (5.3\text{-}303)$$

$$x(n) = l(n) \cdot (1 - \beta) - r(n) \cdot \beta \quad (5.3\text{-}304)$$

Next, in default (no stereo mode switching) MDCT stereo processing, the front pre-processing does not need to recompute the look-ahead of the left $l(n)$ and right $r(n)$ channels of the stereo signal except for its last 0.9375 ms long segment. However, in practice to ease the stereo mode switching, the look-ahead of the length of 7.5 + 0.9375 ms is subject to re-computation at the internal sampling rate. Thus, no specific handling is needed to maintain the continuity of input signals at the input sampling rate.

Then, in default (no stereo mode switching) MDCT stereo processing, the further pre-processing does not need to recompute the look-ahead of the left $l(n)$ and right $r(n)$ channels of the stereo signal except of its last 0.9375 ms long segment. In contrast with the front pre-processing, the input signals (left $l(n)$ and right $r(n)$ channels of the stereo sound signal) at the internal sampling rate of a length of only 0.9375 ms are recomputed in the further pre-processing module.

Consequently, the duration of the recomputed segment is different in the front and further pre-processing of the MDCT stereo encoder.

5.3.4.6.2 Switching from MDCT stereo to TD stereo

Similarly to the switching from the TD stereo mode to the MDCT stereo mode, two input channels are always available and two core-coder instances are always employed in this scenario. The main obstacle is again to maintain the correct phase of the input left and right channels. Thus, in the first TD stereo frame after the last MDCT stereo frame, the stereo mode switching mechanism sets the TD stereo mixing ratio to $\beta = 1.0$ and alters TD stereo down-mixing by using the opposite-phase mixing scheme similarly as described in clause 5.3.4.6.1.

Another particularity about the switching from the MDCT stereo mode to the TD stereo mode is that the stereo mode switching mechanism properly reconstructs in the first TD frame the past segment of input channels of the stereo signal at the internal sampling rate. Thus, a part of the look-ahead corresponding to 8.75 – 7.5 = 1.25 ms is reconstructed (resampled and pre-emphasized) in the first TD stereo frame.

5.3.5 DTX operation

5.3.5.1 DTX in Unified stereo

5.3.5.1.1 Signal activity detection in Unified stereo

The signal activity detection is run on the down-mix signal as described in 5.2.2.2.5. To aid in the stereo classification and selection between the TD-based stereo and DFT-based stereo, as well as activating the Stereo CNG mode, an additional signal activity detection is run on the input stereo signals coordinating the selection of encoding mode, see clause 5.3.2.2.2. Based on the encoding mode selected for each channel (as determined by VAD_0 and VAD_1), CNG encoding is applied in accordance with the joint VAD decision VAD_{01} as obtained from equation (5.3-41). If active encoding is selected for at least one of the channels (i.e., $VAD_0 = 1$ and/or $VAD_1 = 1$), active encoding is selected for both channels. However, if a speech pause is detected (i.e., $VAD_0 = 0$ and $VAD_1 = 0$), the DFT-based stereo mode is selected to be prepared to encode and transmit CNG frames. Within the DFT-based stereo mode, the IVAS core signal activity detector (see clause 5.2.2.2.5) is further run on the downmix signal, however this time without DTX hangover addition. Although the signal activity detection may not have been triggered for each channel separately, the combination of the two input channels may still trigger the signal activity detection.

In the case signal activity is detected in the downmix signal ($VAD_M = 1$), a VAD decision VAD_{DTX} determining whether to apply active encoding mode or CNG encoding mode within the stereo encoding is determined by:

$$VAD_{DTX} = VAD_{01} \vee VAD_M \tag{5.3-304a}$$

If $VAD_{DTX} = 1$, active encoding mode is applied. In the case signal activity is neither detected in the downmix signal, nor in the individual input signals, CNG encoding is applied. Figure 5.3-45 illustrates the signal activity detection logic.

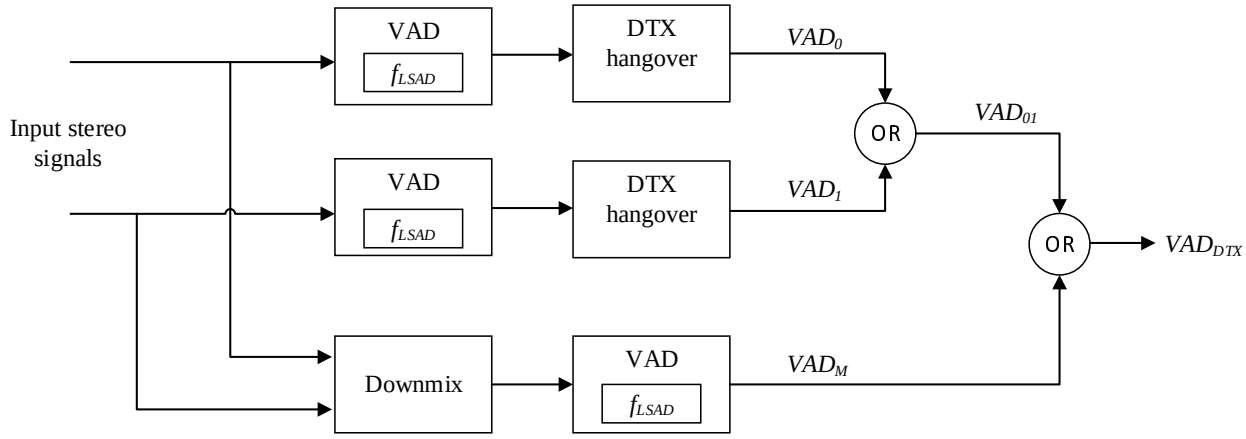


Figure 5.3-45: Signal activity detection in Unified stereo

The encoder is determined to be in DTX hangover mode if $VAD_{DTX} = 1$ and the local VAD flag $f_{LSAD} = 0$ for each of the front VAD channels and the downmix VAD. The local VAD f_{LSAD} represents an instantaneous VAD decision and is defined in clause 5.1.12.3 of [3].

5.3.5.1.2 General

The unified stereo mode supports Comfort Noise Generation (CNG) with a stereo CNG mode based on the DFT stereo mode. If the encoder selects a CNG frame to be encoded, the DFT-based stereo analysis is performed in the same way as described in 5.3.2.4. A down-mix signal $s_M(n)$ is prepared according to 5.3.2.4.10 and is input to the IVAS core encoder. The setting of the core bit rate indicates a CNG frame is to be encoded and transmitted to the decoder, generally following the EVS CNG operation as described in clause 5.6 of [3]. The spectral characteristics of the input audio channels are represented by the encoded down-mix CNG frame, in combination with the stereo parameters. There are however four differences in the down-mix CNG encoding compared to the EVS CNG encoding:

- Internal sampling frequency configuration
- LP-CNG high band analysis and quantization
- Core codec reset in active frame onset
- Energy scaling of CNG

An internal sampling frequency of 12.8 kHz is used for 13.2 kbps and 16.4 kbps LP-CNG WB, matching the internal sampling frequency of the active frame encoding. For all other configurations, 16 kHz internal sampling frequency is used.

The LP-CNG high band in Unified stereo CNG operating at SWB or FB bandwidth always uses SWB SID frames. WB SID frames are not used for SWB or FB operation.

The Unified stereo CNG decoder synthesizes the comfort noise in DFT domain and does not maintain the states of the core encoder and decoder. For this reason, the core encoder and decoder states are reset in the first active frame after an inactive segment.

The energy attenuation scheme of [3] 5.6 intends to match the level of the CNG with the level of the background noise when encoded with the active frame mode. For Unified Stereo CNG the same energy attenuation scheme is used both for LP-CNG and FD-CNG. This means that the energy offset selection for LP-CNG as described in clause 5.6.2.1.5 of [3] and Table 151a of [3] is not used. Instead, the gain scaling described in clause 5.6.2.3.5 of [3] is used to calculate the attenuation for both types of CNG. The attenuation values have been updated for Unified stereo CNG operation as shown in Table 5.3-21.

Table 5.3-21: Energy scaling values in Unified stereo CNG

Bandwidth	WB				SWB & FB			
Bitrate [kbps]	13.2	16.4	24.4	32	13.2	16.4	24.4	32
Scaling factor [dB]	-4	-3	-1.6	0.2	-2	-3	-0.8	-0.25

In Unified stereo CNG operation, the DFT-based stereo parameters $ITD(m)$, $gIPD(m)$, $g_S[m,b]$ are obtained and quantized in the same way as for active frames as described in 5.3.2.4. These parameters are used in the preparation of the down-mix $s_M(n)$. The resolution of the stereo parameters is however reduced for CNG operation. The side gain prediction parameters as described in 5.3.2.4.8 and the side gain prediction residual in the stereo decoder 0 are however not used in CNG operation. Instead, a coherence parameter $C_{01}[b,m]$ as described in (5.3-312) is derived and encoded.

While the $ITD(m)$ and $gIPD(m)$ are applied on the full frequency band, $g_S[m,b]$ is applied on a band structure. The band resolution is generally lower than in active coding and the same band resolution is used for $g_S[m,b]$ and $C_{01}[m,b]$. The band resolutions are listed in comparison to the active encoding band resolution in Table 5.3-22.

Table 5.3-22: Number of frequency bands in Unified stereo CNG

Bitrate [kbps]	WB		SWB		FB	
	Active	CNG	Active	CNG	Active	CNG
13.2	10	5	12	6	12	6
16.4	10	5	12	6	12	6
24.4	10	5	12	6	12	6
32	10	5	12	6	13	6

5.3.5.1.3 Stereo CNG side gain encoding

The side gain parameter $g_S[m,b]$ is averaged over the SID interval, i.e., side gain values calculated in the NO_DATA frames are included in the average to provide a more stable estimate for CNG frames. For the very first SID frame after starting the codec, or SID update frames during an inactive period, the average side gain $\tilde{g}_S[m,b]$ is defined according to

$$\tilde{g}_S[m,b] = \frac{\sum_{i=0}^{N_{curr}-1} g_{curr}[i,b]}{N_{curr}} \tag{5.3-305}$$

where N_{curr} denotes the number of frames in the current inactive segment including the current frame, $g_{curr}[i,b]$ is the side gain value for frame i of the current inactive segment and $b = 0,1,\ldots,N_{bands}-1$ is the band index corresponding to the band resolution of the active frames. For the very first SID frame, $N_{curr} = 1$ and the average corresponds to the current side gain. In the first SID frame of an inactive segment, excluding the very first SID frame since startup, the side gain average $\tilde{g}_S[m,b]$ is computed according to

$$\tilde{g}_S[m,b] = \frac{\sum_{i=0}^{N_{curr}-1} g_{curr}[i,b] + W(N_f)\sum_{j=0}^{N_{prev}-1} g_{prev}[j,b]}{N_{curr} + W(N_f)N_{prev}} \tag{5.3-306}$$

where $g_{prev}[j,b]$ is the side gain value for frame j of the previous inactive segment, N_{prev} is the number of frames in the previous inactive segment and $W(N_f)$ is a weighting function defined as

$$W(N_f) = \begin{cases} \frac{0.8(1500-N_f)}{1500} + 0.2, & N_f < 1500 \\ 0.2, & N_f \geq 1500 \end{cases} \tag{5.3-307}$$

where N_f is the number of frames in the active segment between the previous inactive segment and the current inactive segment. In this case N_{curr} corresponds to the hangover period and the first SID frame. If the number of bands used in the SID frames $N_{SIDbands}$ is lower than the number of bands in active frames N_{bands} as shown in Table 5.3-22, the band values are combined according to

$$g_{S,SID}[m,b] = \begin{cases} \frac{\tilde{g}_S[m,2b]BW[2b]+\tilde{g}_S[m,2b+1]BW[2b+1]}{BW[2b]+BW[2b+1]}, & 2b+1 < N_{bands} \\ \tilde{g}_S[m,2b], & otherwise \end{cases} \quad (5.3\text{-}308)$$

where $BW[b]$ is the bandwidth of band b as defined in Table 5.3-6, used in calculation of $\tilde{g}_S[m,b]$ and $b = 0,1,\ldots,N_{SIDbands}-1$. If the number of bands in active and inactive frames is the same, i.e. $N_{bands} = N_{SIDbands}$, the SID side gains are simply set to the averaged side gains $g_{S,SID}[m,b] = \tilde{g}_S[m,b]$. The side gain for the SID $g_{S,SID}[m,b]$ is encoded using the same encoding routine as for the active frames as described in clause 5.3.2.4.8.

5.3.5.1.4 Stereo CNG ITD and IPD encoding

The ITD value $ITD(m)$ is obtained as in 5.3.2.4.4 and the IPD value $gIPD(m)$ is obtained as in clause 5.3.2.4.7 and applied in the down-mix creation as described in 0 even before the decision is made to activate the Unified stereo CNG. In SID frames a coarser quantization step and narrower range is applied on the ITD in comparison to the active frames. The SID ITD values are calculated as follows,

$$ITD_{SID}(m) = \begin{cases} \max\left(15, \left\lfloor \frac{ITD(m)-256}{2} \right\rfloor\right), & ITD(m) > 255 \\ \max\left(15, \left\lfloor \frac{ITD(m)}{2} \right\rfloor\right), & ITD(m) \leq 255 \end{cases} \quad (5.3\text{-}309)$$

where $\lfloor \cdot \rfloor$ denotes a round-down or truncate operation. Additionally, a sign bit $ITD_{SID,sign}(m)$ is derived according to

$$ITD_{SID,sign}(m) = \begin{cases} 0, ITD(m) > 255 \\ 1, ITD(m) \leq 255 \end{cases} \quad (5.3\text{-}310)$$

The ITD parameters $ITD_{SID}(m)$ and $ITD_{SID,sign}(m)$ are encoded using the same scheme as the active ITD values, including a flag for non-zero ITD. If the non-zero ITD is set, only this flag is included in the bitstream. The resolution of the IPD value is reduced in a similar way in CNG operation. The index of the IPD value $gIPD_{SID,ind}(m)$ is calculated as

$$gIPD_{SID,ind}(m) = \begin{cases} I_{SID}, & I_{SID} < 4 \\ 0, & I_{SID} \geq 4 \end{cases}$$
$$I_{SID} = \left\lfloor \frac{2(gIPD(m)+\pi)}{\pi} + 0.5 \right\rfloor \quad (5.3\text{-}311)$$

Like the ITD parameter, the SID IPD index $gIPD_{SID,ind}(m)$ is included in the bitstream if the encoded IPD parameter non-zero flag is set.

5.3.5.1.5 Stereo CNG coherence encoding

The coherence $C_{band}[m,b]$ controls the perceived spatial width of the rendered comfort noise per band b in frame m. The range of $C_{band}[m,b]$ is $[0,1]$. The coherence $C_{01}(m,k)$ per bin k is determined as

$$C_{01}(m,k) = \frac{|S_{01,LP}(m,k)|^2}{P_{0,LP}(m,k)P_{1,LP}(m,k)} \quad (5.3\text{-}312)$$

where $S_{01,LP}(m,k)$ is the adaptively low-pass filtered cross-spectrum representing the cross-spectral density, as defined in (5.3-103) and $P_{0,LP}(m,k)$ and $P_{1,LP}(m,k)$ are the low-pass filtered power spectra, representing the auto-spectral density of channels 0 and 1 according to

$$\begin{cases} P_{0,LP}(m,k) = \beta|S_0(m,k)|^2 + (1-\beta)P_{0,LP}(m-1,k) \\ P_{1,LP}(m,k) = \beta|S_1(m,k)|^2 + (1-\beta)P_{1,LP}(m-1,k) \end{cases} \quad (5.3\text{-}313)$$

where $\beta = SFM$ is the adaptive low-pass filter coefficient defined as in (5.3-102). The coherence per band $C_{band}[m,b]$ is a weighted average of $C_{01}(m,k)$, using the power spectrum of the down-mix channel $S_M(m,k)$ as a perceptual weighting according to

$$C_{band}[m,b] = \frac{\sum_{k=k_{start}(b)}^{k_{start}(b+1)-1} C_{01}(m,k)|S_M(m,k)|^2}{\sum_{k=k_{start}(b)}^{k_{start}(b+1)-1} |S_M(m,k)|^2} \quad (5.3\text{-}314)$$

where $b = 0,1, \dots, N_{bands} - 1$, $k_{start}(b)$ is the start index of frequency band b and N_{bands} is the number of bands as defined in 5.3.2.4.6. For the first CNG frame in an inactive period, after being copied to $S_{01,LP}(m,k)$, $S_{01,LPCNG}(m,k)$ is reinitialized according to

$$S_{01,LPCNG}(m,k) \coloneqq \sqrt{C_{band}[m_{inactive}-2,b]P_{0,LP}(m,k)P_{1,LP}(m,k)}\frac{S_{01,LPCNG}(m,k)}{|S_{01,LPCNG}(m,k)|} \quad (5.3\text{-}315)$$

where the $C_{band}[m_{inactive}-2,b]$ denotes the coherence for band b for the second to last frame of the previous inactive period. $m_{inactive}$ denotes the frame index where $C_{band}[m,b]$ is updated, i.e. for inactive frames where signal activity is not detected.

The coherence $C_{band}[m,b]$ is encoded using a combination of intra-frame and inter-frame prediction and a residual encoding. An intra-frame prediction based on the target $C_{band}[m,b]$ is used to select a predictor index q according to

$$q = \underset{z}{\operatorname{argmin}} \sum_{b=0}^{N_{bands}-1} \left| C_{intra}^{(z)}[m,b] - C_{band}[m,b] \right|^2 \quad (5.3\text{-}316)$$

where the intra-frame prediction $C_{intra}^{(z)}[m,b]$ with predictor index z may be written as

$$C_{intra}^{(z)}[m,b] = \begin{cases} 0, & b = 0 \\ \sum_{i=0}^{b-1} p^{(z)}[b,i] C_{band}[m,b], & b = 1, \dots, N_{bands}-1 \end{cases} \quad (5.3\text{-}317)$$

where the intra-frame predictor $p^{(z)}[b,i]$ are a set of predictor coefficients that predicts the next coherence value $C_{band}[m,b]$ based on the previous band values. There are 4 different intra-frame predictors $p^{(z)}[b,i]$, $z = 0,1,2,3$. The selected intra-frame predictor q is encoded using 2 bits. The combined prediction $\hat{C}_{pred}[m,b]$, now using reconstructed values, can be written as

$$\hat{C}_{pred}[m,b] = \alpha \hat{C}_{intra}^{(q)}[m,b] + (1-\alpha)\hat{C}_{band}[m-1,b] \quad (5.3\text{-}318)$$

where the intra-frame prediction $\hat{C}_{intra}^{(q)}[m,b]$ now uses the selected predictor q and the previously reconstructed coherence values $\hat{C}_{band}[m,b]$.

$$\hat{C}_{intra}^{(q)}[m,b] = \begin{cases} 0, & b = 0 \\ \sum_{i=0}^{b-1} p^{(q)}[b,i] \hat{C}_{band}[m,b], & b = 1, \dots, N_{bands}-1 \end{cases} \quad (5.3\text{-}319)$$

The locally decoded coherence values $\hat{C}_{band}[m,b]$ are reconstructed in the encoder to match the decoder state as basis for the prediction. Further, α is a weighting factor that combines the intra-frame prediction $\hat{C}_{intra}^{(q)}[m,b]$ with the inter-frame prediction $\hat{C}_{band}[m-1,b]$ which corresponds to the coherence value for band b in the previous frame $m-1$. The weighting factor α is selected from two candidate weighting factors $\alpha_{low}, \alpha_{high}$ that depend on the available bit budget for coherence encoding B_C in the current frame m. The selection of the candidate weighting factors is implemented using a lookup table, as illustrated in Table 5.3-23.

Table 5.3-23: Candidate weighting factors based on coherence bit budget

Number of available bits	α_{low}	α_{high}
$B_C \leq 15$	0.1	0.6
$15 < B_C \leq 16$	0.1	0.6
$16 < B_C \leq 17$	0.1	0.7
$17 < B_C \leq 18$	0.1	0.9
$18 < B_C$	0.2	0.9

The principle of selecting the weighting factor α is to select the largest value which still fits within the bit budget B_C, since a larger α gives lower inter-frame prediction and hence reduces error propagation. A trial encoding is run with each weighting factor $\alpha_{low}, \alpha_{high}$ and a selection is done according to

$$\alpha = \begin{cases} \alpha_{high}, & B_{high} \leq B_C \\ \alpha_{low}, & B_{low} \leq B_C < B_{high} \\ \underset{\alpha}{\operatorname{argmin}}\big((B(\alpha)\big), \min\big(B_{low}, B_{high}\big) > B_C \end{cases} \quad (5.3\text{-}320)$$

where $B(\alpha)$ denotes the number of bits to encode the coherence vector using the weighting factor α such that $B_{low} = B(\alpha_{low})$ and $B_{high} = B(\alpha_{high})$. The bottom line of (5.3-320) may be read as if both α yields a number of bits that exceeds the coherence bit budget B_C, then the candidate α giving the lowest number of bits is selected. The selected candidate α is encoded using one bit. Note that this bit may be subtracted in the comparison with the bit budget B_C, in which case the conditions may be written as $B(\alpha) < B_C - 1$. A prediction error is formed by a prediction residual $C_{res}[m, b]$, calculated according to

$$C_{res}[m, b] = C_{band}[m, b] - \hat{C}_{pred}[m, b] \quad (5.3\text{-}321)$$

The prediction error is uniformly scalar quantized to 9 levels between -0.4 and 0.4 in steps of 0.1 and rearranged as in Table 5.3-24.

Table 5.3-24: Coherence prediction residual codebook

Index	Codeword	Value
0	0	0.0
1	10	0.1
2	110	-0.1
3	1110	0.2
4	11110	-0.2
5	111110	0.3
6	1111110	-0.3
7	11111110	0.4
8	111111110	-0.4

The residual $C_{res}[m, b]$ is quantized and encoded using a variable rate unary code in Table 5.3-24. If this is a trial encoding, the full number of bits is accumulated in $B(\alpha)$, but if it is the final encoding run the remaining number of bits is calculated from $B_C - B(\alpha)$ and the encoding is truncated if the bits run out, i.e. the length of the codeword for band b is larger than $B_C - B(\alpha)$. If the required number of bits to encode $C_{res}[m, b]$ for a certain band b, the index is reduced by 2 until it fits within the bit budget or reaches zero. If the bits run out, all the remaining residual indices are excluded from encoding and the remaining codewords are assumed to be zero. The truncated residual encoding can be described with the following pseudocode:

- $B(\alpha) \coloneqq 0$
- `For all bands` b
  - `Quantize and encode` $C_{res}[m, b]$ `to obtain index` I_{res} `with length` $I_{res} + 1$ `in bits`
  - `While` $I_{res} + 1 > B_C - B(\alpha)$ `and` $I_{res} > 0$
    - `Reduce index by two` $I_{res} \coloneqq \max(0, I_{res} - 2)$
  - `If` $I_{res} + 1 \leq B_C - B(\alpha)$
    - `Include` I_{res} `in encoded bits using codeword representation`
    - `Increment used bits` $B(\alpha) \coloneqq B(\alpha) + I_{res} + 1$

The reconstructed coherence values $\hat{C}_{band}[m, b]$ are formed according to

$$\hat{C}_{band}[m, b] = \alpha \hat{C}^{(q)}_{intra}[m, b] + (1 - \alpha)\hat{C}_{band}[m - 1, b] + \hat{C}_{res}[m, b] \quad (5.3\text{-}322)$$

where α now denotes the selected weighting factor. This reconstruction is done for each band and is used for encoding the next band. Note that this matches the reconstruction that the decoder performs, with the exception that $\hat{C}_{band}[m - 1, b]$ may diverge in case of transmission errors. If there are no bits at all in the bit budget for the stereo coherence coding, $B_C = 0$, then α and $\hat{C}_{res}[m, b]$ is set to zero and the previous coherence is used, i.e. $\hat{C}_{band}[m, b] = \hat{C}_{band}[m - 1, b]$. Note that due to the quantization of the prediction residual, the reconstructed value may fall outside of the valid range $[0,1]$. In case the reconstructed coherence falls outside of this range, it is clamped to the end points according to

$$\hat{C}_{band}[m, b] \coloneqq \begin{cases} 0, & \hat{C}_{band}[m, b] < 0 \\ 1, & \hat{C}_{band}[m, b] > 1 \\ \hat{C}_{band}[m, b], & otherwise \end{cases} \quad (5.3\text{-}323)$$

The encoded quantization indices of the intra-frame predictor, the selected weighting factor and the encoded prediction residual, in combination representing the spatial coherence for stereo CNG generation, are included in the bitstream to be sent to the decoder for stereo comfort noise generation. The reconstructed coherence values are formed according to

$$\hat{C}_{band}[m, b] = \alpha \hat{C}^{(q)}_{intra}[m, b] + (1 - \alpha)\hat{C}_{band}[m - 1, b] + \hat{C}_{res}[m, b] \quad (5.3\text{-}324)$$

where α now denotes the selected weighting factor. This reconstruction is done for each band and is used for encoding the next band. Note that this matches the reconstruction that the decoder performs, with the exception that $\hat{C}_{band}[m-1,b]$ may diverge in case of transmission errors. If there are no bit at all in the bit budget for the stereo coherence coding, $B_C = 0$, then α and $\hat{C}_{res}[m,b]$ is set to zero and the previous coherence is used, i.e. $\hat{C}_{band}[m,b] = \hat{C}_{band}[m-1,b]$.

5.3.5.2 DTX in MDCT-based stereo

5.3.5.2.1 Overview

DTX operation in MDCT-based stereo is based on the FD-CNG functionality from EVS. It is extended by encoding information about the coherence between the two channels in the SID frame and an efficient way of transmitting the parametric background noise description for both channels.

5.3.5.2.2 VAD in MDCT-based Stereo

The stereo signal is analyzed by a voice activity detector that determines a frame to be an inactive frame or an active frame. The activity detector analyzes the two channels separately to classify them as either active or inactive. Then, it determines the whole frame to be inactive if both the first channel and the second channel are classified as inactive. Otherwise, the frame is classified as active. The decision to classify a single channel as active or inactive is done the same as described in clause 5.2.2.2.5.

The VAD functionalities in each of the channels operate completely independent of each other. This also includes the mechanism for determining when to send the next SID frame in variable-DTX-update-interval operation (see clauses 5.6.1.1 and 5.6.1.2 of [3]). To prevent the two channel VADs to go out of synch, a synchronization mechanism is employed as depicted in Figure 5.3-46. If any of the two channel VADs decides on sending an SID in the current frame, the current frame will be an SID frame, even if the other channel's VAD decides that based on the other channel, the current frame would be a NO_DATA frame.

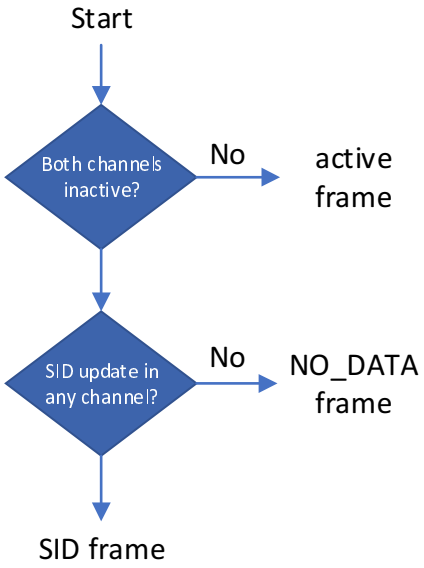


Figure 5.3-46: Frame type decision in MDCT-based Stereo DTX

5.3.5.2.3 Coherence estimation

A coherence calculator is run on the two input channel spectra in every frame to calculate a coherence value. In each of two subframes, four intermediate values (a real intermediate value, c_{real}, and an imaginary intermediate value, c_{imag}, as well as a first energy value for the first channel, e_L, and a second energy value for the second channel, e_R) are calculated as

$$c_{real} = \sum_{i=0}^{M-1} \Re\{L_i \times R_i^*\} \quad (5.3\text{-}325)$$

$$c_{imag} = \sum_{i=0}^{M-1} \Im\{L_i \times R_i^*\} \quad (5.3\text{-}326)$$

$$e_L = \langle L, L \rangle = \sum_{i=0}^{M-1} L_i \times L_i^* \quad (5.3\text{-}327)$$

$$e_R = \langle R, R \rangle = \sum_{i=0}^{M-1} R_i \times R_i^* \quad (5.3\text{-}328)$$

where *L* and *R* denote the subframe's FFT spectrum as calculated in clause 5.1.5 of [3]. The final coherence value is calculated using the real intermediate value, the imaginary intermediate value, the first energy value and the second energy value after smoothing all the intermediate values. The smoothed values are calculated as

$$\overline{c}_{real} = 0.95\, \overline{c}_{real,\,previous} + 0.05\, c_{real} \quad (5.3\text{-}329)$$

$$\overline{c}_{imag} = 0.95\, \overline{c}_{imag,\,previous} + 0.05\, c_{imag} \quad (5.3\text{-}330)$$

$$\overline{e}_L = 0.95\, \overline{e}_{L,\,previous} + 0.05\, e_L \quad (5.3\text{-}331)$$

$$\overline{e}_R = 0.95\, \overline{e}_{R,\,previous} + 0.05\, e_R \quad (5.3\text{-}332)$$

where $\overline{c}_{real,\,previous}$, $\overline{c}_{imag,\,previous}$, $\overline{e}_{L,\,previous}$ and $\overline{e}_{R,\,previous}$ denote the respective intermediate values as calculated for the previous frame. If the current frame is the first frame to be encoded or a rate switch has happened from unified stereo, all the previous-frame values are initialized to 10^{-15}. The coherence value is calculated using the smoothed values as

$$c = \sqrt{\frac{\overline{c}_{real}^2 + \overline{c}_{imag}^2}{\overline{e}_L \times \overline{e}_R}} \quad (5.3\text{-}333)$$

The coherence value is quantized as

$$c_{ind} = \min(15, \lfloor 15 \times c + 0.5 \rfloor) \in [0,15] \quad (5.3\text{-}334)$$

and encoded in the SID bitstream using four bits.

5.3.5.2.4 Noise parameter estimation and encoding

Information about the spectral shape of the background noise is derived by a noise parameter calculator that calculates first parametric noise data for the first channel of the stereo signal and second parametric noise data for the second channel of the stereo signal. The parameter calculator consists of the noise estimation algorithm as described in clause 5.6.3.2 of [3] which is run separately on the two channels of the stereo signal to generate two sets of parametric noise data - $N_{L,FD\text{-}CNG}$ for the left channel and $N_{R,FD\text{-}CNG}$ for the right channel. Additionally, the noise parameter calculator is configured to convert the first parametric noise data and second parametric noise data from a left/right representation to a mid/side representation. This conversion is applied after converting the parametric noise data values to dB as

$$N_{L,FD-CNG}^{dB}(i) = 10 \log_{10}\left(N_{L,FD-CNG}(i) + 0.0001\right) \quad (5.3\text{-}335)$$

$$N_{R,FD-CNG}^{dB}(i) = 10 \log_{10}\left(N_{R,FD-CNG}(i) + 0.0001\right) \quad (5.3\text{-}336)$$

From these, parametric noise data in a mid/side representation are calculated:

$$N_{M,FD-CNG}^{dB} = 0.5\left(N_{L,FD-CNG}^{dB}(i) + N_{R,FD-CNg}^{dB}(i)\right) \quad (5.3\text{-}337)$$

$$N_{S,FD-CNG}^{dB} = 0.5\left(N_{L,FD-CNG}^{dB}(i) - N_{R,FD-CNg}^{dB}(i)\right) \quad (5.3\text{-}338)$$

This parametric noise data is encoded using the MSVQ described in clause 5.6.3.5 of [3]with a modified first stage. For encoding the mid noise data, all 6 stages of the modified MSVQ are used while for the side noise data only the first four are used. The modified first stage MSVQ operation for FD-CNG is described in clause 5.2.2.3.5.2 .

The output of the MSVQ decoder is denoted as $N^{SID}_{M,FD\text{-}CNG}(i)$ and $N^{SID}_{S,FD\text{-}CNG}(i)$, respectively. The MSVQ output is reconverted from the mid/side representation back into a left/right representation. If the energy of the unquantized side noise data $N^{dB}_{S,FD\text{-}CNG}$ is smaller than 0.1, all values of $N^{SID}_{S,FD\text{-}CNG}$ are set to zero prior to reconverting to the left/right representation. A flag (the "no-side" flag) is encoded in the SID bitstream to signal this to the decoder. If the energy of the unquantized side noise data $N_{S,FD\text{-}CNG}$ is smaller than 0.1, the no-side flag is one, otherwise it is zero. The value is

encoded using a single bit. Thus, the reconverted parametric noise data for each channel in left/right representation is calculated as

$$N_{L,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 1, \\ N_{M,FD-CNG}^{SID}(i) + N_{S,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 0. \end{cases} \quad (5.3\text{-}339)$$

$$N_{R,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 1, \\ N_{M,FD-CNG}^{SID}(i) - N_{S,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 0. \end{cases} \quad (5.3\text{-}340)$$

From the reconverted left/right representation of the parametric noise data, a first global gain value for the first channel and a second global gain value for the second channel are calculated using equation (1399) of [3] with $N_{FD-CNG}^{dB} = N_{L,FD-CNG}^{dB}$ and $\bar{N}_{FD-CNG}^{SID} = N_{L,FD-CNG}^{SID}$ for the left channel and $N_{FD-CNG}^{dB} = N_{R,FD-CNG}^{dB}$ and $\bar{N}_{FD-CNG}^{SID} = N_{R,FD-CNG}^{SID}$ for the right channel. The global gain values for both channels are then quantized according to equation (5.2-240).

To align the SID frame size to that of other modes, the SID bitstream is padded with 16 zeros after completing the previously described encoding steps.

5.4 Scene-based audio (SBA) operation

5.4.1 SBA format overview

IVAS supports coding of ambisonics up to order 3 at bitrates 13.2 kbps to 512 kbps. At bitrates below 32 kbps, the codec supports coding up to SWB bandwidth whereas for bitrates at or above 32 kbps, the codec supports coding up to FB bandwidth. An optional support for planar SBA coding is also provided which limits the coding to only planar SBA channels. IVAS expects SBA input to be SN3D (see [23]) normalized and follow the ACN (see [24]) ordering convention.

The main technologies in the SBA format coding are Spatial Reconstruction (SPAR) and Directional Audio Coding (DirAC) which will be described in detail in further clauses.

5.4.2 Combined DirAC and SPAR based SBA coding

5.4.2.1 Combined DirAC and SPAR based SBA coding overview

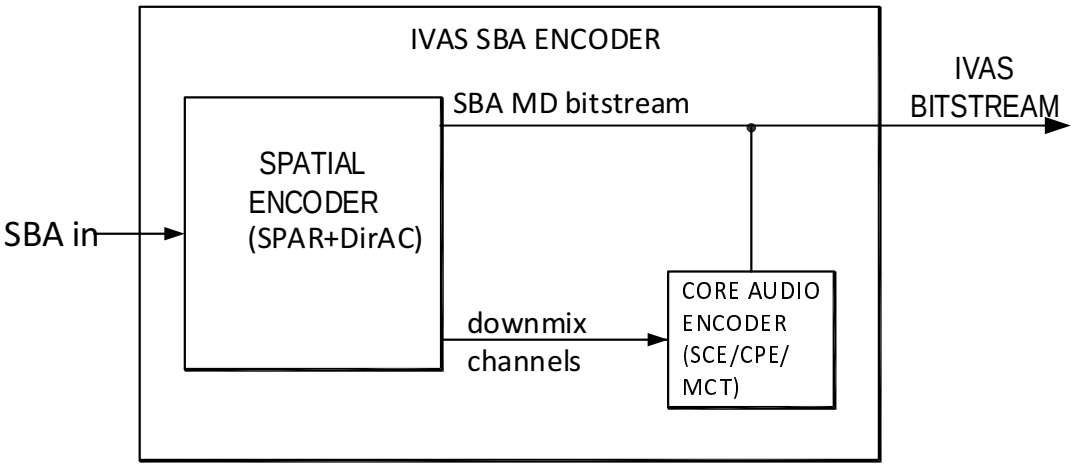


Figure 5.4-1: SBA coding architecture in IVAS

Figure 5.4-1 is a block diagram of SBA coding architecture. In the SBA encoder, though up to 16 SBA input channels may be present at the encoder input, depending on encoder setting, only a smaller number N_{inp} of channels may be subject to analysis and further coding. The relevant set of N_{inp} SBA input channels is high pass filtered (see clause 5.2.1) yielding SBA signal $s^{HP20}(n), n$ being the sample index. This signal is analysed by the SBA spatial encoder block which outputs an SBA metadata (MD) bitstream and N_{dmx} downmix channels. The SBA spatial encoder block

uses a combination of SPAR and DirAC spatial analysis methods (as described in clause 5.4.3) to perform parameter estimation and downmixing.

SPAR seeks to maximize perceived audio quality while minimizing bitrate by reducing the energy of the transmitted audio data while still allowing the second-order statistics of the Ambisonics audio scene (i.e., the covariance) to be reconstructed at the decoder side using transmitted metadata. SPAR seeks to faithfully reconstruct the input Ambisonics scene at the output of the decoder.

The input ambisonics signal is processed with the MDFT analysis filterbank (see clause 5.2.5) prior to the parameter estimation. SPAR and DirAC parameters are computed in the banded domain, as per the frequency banding in the MDFT filterbank (described in clause 5.2.5). These parameters are then converted into a banded downmix matrix (described in clause 5.4.4). The SBA spatial encoder block converts the N_{inp} channel SBA signal $s^{HP20}(n)$ into an N_{dmx} channel downmix signal $s^{DMX}(n)$, such that $N_{dmx} \leq N_{inp}$, using the banded downmix matrix. Downmixing occurs in filterbank domain using the MDFT filterbank and the downmixing process is described in clause 5.4.5. The downmix signal $s^{DMX}(n)$ is coded using the core coding tools described in clause 5.2.3. The core-coded bitstream and MD bitstream are multiplexed to generate the IVAS bitstream in SBA coding mode. Table 5.4-1 shows the mapping between IVAS bitrate in SBA format, number of downmix channels (N_{dmx}) and core coding tool that is used to code the downmix audio signal $s^{DMX}(n)$. The downmix audio signal $s^{DMX}(n)$ consists of a primary downmix channel W' and zero or more residual channels depending on the value of N_{dmx}.

The downmix matrix for the high frequency bands is estimated based on sound field parameter(s). Inter-channel prediction is applied in the downmix based on an inter-channel covariance matrix. This covariance matrix is derived from the sound field parameters. These include a DoA and a global diffuseness parameter.

The SBA MD analysis audio signal and downmix audio signal consists of channels that are selected based on perceptual relevance. Perceptual relevance is ranked based on the following principles:

- First-order channels are ranked more relevant than higher-order channels.
- For a given order *l*,
  - Channels corresponding to a spherical harmonic Y_{lm} (θ, φ) having a larger overlap with the left-right-front-rear plane are ranked more relevant than channels having a larger overlap with the height direction.
  - Channels corresponding to spherical harmonic Y_{lm} (θ, φ) having a larger overlap with the left-right direction are ranked more relevant than channels having a larger overlap with the front-rear direction.
- Higher-order planar channels of the same order, where planar channels for a given order l are the channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$, are ranked more relevant than the non-planar channels of the same order. Moreover, the planar channels of order *l* are ranked more relevant than the non-planar channels of order *l*-1.

Table 5.4-1: Mapping between IVAS bitrate, number of downmix channels and core coding tools

Bitrate [kbps]	Number of downmix channels	Core-coding tool
13.2	1	SCE
16.4	1	SCE
24.4	1	SCE
32	1	SCE
48	2	CPE
64	2	CPE
80	2	CPE
96	3	MCT
128	3	MCT
160	3	MCT
192	3	MCT
256	4	MCT
384	4	MCT
512	4	MCT

The number of downmix channels in SBA coding mode are limited to a maximum number of 4 even if the input is Higher order Ambisonics. This threshold is set based on bitrate limitation, metadata size, core-coder performance and overall audio quality.

5.4.2.2 FOA signal coding at all bitrates and HOA2, HOA3 signal coding at bitrates below 256 kbps

5.4.2.2.1 Overview

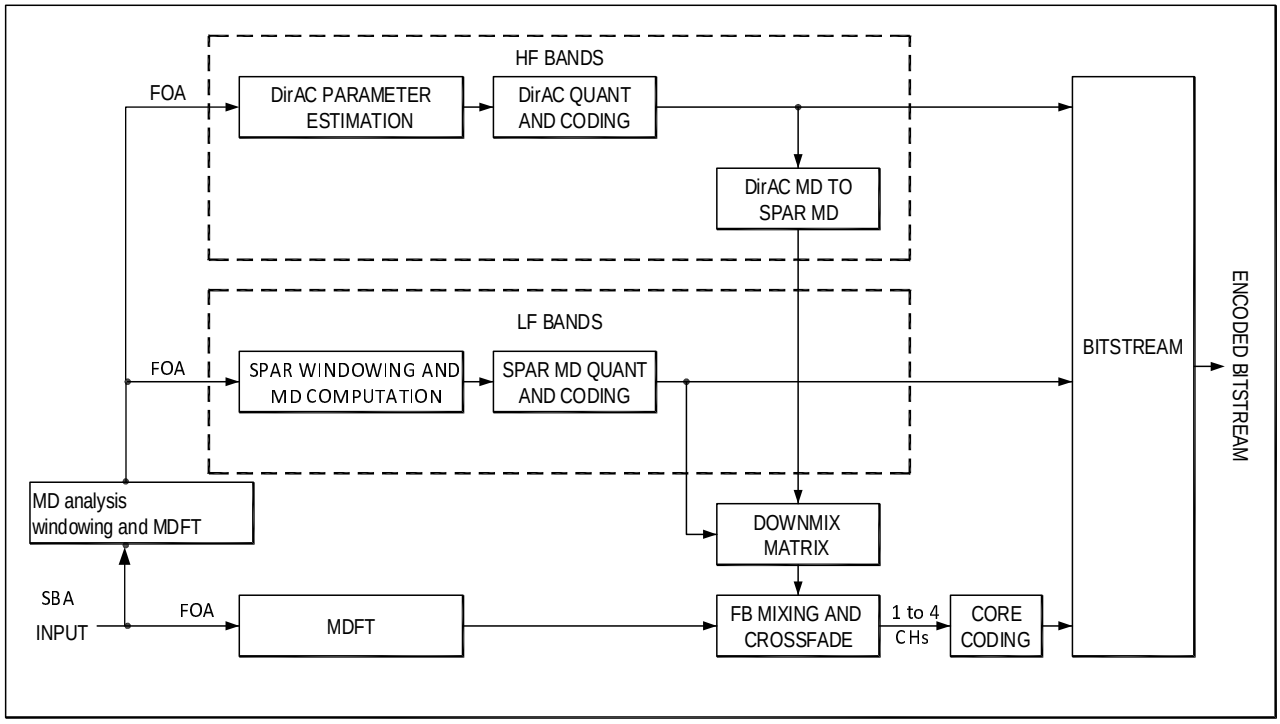


Figure 5.4-2: FOA signal coding at all bitrates and HOA2, HOA3 signal coding at 256 kbps

Figure 5.4-2 shows the block diagram of SBA coding architecture with FOA input signal at all bitrates and HOA2, HOA3 signal coding at bitrates below 256 kbps. It has two major components, that is, Metadata (MD) parameter analysis and downmixing and core coding. Metadata (MD) parameter analysis includes MD analysis windowing and MDFT, DirAC parameter estimation, DirAC quantization and coding, SPAR MD computation, SPAR MD quantization and coding, DirAC MD to SPAR MD conversion and generation of downmix matrix. Downmixing and core coding includes MDFT on FOA signal, filterbank mixing and crossfading to generate downmix signal and core coding of downmix signals. The signal chain for Figure 5.4-2 is described below.

5.4.2.2.2 Metadata parameter analysis

The FOA component of $s^{HP20}(n)$ is windowed using the metadata analysis window (as described in clause 5.4.3.3) and transformed to a frequency domain signal $S_i'^{MDFT}$, where i is the channel index such that $1 \leq i \leq N_{sba\_ana}$ and $N_{sba\_ana} = 4$, using MDFT (as described in clause 5.4.3.3). The high frequency bins, in the frequency range f^{DirAC}, such that $f^{DirAC} > 4.4$ kHz, of $S_i'^{MDFT}$ are processed by DirAC parameter estimation block (described in clause 5.4.3.4) which generates DirAC MD parameters in the f^{DirAC} frequency range, quantizes and codes these parameters into DirAC MD bitstream. The low frequency bins, in the frequency range f^{SPAR}, such that $f^{SPAR} \leq 4.4$ kHz, of $S_i'^{MDFT}$ are processed by SPAR parameter estimation block (described in clause 5.4.3.7) which generates SPAR MD parameters in the f^{SPAR} frequency range, quantizes and codes these parameters into SPAR MD bitstream. The DirAC MD parameters in f^{DirAC} frequency range are converted into SPAR parameters, including prediction, cross-prediction and decorrelation coefficients, MD using the DirAC MD to SPAR MD converter (described in clause 5.4.3.8.2). SPAR MD in f^{SPAR} and f^{DirAC} frequency ranges is combined and is converted into a downmix matrix $DMX_{[N_{dmx}\ x\ N_{sba\_ana}]}$, as described in clause 5.4.4.

5.4.2.2.3 Downmixing and core-coding

The N_{sba_ana} channel SBA analysis signal component of $s^{HP20}(n)$ is windowed using a 40 ms window, as described in clause 5.4.5.1, and transformed into a frequency domain signal S_i^{MDFT}, where $1 \leq i \leq N_{sba\_ana}$ is the channel index, using the MDFT. The frequency-domain signal S_i^{MDFT} is then processed by the filter bank (FB) mixing and crossfade block which generates the downmix signal $s_i^{DMX}(n), 1 \leq i \leq N_{dmx}$, by applying the downmix matrix $DMX_{[N_{dmx}\ x\ N_{sba\_ana}]}$ to S_i^{MDFT} in the filter bank domain. The downmix generation process is explained in detail in clause 5.4.5. The downmix signal $s_i^{DMX}(n)$ is coded with either SCE (clause 5.2.3.2), CPE (clause 5.2.3.3) or MCT (clause 5.2.3.4) core-coding tool as per Table 5.4-1. IVAS bitstream is generated with coded SPAR metadata, coded DirAC metadata and coded downmix signal.

5.4.2.3 HOA2 and HOA3 signal coding overview at 256 kbps

5.4.2.3.1 Overview

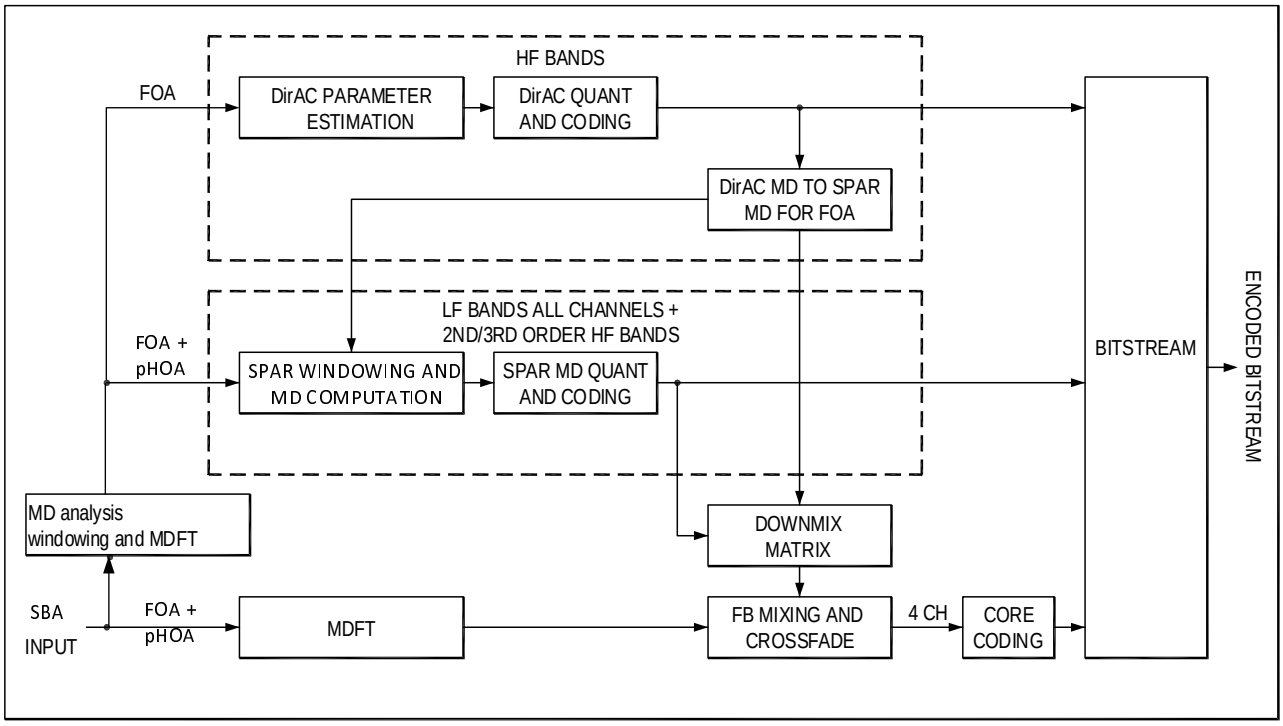


Figure 5.4-3: HOA2 and HOA3 signal coding at 256 kbps

Figure 5.4-3 shows the block diagram of SBA coding architecture with HOA2 and HOA3 signal coding at 256 kbps.

5.4.2.3.2 Metadata parameter analysis

The FOA and planar HOA components of $s^{HP20}(n)$ are windowed using the metadata analysis window (see clause 5.4.3.3) and transformed to a frequency domain signal S'^{MDFT}_i using the MDFT, where i is the channel index such that $1 \leq i \leq N_{sba\_ana}$ and N_{sba_ana} is the number of FOA + planar HOA channels in the input SBA signal. $N_{sba\_ana} = 6$ for HOA2 input and $N_{sba\_ana} = 8$ for HOA3 input. Herein, planar channels for a given ambisonics order l are defined as channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$. The high frequency bins in the frequency range $f^{DirAC} >$ 4.4 kHz of the FOA component of S'^{MDFT}_i are processed by the DirAC parameter estimation block (described in clause 5.4.3.4), which generates DirAC MD parameters in the f^{DirAC} frequency range, quantizes and codes these parameters into DirAC MD bitstream. All frequency bins of S'^{MDFT}_i are processed by the SPAR parameter estimation block (described in clause 5.4.3.7) which generates FOA SPAR MD parameters in the frequency range $f^{SPAR} \leq$ 4.4 kHz, HOA SPAR parameters in both f^{SPAR} and f^{DirAC} frequency ranges, quantizes and codes these parameters into SPAR MD bitstream. The DirAC MD parameters in f^{DirAC} frequency range are converted into FOA SPAR parameters using the DirAC MD to SPAR MD converter (described in clause 5.4.3.8.2). SPAR MD, corresponding to FOA and HOA channels, in f^{SPAR} and f^{DirAC} frequency ranges is combined and is converted into a downmix matrix $DMX_{[N_{dmx}\ x\ N_{sba\_ana}]}$ as described in clause 5.4.4.

5.4.2.3.3 Downmixing and core-coding

Same as described in clause 5.4.2.2.3.

5.4.2.4 HOA2 and HOA3 signal coding overview at 384 kbps

5.4.2.4.1 Overview

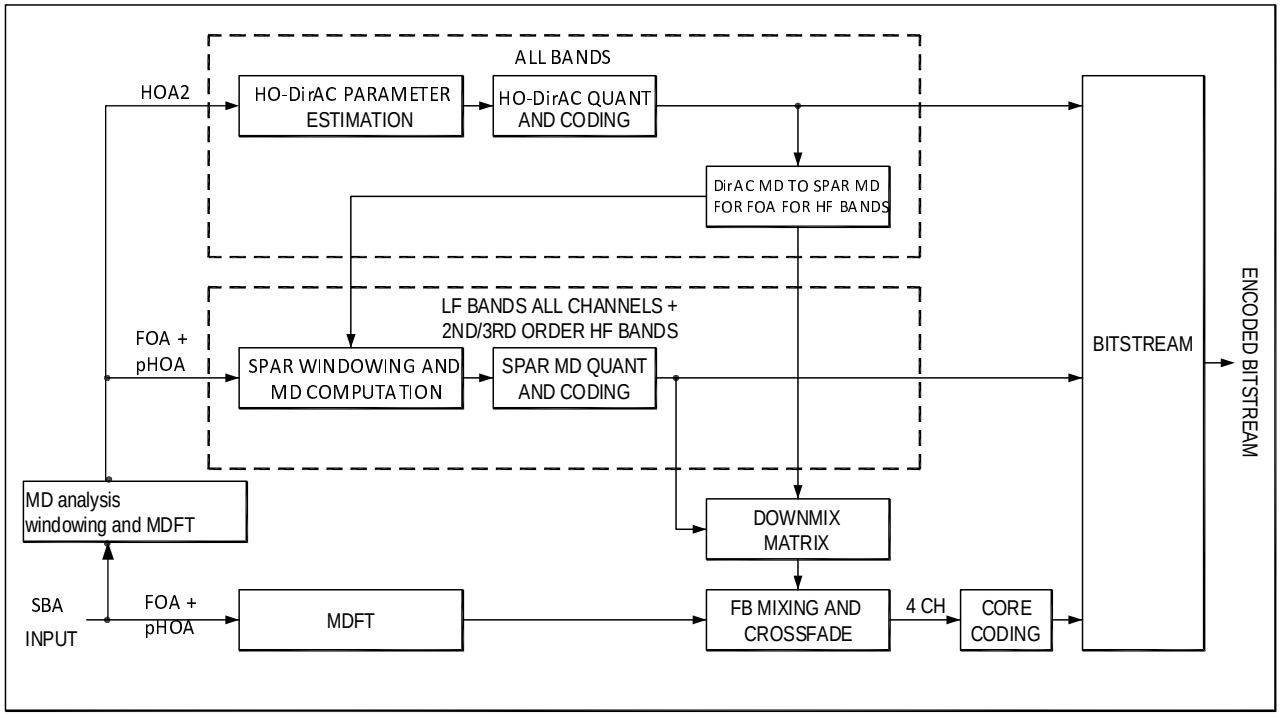


Figure 5.4-4: HOA2 and HOA3 signal coding at 384 kbps

Figure 5.4-4 shows the block diagram of SBA coding architecture with HOA2 and HOA3 signal coding at 384 kbps.

In this mode, SPAR analysis is done on FOA and planar HOA channels and HO-DirAC analysis is done on HOA2 channels. The DirAC to SPAR MD conversion is limited to FOA channels in this mode. The signal chain for Figure 5.4-4.

5.4.2.4.2 Metadata parameter generation

The HOA2 and planar HOA3 components of $s^{HP20}(n)$ are windowed using the metadata analysis window and transformed to a frequency domain signal $S_i'^{MDFT}$ using the MDFT, where i is the channel index such that $1 \leq i \leq N_{sba\_ana}$, and N_{sba_ana} is the number of HOA2 + planar HOA3 channels in the input SBA signal. The number of analysis channels $N_{sba\_ana} = 9$ for HOA2 input and $N_{sba\_ana} = 11$ for HOA3 input. Herein, planar channels for a given ambisonics order l are defined as channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$. All frequency bins in the HOA2 component of $S_i'^{MDFT}$ signal is processed by DirAC parameter estimator block which generates DirAC MD parameters, quantizes and codes these parameters into DirAC MD bitstream. All frequency bins of the FOA and planar HOA component of $S_i'^{MDFT}$ are processed by SPAR parameter estimator block which generates FOA SPAR MD parameters in the f^{SPAR} frequency range ($f^{SPAR} \leq 4.4$ kHz), HOA SPAR parameters in both f^{SPAR} and f^{DirAC} frequency ranges, quantizes and codes these parameters into SPAR MD bitstream. The DirAC MD parameters in f^{DirAC} frequency range are converted into FOA SPAR parameters MD using DirAC MD to SPAR MD converter. SPAR MD, corresponding to FOA and HOA channels, in f^{SPAR} and f^{DirAC} frequency ranges is combined and is converted into a downmix matrix $DMX_{[N_{dmx}\ x\ N_{spar\_ana}]}$, where N_{spar_ana} is the number of FOA + planar HOA channels in the input SBA signal, $N_{spar\_ana} = 6$ for HOA2 input and $N_{spar\_ana} = 8$ for HOA3 input.

5.4.2.4.3 Downmixing and core-coding

Same as described in clause 5.4.2.2.3.

5.4.2.5 HOA2 and HOA3 signal coding overview at 512 kbps

5.4.2.5.1 Overview

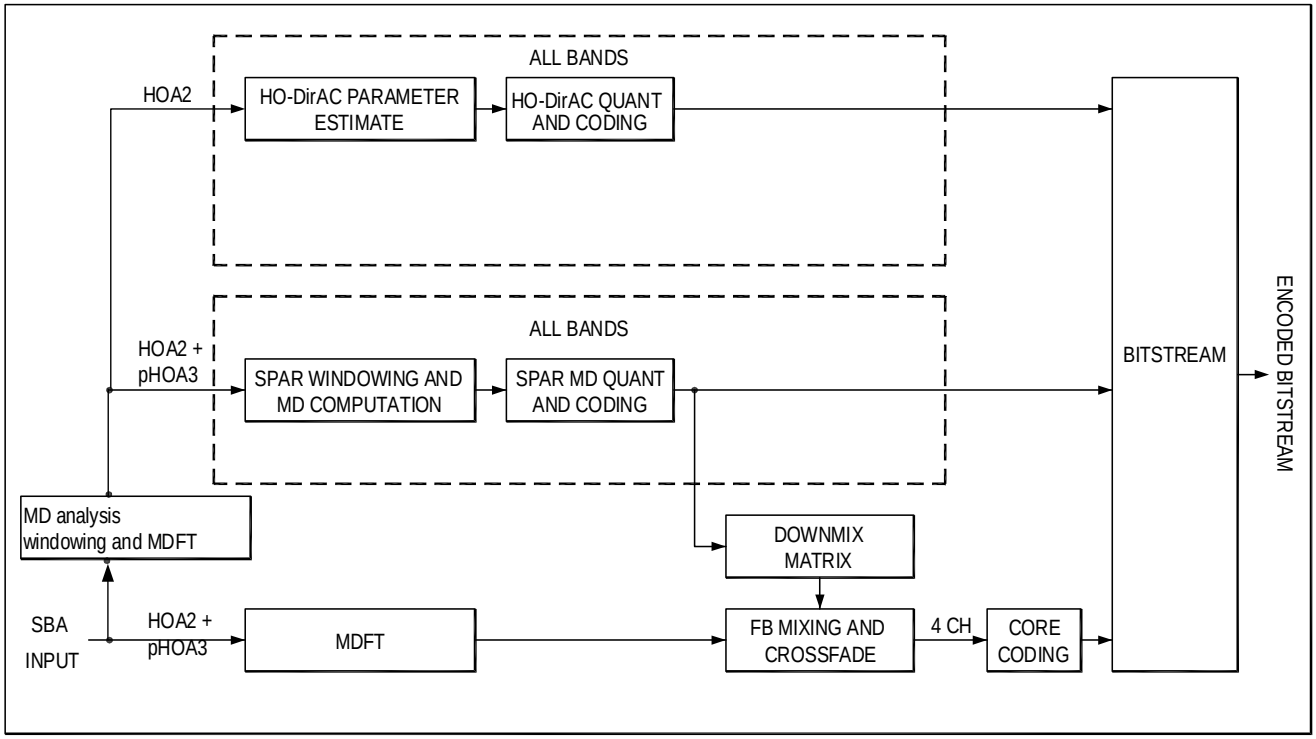


Figure 5.4-5: HOA2 and HOA3 signal coding at 512 kbps

Figure 5.4-5 shows the block diagram of SBA coding architecture with HOA2 and HOA3 signal coding at 512 kbps. In this mode, SPAR analysis is done on HOA2 and planar HOA3 channels and HO-DirAC analysis is done on HOA2 channels. The DirAC to SPAR MD conversion is not done in this mode. The signal chain is described below.

5.4.2.5.2 Metadata parameter generation

The HOA2 and planar HOA3 components of $s^{HP20}(n)$ are windowed using the metadata analysis and transformed to a frequency domain signal $S_i'^{MDFT}$ using the MDFT, where i is the channel index such that $1 \leq i \leq N_{sba\_ana}$, and N_{sba_ana} is the number of HOA2 + planar HOA3 channels in the input SBA signal, $N_{sba\_ana} = 9$ for HOA2 input and $N_{sba\_ana} = 11$ for HOA3 input. Herein, planar channels for a given ambisonics order l are defined as channels corresponding to spherical harmonics $Y_l^m(\theta, \varphi)$ for a given order l with $|m| = l$. All frequency bins in the HOA2 component of $S_i'^{MDFT}$ signal is processed by DirAC parameter estimator block which generates DirAC MD parameters, quantizes and codes these parameters into DirAC MD bitstream. All frequency bins of $S_i'^{MDFT}$ are processed by SPAR parameter estimator block which generates SPAR MD parameters in both f^{SPAR} and f^{DirAC} frequency ranges, quantizes and codes these parameters into SPAR MD bitstream. DirAC MD to SPAR MD conversion is not needed at this bitrate as SPAR parameters corresponding to all analysis channels and entire frequency spectrum are generated by SPAR parameter estimator block. SPAR MD is then converted into a downmix matrix $DMX_{[N_{dmx}\ x\ N_{sba\_ana}]}$.

5.4.2.5.3 Downmixing and core-coding

Same as described in clause 5.4.2.2.3.

5.4.3 SBA parameter estimation

5.4.3.1 SBA front VAD

The SBA front VAD sets three flags W_{vad}, W_{DTXvad} and $W_{ForceFrontVad}$ based on the IVAS bitrate and IVAS DTX mode. These flags are used during metadata computation and downmix coding. If DTX is OFF or if IVAS bitrate is greater than 80 kbps then the VAD flags are set as follows.

$$W_{vad} = 1$$

$$W_{DTXvad} = 0$$

$$W_{ForceFrontVad} = 0$$

When DTX is ON and IVAS bitrate is less than or equal to 80 kbps, then the W channel is delayed by the delay of MDFT filterbank, that is 1ms, and the W_{vad} flag is computed by doing the signal activity detection on the delayed W channel as for the IVAS core, see clause 5.2.2.2.5, including DTX hangover addition as described in clause 5.1.12.8 of [3].

The signal activity detection is done based on a core coding instance that is independent of the core coding instances created to encode the downmix channels. The value of W_{DTXvad} and $W_{ForceFrontVad}$ flags are set as follows.

$$W_{ForceFrontVad} = \begin{cases} 1, & N_{dmx} = 1 \\ 0, & N_{dmx} = 2 \end{cases}$$

$$W_{DTXvad} = \begin{cases} 1, & N_{dmx} = 1, core_{brate} \leq 2400 \\ & 0, otherwise \end{cases}$$

Where $core_{brate} \leq 2400$ indicates SID or NO_DATA frame as per the signal activity detection and DTX hangover addition.

In 1 channel downmix cases, the VAD flag and DTX hangover addition decision of the SCE are overwritten with W_{vad} and W_{DTXvad} flags respectively. Whereas, in 2 channel downmix cases, the VAD flag decision of the CPE is overwritten by W_{vad} flag only if W_{vad} is set to 1.

5.4.3.2 Transient Detection

Transient detection makes use of the transient processor that is part of the time-domain decorrelator described in clause 6.2.1.3. The transient processor outputs a time-varying envelope $e_{in,k}[n]$ for the current audio frame k of length N that is used by the decorrelator to separate the transient and continuous components of the input signal. From this envelope, transients are detected by comparing changes in this envelope against a fixed threshold. Two boolean signals for transient detection, T_0 and T_1 are defined, operating on different time scales. T_0 is computed as follows:

$$\mathrm{T}_0 = \begin{cases} 1, & e_{in,k-1}[N-1] - e_{in,k}[N-1] > 0.1 \\ 0, & \text{otherwise} \end{cases} \quad (5.4\text{-}1)$$

At bitrates below 24.4 kbps, $T_1$is always set to 0. To compute T_1 at higher bitrates, the frame k is subdivided into 16 subframes $m = 0 \,..\, 15$, such that $e_{in,k,m}[n] = e_{in,k}[\frac{Nm}{16} + n]$. T_1 is then computed as follows:

$$\mathrm{T}_1 = \begin{cases} 1, & e_{in,k,m-1}\left[\frac{N}{16} - 1\right] - e_{in,k,m}[\frac{N}{16} - 1] > 0.11, m \in \{1 \,..\, 15\} \\ 1, & e_{in,k-1,15}\left[\frac{N}{16} - 1\right] - e_{in,k,0}\left[\frac{N}{16} - 1\right] > 0.11 \\ 0, & \text{otherwise} \end{cases} \quad (5.4\text{-}2)$$

The main difference between these two transient detection signals is that T_0 samples the transient processor envelope once per 20 ms frame whilst T_1 samples the envelope 16 times per frame (every 1.25 ms). Due to the shorter sampling interval, T_1 is less sensitive to weak transients than T_0.

5.4.3.3 Analysis windowing and MDFT

The SBA metadata analysis signal $s_j^{sba\_ana}(n)$, where $1 \leq j \leq N_{sba\_ana}$ and N_{sba_ana} is the total number of SBA analysis channels, is formed by selecting a subset of channels from $s_i^{HP20}(n)$ signal, where $1 \leq i \leq$

$(ambisonics_{order} + 1)^2$. Table 5.4-2 shows the subset of channel indices i that forms the signal $s_j^{sba\_ana}(n)$ and the subset of channel indices i that are used in DirAC and SPAR metadata analysis.

Table 5.4-2: SBA metadata analysis channels at various IVAS bitrates

IVAS bitrate [kbps]	Input ambisonics order	SBA analysis order	SBA Metadata analysis channel indices	DirAC Metadata analysis channel indices	SPAR Metadata analysis channel indices
13.2 – 192	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	3	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
256	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	2	1, 2, 3, 4, 5, 9	1, 2, 3, 4	1, 2, 3, 4, 5, 9
	3	3	1, 2, 3, 4, 5, 9, 10, 16	1, 2, 3, 4	1, 2, 3, 4, 5, 9, 10, 16
384	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	2	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 9
	3	3	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 9, 10, 16
512	1	1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
	2	2	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9
	3	3	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16

Every frame of $s_j^{sba\_ana}(n)$ is divided into 4 subframes and each subframe is windowed with a sinusoidal window. Figure 5.4-6 shows the subframe windowing and MDFT. The input to MDFT is a 10 ms long time domain signal $s_j^{sba\_ana,mdft}(m,n)$, where m is the subframe index with $1 \le m \le 4$ and n is the sample index with $1 \le n \le \frac{10\ samplingfrequency}{1000}$, that is formed by concatenating two consecutive subframes of $s_j^{sba\_ana}$ and then applying the sine squared window function as shown in Equations (5.4-3), (5.4-4) and (5.4-5).

$$s_j^{sba\_ana,mdft}(m,n) = s'^{sbs\_ana}_j(m,n) Win^{sba\_ana}(n), \tag{5.4-3}$$

$$s'^{sba\_ana}_j(m,n) = \begin{cases} 0, & 1 \le n \le (L_{sf} - L_{prev} + L_{offset}) \\ s_j^{sba\_ana}(m-1, n - L_{offset}), & (L_{sf} - L_{prev} + L_{offset}) < n \le (L_{sf} + L_{offset}) \\ s_j^{sba\_ana}(m, n - L_{sf} - L_{offset}), & (L_{sf} + L_{offset}) < n \le 2L_{sf} \end{cases}, \tag{5.4-4}$$

$$Win^{sba\_ana}(n) = \begin{cases} 0, & 1 \le n \le (L_{sf} - L_{prev} + L_{offset}) \\ \sin^2\left(\frac{n-(L_{sf}-L_{prev}+L_{offset})}{L_{fade}} - \frac{1}{2L_{fade}}\right), & (L_{sf} - L_{prev} + L_{offset}) < n \le (L_{sf} - L_{prev} + L_{offset} + L_{fade}) \\ 1, & (L_{sf} - L_{prev} + L_{offset} + L_{fade}) < n \le (2L_{sf} - L_{fade}) \\ 1 - \sin^2\left(\frac{n-(2L_{sf}-L_{fade})}{L_{fade}} - \frac{1}{2L_{fade}}\right), & (2L_{sf} - L_{fade}) < n \le 2L_{sf} \end{cases} \tag{5.4-5}$$

where $L_{frame} = 20L_{ms}$, $L_{sf} = 5L_{ms}$, $L_{prev} = 1.5L_{ms}$, $L_{curr} = 4.5L_{ms}$, $L_{fade} = L_{ms}$, $L_{offset} = 0.5L_{ms}$, $L_{ms} = \frac{samplingfrequency}{1000}$, m is the subframe index with $1 \le m \le 4$ and $s_j^{sba\_ana}(0,n)$ is the 4th subframe of previous frame.

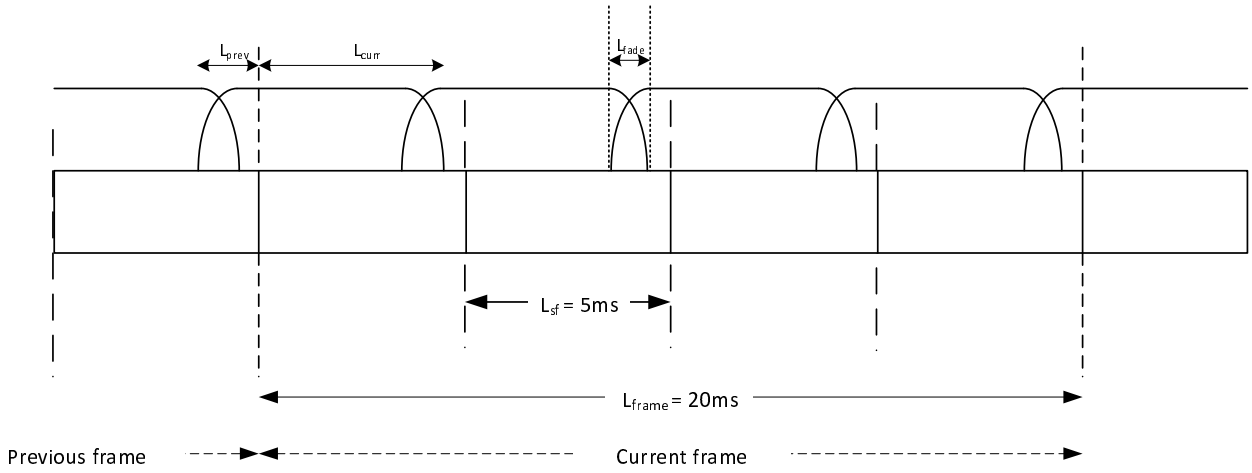


Figure 5.4-6: Subframe windowing and MDFT

For each m, $s_j^{sba\_ana,mdft}(m,n)$ is converted to frequency domain signal $S_j^{sba\_ana,mdft}(m,k)$ by applying the MDFT (as described in clause 5.2.5.2) to $s_j^{sba\_ana,mdft}(m,n)$, where m is the subframe index and k is the frequency bin index with $1 \leq k \leq \frac{L_{frame}}{4}$.

5.4.3.4 DirAC parameter estimation

5.4.3.4.1 General

On the encoder side, psycho-acoustic sound field parameters (DirAC) are estimated from the ambisonics analysis signal $\boldsymbol{s}^{sba\_ana,mdft}(m,k)$ in the time-frequency (TF) domain. These parameters comprise one or more direction(s) of arrival (DoA) and one or more diffuseness parameters per parameter band.

The time resolution depends on the type of parameter: for DoAs a time resolution of 5 ms is employed, while for the diffuseness parameters a time resolution of 20 ms is used instead. The frequency resolution is defined by 12 parameter bands (see Table 5.4-5).

Using these parameters, the ambisonics input signal $\boldsymbol{S}(n)$ (of order 1, 2 or 3) is encoded into the compressed representation comprising audio downmix channels and side information. The audio downmix channels, $\boldsymbol{s}^{DMX}(n)$, the first set of transport channels as decoded in clause 6.4.6.1, are generated using the SPAR downmix according to clause 5.4.5 based on the SPAR and DirAC metadata.

The encoder unit supports a high-order and a low-order operation mode. It can switch between these modes dynamically based on the bitrate. For the bitrates 384 and 512 kbps (cf. clause 5.4.3.4.2), the high-order operation mode is enabled. At all other bitrates the low-order operation mode is active.

The difference between these modes is the number of spatial sectors in the DirAC parameter estimation. In the high-order mode, both sector parameter estimation paths are activated. In the low-order operation mode, one of them is deactivated. The other one processes the input signal with an omni-directional sector filter to cover the sound field on the whole sphere.

The high-order mode requires an input signal of at least order 2. Order 3 can be configured as input, but the DirAC encoder does not evaluate the channels of the third order. If a lower input order is configured, the encoder falls back to the low-order mode, even at the two aforementioned high bitrates. This SBA analysis order is transmitted to the decoder in the bitstream to configure the correct decoding method.

5.4.3.4.2 High-order operation mode

The side information derived by the encoder includes sound field parameters representing the scene described by the second- or third-order ambisonics input signal. These sound field parameters comprise a DoA and a sector diffuseness parameter for each of the two spatial sectors.

For the estimation of the parameters, for each spatial sector, the sector signal is generated by means of spatial filtering (segmentation) of the input ambisonic signal $\boldsymbol{x}_{HOA}\ (m,k)$ (see Figure 5.4-7):

$$\boldsymbol{x_S}(m,k) = \boldsymbol{w}_s^T\ \boldsymbol{s}^{sba\_ana,mdft}(m,k) \tag{5.4-6}$$

where $\boldsymbol{x_S}(m,k)$ is the vector of the spatially filtered (segmental) first-order ambisonics signal in sector s. $\boldsymbol{w}_s^T$ is the matrix projecting the vector on the sector beamforming weights with the entries according to Table 5.4-3 for sector 0 and Table 5.4-4 for sector 1, respectively. It can be viewed as the projector onto the beams describing the sector $\boldsymbol{b}_s$. The beams $\boldsymbol{b}_s$ describe the directivity pattern of the spatial filter that belongs to the sector s.

Table 5.4-3: **Elements of the spatial filter matrix** w_0^T

	w (0,0)	x (1, -1)	y (1,0)	z (1,1)	(2, -2)	(2, -1)	(2,0)	(2,1)	(2,2)
w (0,0)	1.772454	1.772454	0	0	0	0	0	0	0
x (1, -1)	0	0	1.772454	0	1.023326	0	0	0	0
y (1,0)	0.590818	1.772454	0	0	0	0	- 0.590817	0	- 1.023326
z (1,1)	0	0	0	1.772454	0	1.023326	0	0	0

Table 5.4-4: **Elements of the spatial filter matrix** w_1^T

	w (0,0)	x (1, -1)	y (1,0)	z (1,1)	(2, -2)	(2, -1)	(2,0)	(2,1)	(2,2)
w (0,0)	1.772454	-1.772454	0	0	0	0	0	0	0
x (1, -1)	0	0	1.772454	0	-1.023326	0	0	0	0
y (1,0)	-0.590818	1.772454	0	0	0	0	0.590817	0	1.023326
z (1,1)	0	0	0	1.772454	0	-1.023326	0	0	0

To take full advantage of the sparse structure of the matrix $\boldsymbol{w}_s^T$ for complexity reduction, in the reference implementation, the matrix multiplication is explicitly written out as the individual multiplications and additions with the non-zero matrix elements.

Each of these sector signals is then fed into a sector parameter estimator (parameter generator, see Figure 5.4-7), which derives the parameters for the sector, a sector DoA and a sector diffuseness.

The DoA is derived based on the pseudo-intensity vector, whose components are given by

$$\mathrm{I}_x\ (m,j) = \sum_{k=0}^{N_{\mathrm{MDFT}}} {w_k}^j\ \left\{\mathrm{Re}\left[\mathrm{x}_{s,w}(m,k)\right] * \mathrm{Re}\left[\mathrm{x}_{\mathrm{s,x}}(m,k)\right] + \mathrm{Im}\left[\mathrm{x}_{s,w}(m,k)\ \right] * \mathrm{Im}\left[\mathrm{x}_{\mathrm{s,x}}(m,k)\right]\right\}, \tag{5.4-7}$$

$$\mathrm{I}_\mathrm{y}\ (m,j) = \sum_{k=0}^{N_{\mathrm{MDFT}}} {w_k}^j\ \left\{\mathrm{Re}\left[\mathrm{x}_{s,w}(m,k)\right] * \mathrm{Re}\left[\mathrm{x}_{\mathrm{s,y}}(m,k)\right] + \mathrm{Im}\left[\mathrm{x}_{s,w}\ (m,k)\right] * \mathrm{Im}\left[\mathrm{x}^i_{\mathrm{s,y}}(m,k)\right]\right\}, \tag{5.4-8}$$

$$\mathrm{I}_\mathrm{z}\ (m,j) = \sum_{k=0}^{N_{\mathrm{MDFT}}} {w_k}^j\ \ \left\{\mathrm{Re}\left[\mathrm{x}_{s,w}(m,k)\right] * \mathrm{Re}\left[\mathrm{x}^i_{\mathrm{s,z}}(m,k)\right] + \mathrm{Im}\left[\mathrm{x}_{s,w}(m,k)\ \right] * \mathrm{Im}\left[\mathrm{x}_{\mathrm{s,z}}(m,k)\right]\right\} \tag{5.4-9}$$

The $\mathrm{x}_{w;x;y;z}(m,k)$ are the w;x;y;z components of the sector signal $\boldsymbol{x}_s(m,k)$ in the MDFT bin with index k. The contributions of the MDFT bins are weighted by the numbers w_k^j to achieve a grouping of the bins into parameter bands. w_k^j is the weight of the bin k in the group j. This results in one intensity vector for each parameter band indexed by j.

The choice of the groups is perceptually motivated. The frequency resolution of the model parameters decreases at higher frequencies. The choice of the groups is aligned with the grouping of the CLDFB bands on the decoder side and the frequency responses of the bands of the MDFT-based filter bank in the signal path (cf. 5.4.5.1).

The groups are defined according to Table 5.4-5.

Table 5.4-5: **MDFT start bins of the parameter bands**

Group index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Start bin	0	4	8	12	16	20	24	28	44	68	100	240

The weights w_k^j are non-zero only where the bin index k falls into the range of the group j. As each bin falls into exactly one group, there is exactly one weight to store in a table for each bin. The values are listed in Table 5.4-6.

Table 5.4-6: **Weight w_k^j of each MDFT bin k in the correspoonding parameter band j**

9.962447e-02 9.627997e-01 9.926667e-01 9.981028e-01 9.996648e-01 1.000000e+00 9.997692e-01 9.992002e-01 9.983890e-01

9.973818e-01 9.962037e-01 9.948692e-01 9.933876e-01 9.917654e-01 9.900073e-01 9.881169e-01 9.860975e-01 9.839516e-01

9.816818e-01 9.792906e-01 9.767801e-01 9.741527e-01 9.714106e-01 9.685560e-01 9.655913e-01 9.625187e-01 9.593406e-01

9.560594e-01 9.526774e-01 9.491970e-01 9.456208e-01 9.419512e-01 9.381908e-01 9.343420e-01 9.304075e-01 9.263898e-01

9.222915e-01 9.181152e-01 9.138636e-01 9.095392e-01 9.051447e-01 9.006827e-01 8.961559e-01 8.915668e-01 8.869181e-01

8.822123e-01 8.774521e-01 8.726400e-01 8.677785e-01 8.628702e-01 8.579176e-01 8.529231e-01 8.478893e-01 8.428184e-01

8.377130e-01 8.325753e-01 8.274077e-01 8.222124e-01 8.169917e-01 8.117478e-01 8.064829e-01 8.011990e-01 7.958982e-01

7.905827e-01 7.852543e-01 7.799150e-01 7.745667e-01 7.692112e-01 7.638505e-01 7.584861e-01 7.531199e-01 7.477535e-01

7.423885e-01 7.370265e-01 7.316691e-01 7.263176e-01 7.209736e-01 7.156384e-01 7.103134e-01 7.049999e-01 6.996990e-01

6.944121e-01 6.891403e-01 6.838847e-01 6.786464e-01 6.734265e-01 6.682258e-01 6.630455e-01 6.578863e-01 6.527492e-01

6.476350e-01 6.425445e-01 6.374784e-01 6.324376e-01 6.274226e-01 6.224341e-01 6.174729e-01 6.125393e-01 6.076341e-01

6.027577e-01 5.979106e-01 5.930932e-01 5.883061e-01 5.835497e-01 5.788242e-01 5.741301e-01 5.694676e-01 5.648372e-01

5.602390e-01 5.556734e-01 5.511404e-01 5.466405e-01 5.421737e-01 5.377402e-01 5.333402e-01 5.289738e-01 5.246411e-01

5.203422e-01 5.160771e-01 5.118460e-01 5.076489e-01 5.034858e-01 4.993567e-01 4.952616e-01 4.912005e-01 4.871734e-01

4.831802e-01 4.792209e-01 4.752955e-01 4.714037e-01 4.675457e-01 4.637212e-01 4.599302e-01 4.561725e-01 4.524481e-01

4.487567e-01 4.450983e-01 4.414728e-01 4.378799e-01 4.343195e-01 4.307915e-01 4.272956e-01 4.238318e-01 4.203997e-01

4.169993e-01 4.136303e-01 4.102926e-01 4.069859e-01 4.037101e-01 4.004649e-01 3.972501e-01 3.940655e-01 3.909109e-01

3.877861e-01 3.846909e-01 3.816250e-01 3.785882e-01 3.755803e-01 3.726010e-01 3.696501e-01 3.667275e-01 3.638328e-01

3.609658e-01 3.581263e-01 3.553141e-01 3.525289e-01 3.497705e-01 3.470387e-01 3.443331e-01 3.416537e-01 3.390001e-01

3.363720e-01 3.337694e-01 3.311919e-01 3.286393e-01 3.261114e-01 3.236079e-01 3.211286e-01 3.186733e-01 3.162418e-01

3.138337e-01 3.114490e-01 3.090872e-01 3.067484e-01 3.044321e-01 3.021382e-01 2.998664e-01 2.976166e-01 2.953885e-01

2.931819e-01 2.909966e-01 2.888323e-01 2.866889e-01 2.845661e-01 2.824637e-01 2.803816e-01 2.783194e-01 2.762770e-01

2.742543e-01 2.722509e-01 2.702667e-01 2.683014e-01 2.663550e-01 2.644271e-01 2.625177e-01 2.606264e-01 2.587531e-01

2.568977e-01 2.550599e-01 2.532395e-01 2.514364e-01 2.496503e-01 2.478811e-01 2.461287e-01 2.443928e-01 2.426732e-01

2.409698e-01 2.392824e-01 2.376109e-01 2.359550e-01 2.343146e-01 2.326895e-01 2.310797e-01 2.294848e-01 2.279047e-01

2.263394e-01 2.247886e-01 2.232521e-01 2.217299e-01 2.202217e-01 2.187274e-01 2.172469e-01 2.157800e-01 2.143266e-01

2.128865e-01 2.114596e-01 2.100457e-01 2.086447e-01 2.072564e-01 2.058808e-01

The fast-varying vector components in Equations (5.4-7) to (5.4-9) are then smoothened with an averaging filter with an update rate of $\beta = 0.2$ to obtain the smooth versions $\bar{I}_{w;x;y;z}^j$. The application of the inverse trigonometric function then yields the DoA angles azimuth

$$\phi^j = \arctan\left(\frac{\bar{I}_y^j}{\bar{I}_w}\right), \tag{5.4-10}$$

and elevation

$$\theta^j = \arcsin\left(\frac{\bar{I}_z^j}{\|\boldsymbol{I}+\epsilon\|}\right). \tag{5.4-11}$$

The sector diffuseness is then given by

$$\Psi = 1 - \left\| \frac{\boldsymbol{I}}{\bar{E}+\epsilon} \right\|, \tag{5.4-12}$$

where ϵ is a small parameter for numerical stability and $\bar{E}$ is the smoothened version of the energy defined as

$$E = \frac{1}{2}(\boldsymbol{x}_s \cdot \boldsymbol{x}_s^*) \tag{5.4-13}$$

The asterisk * denotes the complex conjugate and · denotes the inner product of the vectors. If E is less than ϵ in one subframe, then the azimuth and elevation angles are set to zero and the diffuseness Ψ is set to 1. The subframe and parameter-band indices have been omitted from the notation here.

This calculation is repeated in each sector parameter estimator to obtain a sector DoA and a sector diffuseness parameter.

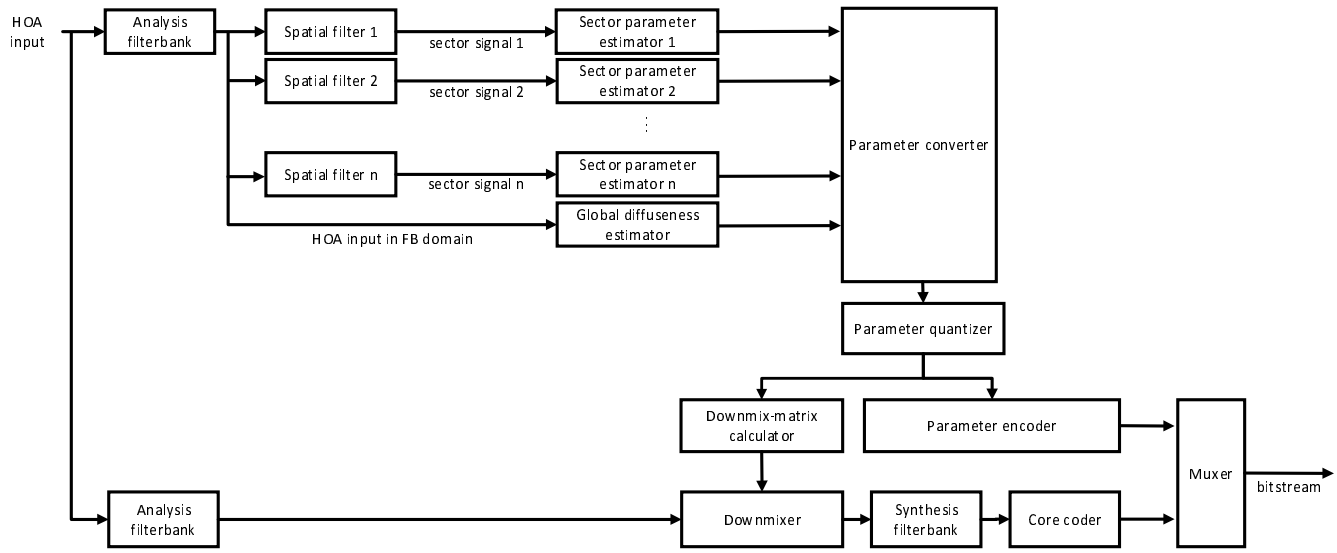


Figure 5.4-7: SBA encoder block diagram in high-order operation mode

In addition to the sector parameter estimators operating on the spatially filtered sector signals $\boldsymbol{x}_s$, there is a global diffuseness path. The global diffuseness estimator operates on the spatially unfiltered first-order input channels and returns a global diffuseness parameter Ψ. This global diffuseness parameter indicates how diffuse the whole audio scene is. It takes values between 0 and 1, where 0 means that the scene has no diffuse energy and 1 means that the scene has exclusively diffuse energy. Therefore, the general nature of the scene is always captured correctly, irrespective of the number of sector paths activated.

The smoothing of the intensity vector $\boldsymbol{I}$ and the Energy E are realized by storing the values of the last 40 ms and averaging over them. Moreover, and for reducing the number of parameters, the global diffuseness computed in each parameter band and for each time slot of 5 ms is averaged along the time axis to achieve a transmitted global diffuseness parameter of 20 ms. Therefore, the time resolution is decoupled between the global diffuseness parameter and the DoAs, exploiting the fact that diffuseness retains a longer-term characteristic of the sound field than the directions, which is are more reactive spatial cue.

This global diffuseness parameter is then quantized and encoded into the bitstream according to clause 5.2.4. In the reference implementation, the global diffuseness estimator is the same code as for the low-order mode.

The global diffuseness parameter is computed as:

$$\bar{\Psi}\ \ (b) = \frac{\sum_{n=0}^{3} \Psi(\mathrm{m}, \mathrm{b}).\, E(m,b)}{\sum_{n=0}^{3} E(m,b)}$$

where b is parameter band index, m is the subframe index. It corresponds to the stride index of the MDFT corresponding and the block index in the spatial metadata coding according to clause 5.2.4.

The sector diffuseness parameters are converted to relative directionalities of each of the two spatial sectors with respect to both spatial sectors by a parameter converter (cf. Figure Figure 5.4-7). These relative directionalities are indicative of the relative diffuseness of one sector to all other sectors and are encoded and written to the bitstream as parameters.

The relative directionalities of the two spatial sectors are defined as

$$a_1 = \frac{1-\Psi_1}{(1-\Psi_1)+(1-\Psi_2)} \tag{5.4-14}$$

and

$$a_2 = 1 - a_1, \tag{5.4-15}$$

where Ψ_1 is the sector diffuseness parameter for the first spatial sector and Ψ_2 is, the sector diffuseness parameter for the second spatial sector. The frame and parameter band indices have been omitted from the notation here.

This is a special case of the general formula

$$a_i = \frac{1-\Psi_i}{\sum_j (1-\Psi_j)} \tag{5.4-16}$$

with

$$\sum_j a_j = 1 \tag{5.4-17}$$

for more than two sectors. It is important to note that the relative directionalities always add up to 1. The relative directionality of a spatial sector indicates the ratio of the directional energy in that sector to the sum of the directional energies of all sectors. The relative directionalities are quantized and encoded in the bitstream (cf. clause 5.2.4).

5.4.3.4.3 Low-order operation mode

In the low-order operation mode, only one sector parameter estimator path is activated, and the other is deactivated. The spatial filtering according to Equation (5.4-6) becomes trivial as the matrix $\boldsymbol{w}_0^T$ is the unit matrix. The activated sector parameter estimator effectively operates on the first order ambisonics component signals in the vector $\boldsymbol{s}^{sba\_ana,mdft}$.

The global diffuseness and sector diffuseness are the same quantity in this case. Hence, only the global diffuseness estimator runs.

The DoA estimation in this single-sector case is modified as compared to the high-order mode. The smoothing of the intensity vector is omitted. The azimuth and elevation angles are calculated as

$$\phi^b = \operatorname{atan}\left(\frac{I_y}{I_x}\right),$$

and

$$\theta^b = \operatorname{atan}\left(\frac{I_z}{\sqrt{I_x^2+I_y^2}}\right),$$

respectively.

In the reference code, the DoA calculation is, for efficiency and simplicity, partly implemented in different functions than for the high-order operation modes. In this implementation, the smoothing of the intensity vector is achieved by storing the values of the last 40 ms and averaging over them.

The DirAC parameter estimation runs only for the 4 highest parameters bands (bin groups) according to Table 5.4-5. The bins of the two highest bands are grouped together into one parameter band, effectively encoding 3 parameter bands. The DirAC parameters for the lower parameter bands are estimated for the upmixed signal on the decoder side.

For the two lowest bitrates, 13.2 and 16.4 kbps, the time resolution of the direction parameters is reduced to one frame (20 ms). This is achieved by a weighted average of the intensity vector over all subframes of the frame. The weights are given by the associated directional energy. The frame-wise direction vector in cartesian coordinates is computed as:

$$\boldsymbol{r}_{DoA}(\mathrm{b}) = \frac{\sum_{n=0}^{3}(1-\Psi(\mathrm{n},\mathrm{b})).E(n,b).\frac{\mathrm{I}^b(n)}{\|\mathrm{I}^b(n)\|}}{\sum_{n=0}^{3}(1-\Psi(\mathrm{n},\mathrm{b})).E(n,b)}$$

where $E(n, b)$ and $\Psi(\mathrm{n}, \mathrm{b})$ are the energy and the diffuseness, respectively, of the input signal measured in the parameter band b and in the time block n, and $\mathrm{I}^{b}(n)$ is the intensity vector in three-dimensional Cartesian coordinates for the same parameter band *b* and time block *n*. After averaging, the resulting directional vector $\boldsymbol{r}_{DoA}^{b}$ can no longer be a unit vector and normalization is therefore necessary:

$$\boldsymbol{r}_{DoA}(\mathrm{b}) \leftarrow \frac{\boldsymbol{r}_{DoA}(b)}{\left\|\boldsymbol{r}_{DoA}(b)\right\|}$$

5.4.3.5 Hybrid encoder

With only one sector encoding path enabled, the encoding of the audio scene is achieved in a more efficient way employing a hybrid encoding approach. Specifically, the encoding scheme of the parametric spatial audio-scene encoder (DirAC and SPAR) is aligned with the methods of the core encoding of the component channels of the downmix audio signal, which is generated according to clause 5.4.5.

The core coding is afforded by SCE, CPE, or MCT for 1, 2, or more transport channels, respectively. According to clause 5.2.3, the component channels of the downmix are, in each time-frame, split into two portions.

These portions consist of different frequency bands. The first portion contains low frequency bands up to a border frequency. The second portion contains the high frequency bands above this border frequency. These two portions are encoded into two different encoded representations.

The first portion (low frequencies) is encoded into a waveform-preserving representation, whereas the second portion in encoded into a more efficient representation that makes use of perceptual considerations. For example, in TCX mode, IGF is applied to fill gaps in the spectrum. The amount of TF tiles in the spectrum filled by IGF increases gradually with the frequency. For details see clause 5.3.3.2.11 of [3]. The border frequency depends on the bitrate according to Table 92 in [3].

The encoded representation of the first portion comprises a set of compressed ambisonics transport channels (first set), SPAR metadata, and no DirAC metadata. The DirAC metadata is omitted from the estimation and the bitstream writing and the spatial parameter estimator (DoA estimator and diffuseness estimator) runs on the decoder side. This portion includes the frequency bins up to including the 8th parameter band in Table 5.4-5.

The encoded representation of the second portion comprises the same ambisonics transport channels, DirAC metadata, and no SPAR metadata. Instead, the SPAR metadata are derived from a DirAC based model independently on the encoder and decoder side according to clauses 5.4.3.8.2 and 6.4.4.3.

Hence, the encoder writes to the output interface (bitstream) the SPAR parameters for low-frequency bands, the DirAC parameters for high frequency bands, and the encoded representations of the first and second portions of the component channels of the downmix.

5.4.3.6 Mono signal detection

5.4.3.6.1 Overview

During SBA operation, there are situations where a single Mono channel is sent as the W channel of an Ambisonics input, with silence or near-silence in the other channels.

SBA mono handling ensures proper behaviour of the codec with mono input whilst operating in SBA format. The algorithm consists of a mono detector in the encoder, signalling of mono through the bitstream and a mono detector/preserver to ensure the W channel is treated as mono in the decoder.

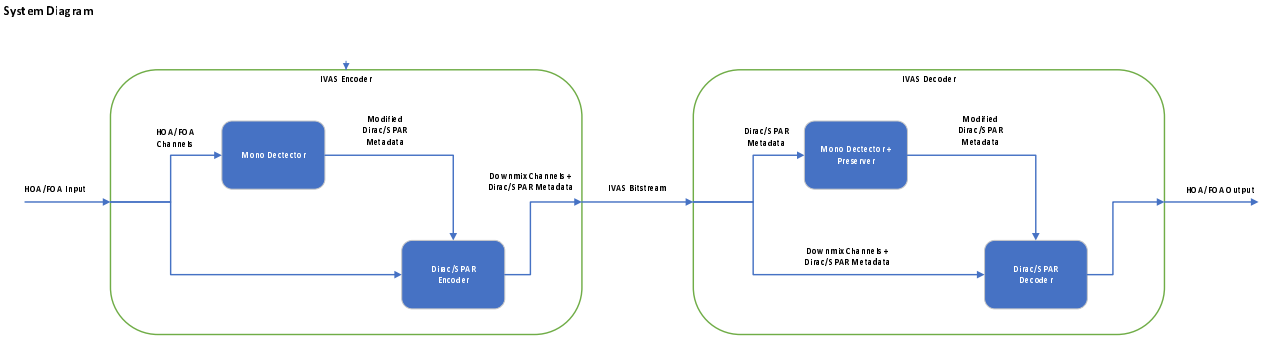


Figure 5.4-8: Mono Detector Algorithm

Encoder Mono Detector

The encoder mono detector determines when an Ambisonics input has content in the W channel only. For each audio block the detector analyses the content in all the channels and decides whether the block of audio can be declared mono or Ambisonics.

The detector works in the MDFT domain by, calculating, for each band, the power of the W channel W_p and the sum of the power of the other channels $\Sigma_p$in the input signal. The detector uses W_p, Σ_p and their ratio $\frac{W_p}{\Sigma_p}$ to determine if a band is mono, Ambisonics or silence. The classification decision (mono, multi-channel or silence) is based on two thresholds: a static noise threshold N_{thr} and a dynamic ratio threshold R_{thr}.

The noise threshold N_{thr} is used to determine whether a band is silent/near silent or has content. The ratio threshold R_{thr} is dynamic, such that it scales with the signal as the signal's amplitude decreases. R_{thr} is used to classify a band as either Ambisonics or mono It is calculated using a linear function with W_p as input and is also clipped at a maximum and minimum threshold.

After determining the state of all the frequency bands, the detector checks if any band was classified as Ambisonics. If any band was Ambisonics, the entire audio frame is declared as non-mono. If none of the bands were classified as Ambisonics, then all the bands must be either silence or mono. If there is at least one band that is mono then the audio frame is declared as mono.

There is further processing to ensure that the mono flag cannot toggle on and off quickly, causing instability. The processing looks at the history of mono flags at the block level and only after there have been consecutive mono flags for 30 ms does it classify the signal as mono and communicate this decision to the decoder as described in clause 5.4.3.6.2.

The process for detecting mono in the encoder is as follows:

- For each frame of the encoder, loop over each MDFT frequency band
- For each frequency band, calculate:
  - W channel power (W_p)
  - Sum of the power of each individual non-W channel (Σ_p)
- Check if the sum power is close to zero to prevent divide by zero errors.
  - If the sum power is close to zero check if the W_p is greater than the noise threshold N_{thr} and declare the band as mono if it is.
  - Otherwise restart the loop iteration.
- Calculate the $\frac{W_p}{\Sigma_p}$.
- Calculate the ratio threshold R_{thr}.
  - Normalise W_p to be between 0 and 1.

- Scale MAX_TRESHOLD by $norm(W_p)$.
- Clip the calculated threshold to ensure the range is within MIN_THRESHOLD and MAX_TRESHOLD.
- Check if the calculated ratio is above R_{thr}. If $\frac{W_p}{\Sigma_p} > R_{thr}$, declare the band as mono
- Otherwise, declare the band as Ambisonics.

- If none of these checks pass, then the band is declared as silence.
- After the frequency band loop, check if any band was declared as Ambisonics.
  - If true, the encoder frame is non-mono.
  - Otherwise, check if any band was declared as mono.
- If true, the encoder frame is mono.
- Otherwise, encoder frame is non-mono.
- If the frame is mono, increment the mono frame counter.
  - If the mono frame counter is > 30ms, then set the global mono flag to 1.
- If the frame is non-mono, reset the mono frame counter to zero.

Detector Flow Diagram

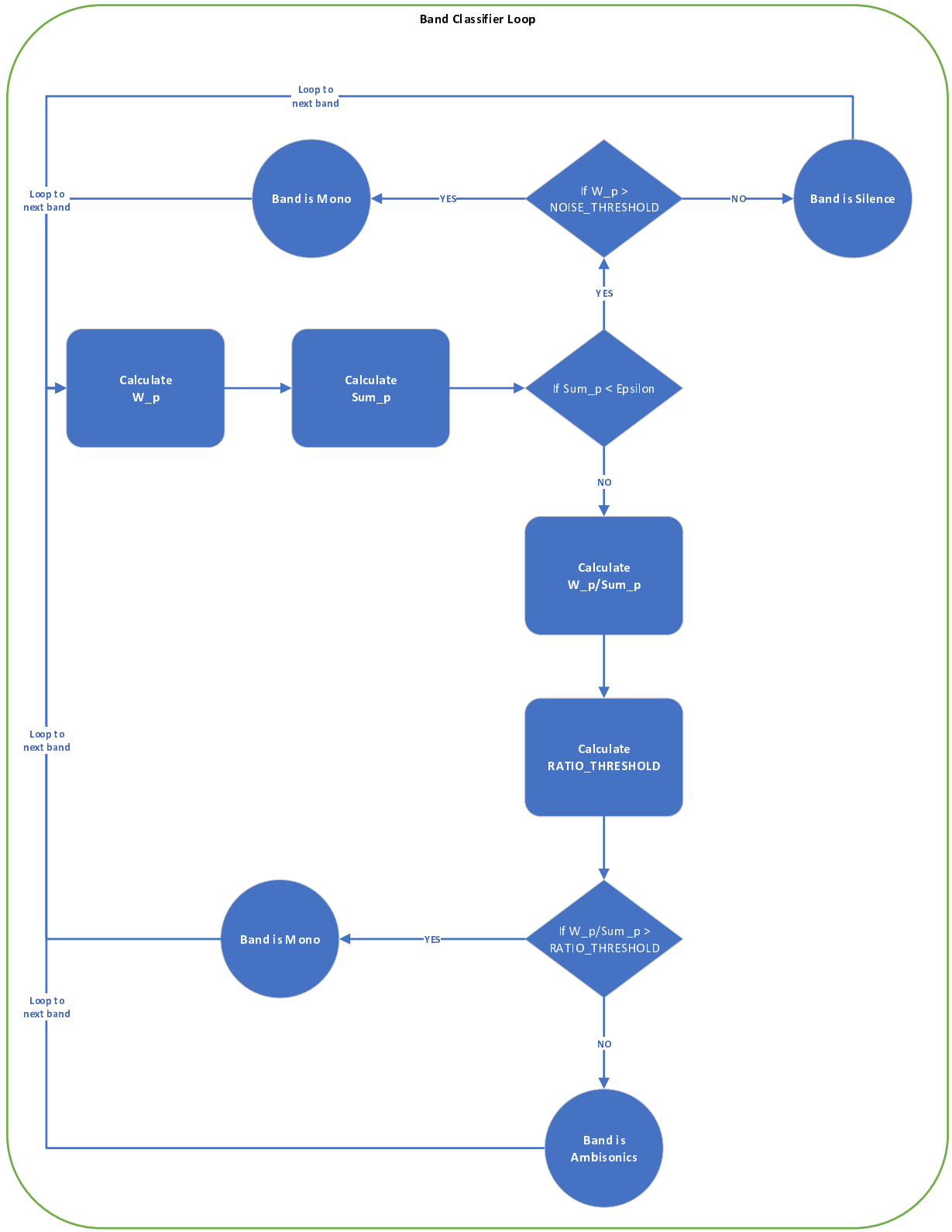


Figure 5.4-9: Mono Detector Band Classifier Loop

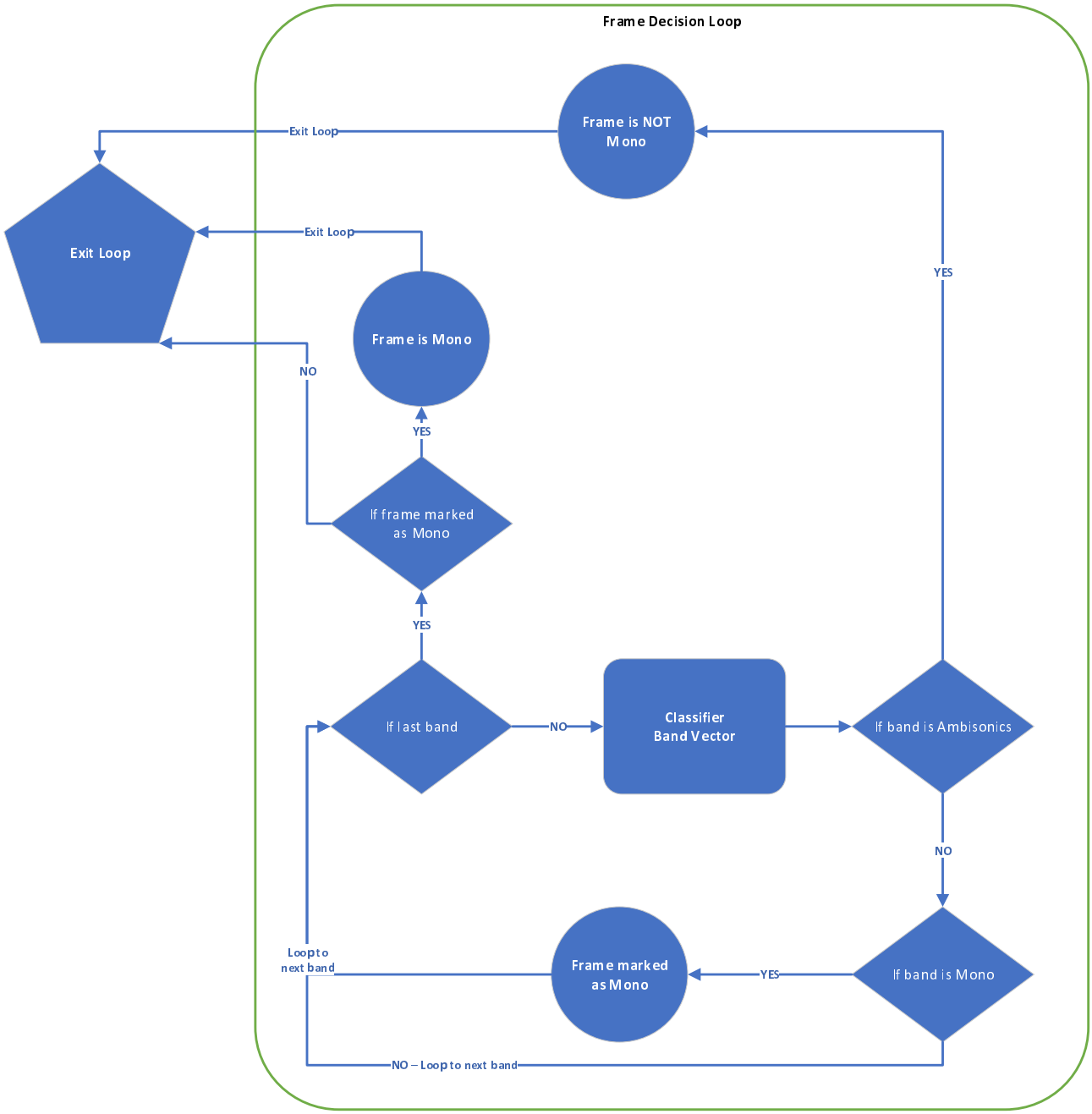


Figure 5.4-10: Mono Detector Frame Decision Loop

5.4.3.6.2 Signalling of Mono through the bitstream

The mono detection information is transmitted from the encoder to the decoder implicitly by modifying existing IVAS metadata fields to specific values. This ensures no additional bitrate is consumed by this feature.

Six different SPAR and DIRAC metadata fields are set to zero at the encoder and then checked at the decoder. If the 6 metadata fields are zero in the bitstream then the decoder will process the frame as mono. The signalling also considers quantisation of the metadata values through the codec. Due to quantisation, certain values cannot be encoded in the bitstream. The detector in the decoder accounts for this quantisation to detect mono correctly.

The metadata values are:

- SPAR metadata
  - Prediction coefficients (PR)

- Cross-prediction coefficients (CP)
- Decorrelation coefficients (D)

- DiRAC metadata
  - energyratio ($1 - \Psi$)
  - azimuth (ϕ)
  - elevation (θ)

These metadata are all set to zero when mono is detected. The SPAR values are related to extracting higher-order channels out of the downmixed channels sent in the bitstream, when they are set to zero, the extraction is not carried out. The DiRAC values are related to diffuseness and position. When they are set to zero the mono channel is not spread across other channels in the Ambisonics output.

5.4.3.7 SPAR Metadata computation

5.4.3.7.1 Overview

SPAR encodes Ambisonics signal by determining N_{dmx} channel downmix signal and performing energy compaction to generate compacted downmix signal. SPAR metadata is generated based on Ambisonics input and compacted downmix signal such that the compacted downmix signal can be upmixed to reconstruct an approximation of Ambisonics input to SPAR. The SPAR metadata and compacted downmix signal is coded in the bitstream. SPAR metadata consist of quantized and coded prediction (PR) coefficients, cross-prediction (CP) coefficients and decorrelation (D) coefficients. PR coefficients are always coded in SPAR metadata bitstream whereas presence of CP and D coefficients depends on number of downmix channels N_{dmx}. Table 5.4-7 shows which SPAR metadata parameters are computed and coded at a given downmix configuration.

Table 5.4-7: SBA metadata parameters at various downmix configurations

N_{dmx}	Metadata parameters
1	PR, D
1< N_{dmx} < $N_{spar_{ana}}$	PR, C and D
$N_{spar_{ana}}$	PR

Prediction coefficients indicate the correlation between W channel and other channels of SBA input (or side channels) and are computed such that SBA input can be converted into a downmix signal that has lower energy than input signal by subtracting the prediction component from the side channels as follows.

$$S' = S - pr_s W'$$

where W' is the primary downmix channel, S is the side channel and S' is either the side channel prediction error or the residual side channel. With PR coefficients, an N_{spar_ana}-channel input is converted to an N_{dmx}-channel downmix signal, wherein the downmix signal includes W' and zero or more residual side channels.

At the decoder, the side channel(s) are reconstructed as follows:

$$S = S' + pr_s W'.$$

At low bitrates, the number of downmix channels that are coded in the IVAS bitstream is less than the number of input channels to SPAR and additional parameters CP and D coefficients are computed to reconstruct an ambisonics signal from the downmix signal that preserves the second-order statistics of the Ambisonics audio scene (i.e. the covariance) after decoding. The CP coefficients achieve further energy compaction by using the correlation between residual side channels that are coded in the bitstream and the parametric channels, that is, the residual side channels that are not coded in the bitstream.

The decorrelation coefficients indicate the residual energy in the parametric channels that cannot be predicted using either prediction or cross-prediction. At the SPAR decoder the time domain decorrelator described in clause 6.2.1.3 is used to separate the transient component of primary downmix channel W' and generate a slowly-fluctuating

decorrelated output, corresponding to each parametric channel. The D coefficients are then used to scale the decorrelator output signal, which is then combined with the prediction signal, where prediction signal is generated based on the PR, CP coefficients and W' channel and includes a transient component, to construct the side channels.

The energy compaction performed with PR and CP coefficients is an orthogonal transform such that residual signal is uncorrelated with respect to the primary downmix channel, and the parametric signal is uncorrelated with respect to the residual signal that is coded in IVAS bitstream. As part of downmixing, the FOA component of SBA signal is rotated along the X and Z axis, such that WYZX ordering becomes WYXZ, to generate the energy compacted rotated signal S. For cases, where $N_{dmx} < N_{spar\_ana}$, the remaining channels are reconstructed at decoder as

$$S_2 = \mathrm{P}\,\mathrm{S}_1 + D\ decorr(S_1)$$

where P is a combination of PR and CP coefficients, S_1 is the residual signal component of signal S and is waveform coded using core coding tools, $decorr(S_1)$ is the decorrelated signal with respect to primary downmix channel of S_1, S_2 is the parametrically reconstructed signal component of signal S that is not transmitted in IVAS bitstream.

A detailed description of the computation of SPAR metadata parameters is provided in the following clauses.

5.4.3.7.2 Banded covariance computation

5.4.3.7.2.1 Overview

SPAR metadata analysis channels from MDFT signal $S_j^{sba\_ana,mdft}(\mathrm{m},k)$ are processed through magnitude response of MDFT filterbank to generate banded covariance matrix $C^{in}_{[N_{spar_{ana}} x N_{spar_{ana}}]}(b)$, here b is the band index such that $1 \leq b \leq nB$, where $nB = 12$ for 32 kHz and 48 kHz sampling rates and 10 for 16 kHz sampling rate.

When sampling frequency is 16 kHz, number of MDFT filterbank bands nB are set to 10. However, to avoid additional signalling bit in the bitstream, SPAR metadata computation is still done on 12 bands by appending 2 bands with magnitude response set to zero.

5.4.3.7.2.2 Band covariance

The channels of signal $S_j^{sba\_ana,mdft}$ are grouped into a column vector $V(m,k)$ as given in Equation (5.4-18).

$$V(m,k) = \begin{bmatrix} S_1^{sba\_ana,mdft}(\mathrm{m},k) \\ S_2^{sba\_ana,mdft}(\mathrm{m},k) \\ \cdot \\ \cdot \\ \cdot \\ S_{N_{spar\_ana}}^{sba\_ana,mdft}(\mathrm{m},k) \end{bmatrix} \quad (5.4\text{-}18)$$

Then covariance for each MDFT bin k is computed as follows:

$$C^{in,mdft}_{[N_{spar\_ana} x N_{spar\_ana}]}(m,k) = re(V(m,k)V^H(m,k)) \quad (5.4\text{-}19)$$

where V^H is the conjugate transpose of matrix V and $re(x)$ returns real value of complex variable x.

Then banded covariance is computed as follows:

$$C^{in,banded}_{[i,j]}(b) = \sum_{m=1}^{m=4} \sum_{k=k_0^b}^{k=k_1^b} (C^{in,mdft}_{[\mathrm{i,j}]}(m,k)\, H^{mdft}_{abs,5ms}(b,k)) \quad (5.4\text{-}20)$$

where i and j are channel indices with $1 \leq i \leq N_{spar\_ana}$, $1 \leq j \leq N_{spar\_ana}$, $H^{mdft}_{abs,5ms}(b,k)$ is the magnitude response of MDFT filterbank (described in clause 5.2.5.3.2) for band index b and frequency bin index k, k_0^b is the bin index of first non-zero magnitude response bin of MDFT filter bank at band index b and k_1^b is the bin index of last non-zero magnitude response bin of MDFT filter bank at band index b.

5.4.3.7.2.3 Covariance smoothing

In SPAR, the banded covariance matrix is smoothed across signal frames by a first-order low-pass filter and a forgetting factor. The smoothing process computes a forgetting factor $\alpha_{sm}(b)$ based on the ratio of the effective number of bins to the desired number of bins in a band, where the effective number of bins is computed by summing the filter bank's magnitude response in that band, as shown in equation (5.4-24). When DTX is off or when the VAD detector output (described in clause 5.4.3.1) W_{vad} is set to 1, the smoothed covariance is generated based on the previous frame's smoothed covariance matrix, forgetting factor and banded covariance in the current frame, $C_{[i,j]}^{in,banded}(b)$, as follows:

$$C_{[i,j]}^{in}(b) = C_{[i,j]}^{in,prev}(b) + \alpha(b)(C_{[i,j]}^{in,banded}(b) - C_{[i,j]}^{in,prev}(b)) + \alpha(b)fac \quad (5.4\text{-}21)$$

where

$$fac = \begin{cases} 0, & if\ i\ !=\ j \\ \varepsilon, & if\ i == j \end{cases} \quad (5.4\text{-}22)$$

is a small value added for numerical stability, $C_{[i,j]}^{in,prev}(b)$ is the value of $C_{[i,j]}^{in}(b)$ in previous frame, and $\alpha(b)$ is the time-varying forgetting factor for frequency band b. Covariance smoothing is reset when a transient is detected using the transient detector described in clause 5.4.3.2. When a transient is detected on the W channel, covariance smoothing is reset for all channels and $C_{[i,j]}^{in}(b) = C_{[i,j]}^{in,banded}(b)$. Resetting can be equivalently achieved by setting $\alpha(b)$ to a value of 1.0. The resetting mechanism is described in eq. (5.4-23), where T_0 and T_1 are outputs of the transient detector.

$$\alpha(b) = \begin{cases} 1, & frameIdx_{prev} = -1\ OR\ T_1 = 1 \\ 1, & T_0 = 1\ AND\ b > 2 \\ \alpha_{sm}(b), & otherwise \end{cases} \quad (5.4\text{-}23)$$

The forgetting factor $\alpha_{sm}(b)$ is computed as follows.

$$N_b = \sum_{k=k_0^b}^{k=k_1^b} H_{abs,5ms}^{mdft}(b,k) \quad (5.4\text{-}24)$$

$$\alpha_{sm}(b) = \min\left(0.8, \frac{N_b}{N_{min}}\right) \quad (5.4\text{-}25)$$

where

$$N_{min} = \begin{cases} \frac{24}{0.5b}, & if\ ivas\ bit\ rate < 24.4kbps \\ \frac{24}{0.75b}, & if\ ivas\ bit\ rate \geq 24.4kbps \end{cases} \quad (5.4\text{-}26)$$

represents the desired number of bins in a band. The value of N_{min} is dependent on both the band index b and the bitrate. the dependency on b provide additional smoothing at low-frequency bands for a given bitrate. At bit rates below 24.4kbps, the factor of 0.5 in equation (5.4-26) has the effect of further increasing smoothing at the lower bands (i.e., bands in the f^{SPAR} frequency range), while avoiding modifying smoothing at higher bands ((i.e., bands in the f^{DirAC} frequency range)).

When DTX is enabled and the VAD detector output W_{vad} is set to 0, the covariance matrix in each band $C_{[i,j]}^{in,banded}(b)$ is smoothened, as described in clause 5.4.9.3.2, to generate $C_{[i,j]}^{in}(b)$.

5.4.3.7.3 Active W downmix detector

SPAR uses active W downmixing to mix contents of X, Y, and Z channels into the W channel in the downmixing process described in clause 5.4.5. Active W downmixing is activated based on the output of active W downmix detector $Flag_{activeW}$ given in Equation (5.4-27).

$$Flag_{activeW} = \begin{cases} 1 & , if\ IVAS\ bitrate \leq 32\ \text{kbps} \\ Dyn_{activeW} & , if\ 48\ \text{kbps} \leq IVAS\ bitrate \leq 192\ \text{kbps} \\ 0 & , if\ IVAS\ bitrate \geq 256\ \text{kbps} \end{cases} \quad (5.4\text{-}27)$$

Computation of $Dyn_{activeW}$ is as follows. First, broadband variance for all FOA channels is computed

$$Var_{[i]}^{in} = \sum_{b=1}^{b=nB} C_{[i,i]}^{in,banded}(b) \quad (5.4\text{-}28)$$

where nB is the number of MDFT filter bank bands, i is the channel index with $1 \leq i \leq 4$.

Then parametric channel variance is computed by summing up the variance of FOA channels that are not present in the downmix signal as shown below.

$$Var_{p}^{in} = \sum_{i=N_{dmx}+1}^{i=4} Var_{[i]}^{in} \quad (5.4\text{-}29)$$

where N_{dmx} is the number of downmix channels at a given bitrate as given in Table 5.4-1.

Then ratio of parametric channel variance to W channel variance is computed as follows.

$$En_{ratio} = \frac{Var_{[1]}^{in}}{Var_{p}^{in}} \quad (5.4\text{-}30)$$

Then $Dyn_{activeW}$ is computed by comparing Var_{p}^{in} and En_{ratio} against fix threshold values as follows.

$$Dyn_{activeW} = \begin{cases} 0 & , if\ Var_{p}^{in} < 1024 \\ 0 & , if\ En_{ratio} \geq 0.00001521 \\ 1 & , if\ En_{ratio} < 0.00001521 \end{cases} \quad (5.4\text{-}31)$$

At bitrates where the number of downmix channels is equal to 2 or 3 (as given in Table 5.4-1.), a channel index, Res_{ind} as given in Equation (5.4-32), is also computed which indicates the side channel index that is to be mixed into the W channel in the downmixing process described in clause 5.4.5.

$$Res_{ind} = \begin{cases} 3 & , if\ N_{dmx} = 2\ , Var_{[3]}^{in} \geq Var_{[4]}^{in} \\ 4 & , if\ N_{dmx} = 2\ , Var_{[4]}^{in} < Var_{[3]}^{in} \\ 3 & , if\ N_{dmx} = 3 \end{cases} \quad (5.4\text{-}32)$$

Note that channel index 3 corresponds to the Z channel and channel index 4 corresponds to X channel of the ambisonics signal with ACN ordering.

Dynamic active W flag ($Dyn_{activeW}$) is coded in metadata bitstream when number of downmix channels N_{dmx} is set to 2 or 3. Active W residual index (Res_{ind}) is coded in metadata bitstream when number of downmix channels N_{dmx} is set to 2 as described in clause 8.3.1.

5.4.3.7.4 Banded covariance re-mixing

SPAR (Spatial Reconstruction) computes parameters for band indices 1 to nB_{md}, where nB_{md} is computed as follows.

First, number of SPAR bands are computed:

$$nB_{spar} = \begin{cases} 8 & , if\ N_{spar\_ana} = 4 \\ nB & , otherwise \end{cases} \quad (5.4\text{-}33)$$

Then, nB_{md} is computed as follows:

$$nB_{md} = \frac{nB_{spar}}{bw_{bands}} \quad (5.4\text{-}34)$$

where

$$bw_{bands} = \begin{cases} 4 & , if\ W_{vad} = 0 \\ 2 & , if\ W_{vad} = 1\ , IVAS\ bitrate \leq 16.4\ \text{kbps} \\ 1 & , otherwise \end{cases} \quad (5.4\text{-}35)$$

The frequency resolution of the covariance matrix $C_{[i,j]}^{in}$ computed in Equation (5.4-21) is modified as per the value of bw_{bands} and a new covariance matrix $C_{[i,j]}^{md}$ is computed as given below.

$$C_{[i,j]}^{md}(b) = \sum_{b1=1+(b-1)*bw_{bands}}^{b1=b*bw_{bands}} C_{[i,j]}^{in}(b1) \quad (5.4\text{-}36)$$

where b is the band index with $1 \leq b \leq nB_{md}$.

5.4.3.7.5 SPAR parameter estimation

The computation of SPAR parameters, that is PR, CP and D coefficients, is described in detail in the following subclauses.

5.4.3.7.6 Passive-W channel prediction coefficients computation

Passive-W channel prediction mode is set when $Flag_{activeW}$ is 0, as described in clause 5.4.3.7.3. In passive-W channel prediction mode, the primary downmix channel *W'* is simply a delayed version of W-channel input to SPAR and the computation of PR coefficients is as follows.

$$PR_{j-1}(b) = \frac{C^{md}_{[1,j]}(b)}{\max\left(C^{md}_{[1,1]}(b), \sqrt{\sum_{k=2}^{k=N_{spar\_ana}} \left|C^{md}_{[1,k]}(b)\right|^2}, \varepsilon\right)} \quad (5.4\text{-}37)$$

where b is the band index with $1 \leq b \leq nB_{md}$ and j is the channel index with $2 \leq j \leq N_{spar\_ana}$.

5.4.3.7.7 Active-W channel prediction coefficients computation

5.4.3.7.7.1 Overview

Active-W channel prediction mode gets enabled at IVAS bitrates ≤ 192 kbps when $Flag_{activeW}$, described in clause 5.4.3.7.3, is 1. In active-W channel prediction mode, encoding downmix that is applied at the encoder is different than the decoding re-mix or upmix applied at the decoder. The primary input channel, that is the W channel, and the side channels, that is the X, Y and Z channels of Ambisonics input are processed to compute banded covariance as described in Equation (5.4-36). Then downmixing gains are computed corresponding to each side channel and the primary downmix channel is computed by scaling the side channels with the corresponding downmixing gains and then adding them together with W channel as given below.

$$W'' = W + r_y Y + r_z Z + r_x X$$

where r_y, r_z, r_x are downmixing gains corresponding to Y, Z and X channel respectively. These gains are determined such that overall prediction error on the side channels is minimized as described in clause 5.4.3.7.7.2.

Then, a downmix scaling gain is computed to further scale the primary downmix channel such that the energy difference between the up-mixed signal at the decoder and input signal at the encoder is minimized.

$$W' = W_{scale} W''$$

where W_{scale} is the downmix scaling gain as computed in Equation (5.4-51).

Combining the downmixing gains and downmix scaling gains, the computation of *W'* in a given time-frequency tile is given as

$$W' = s_w W + s_y Y + s_z Z + s_x X, \quad (5.4\text{-}38)$$

where s_w, s_y, s_z, s_x are the gains applied to the FOA channels.

Then active W prediction coefficients are computed corresponding to each side channel based on the input covariance matrix, downmixing gains and the downmix scaling gains as described in clause 5.4.3.7.7.2. These prediction coefficients are applied to the primary downmix channel W' to generate the predicted side channels. Then residual or post-prediction channels are computed to by subtracting the predicted side channel from the corresponding input side channel as given below.

$$X' = X - pr_x W'$$

$$Y' = Y - pr_y W'$$

$$Z' = Z - pr_z W'$$

where pr_x, pr_y, pr_z are the prediction coefficients corresponding to X, Y and Z channel respectively. Then cross-prediction coefficients are computed as described in clause 5.4.3.7.8 and decorrelation coefficients are computed for the residual channels that are not transmitted to the decoder by normalizing the energy of the corresponding residual channel as described in clause 5.4.3.7.9.

The prediction coefficients, cross-prediction coefficients, decorrelation coefficients and the downmix signal, including primary downmix channel W' and zero or more residual signals, are coded in IVAS bitstream and transmitted to the decoder. In active-W channel prediction mode, encoding downmix that is applied at the encoder is different than the decoding re-mix or upmix applied at the decoder, where the primary downmix channel W' is computed at the encoder as per Equation (5.4-38) whereas at the decoder, W channel is reconstructed by just applying a gain to primary downmix channel W' as given below.

$$W = g_w W'$$

where g_w is the energy compensation gain.

The following clauses contain a more detailed description of the computation of PR coefficients in active W mode.

5.4.3.7.7.2 Active W prediction coefficients in 1 channel downmix mode

PR coefficients $PR_{[3x1]}(b)$ with $1 \le b \le nB_{md}$ are given in Equation (5.4-52). The following steps are performed to compute $PR_{[3x1]}(b)$ from covariance matrix $C^{md}_{[4x4]}(b)$.

First, the covariance matrix that is computed in Equation (5.4-36) is represented as follows.

$$C^{md}_{[4x4]}(b) = \begin{pmatrix} w & \alpha\hat{u}^* \\ \alpha\hat{u} & U \end{pmatrix} \quad (5.4\text{-}39)$$

where $\hat{u}\ is\ 3x1$ unit vector, $\alpha\hat{u} = \begin{pmatrix} C^{md}_{[2,1]}(b) \\ C^{md}_{[3,1]}(b) \\ C^{md}_{[4,1]}(b) \end{pmatrix}, w = C^{md}_{[1,1]}(b), \alpha = \sqrt{\sum_{k=2}^{k=N_{spar\_ana}} \left|C^{md}_{[k,1]}(b)\right|^2}, U = \begin{pmatrix} C^{md}_{[2,2]}(b) & C^{md}_{[2,3]}(b) & C^{md}_{[2,4]}(b) \\ C^{md}_{[3,2]}(b) & C^{md}_{[3,3]}(b) & C^{md}_{[3,4]}(b) \\ C^{md}_{[4,2]}(b) & C^{md}_{[4,3]}(b) & C^{md}_{[4,4]}(b) \end{pmatrix}$.

Then, a passive prediction factor $g_{passive}$ is computed as given below:

$$g_{passive} = \frac{\alpha}{w} \quad (5.4\text{-}40)$$

Computation of active W prediction coefficients depend on the value of $g_{passive}$ as given below.

When $g_{passive} < 3$, the downmix matrix with active W prediction coefficients is computed as

$$PRdmx_{[4x4]} = \begin{pmatrix} 1 & (0,0,0) \\ -g\hat{u} & I_3 \end{pmatrix} \begin{pmatrix} 1 & F\hat{u}^H \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & I_3 \end{pmatrix} = \begin{pmatrix} 1 & F\hat{u}^H \\ -g\hat{u} & I_3 - gF\hat{u}\hat{u}^H \end{pmatrix} \quad (5.4\text{-}41)$$

where $g\hat{u}$ are the prediction coefficients, $F = \min\left(1, \frac{f\alpha}{max\left(mw, \frac{trace(U)}{1}\right)}\right)$ $I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$,

$$f = \begin{cases} 0.25 & if\ ivas\ bitrate \le 16.4\ kbps\ OR\ W_{vad} == 0 \\ 1 & otherwise \end{cases} \quad (5.4\text{-}42)$$

$$m = \begin{cases} 1 & if\ b > 8, N_{spar\_ana} = 4 \\ 3 & otherwise \end{cases} \quad (5.4\text{-}43)$$

To compute g, the post prediction matrix is computed and then the covariance between primary downmix channel and predicted channels is minimized which results in a linear equation in g as given in Equation (5.4-45).

$$Post\_prediction_{[4x4]} = PRdmx_{[4x4]} * C^{md}_{[4x4]}(b) * PRdmx_{[4x4]}{}^H = \begin{pmatrix} \cdot & \cdot \\ \hat{r} & \cdot \end{pmatrix} \quad (5.4\text{-}44)$$

In equation (5.4-44), $\hat{r}$ is the indicator of overall prediction error on the side channels and is minimized by setting $\hat{u}^H \hat{r} = 0$. This results in linear equation in g as given below.

$$g\beta F^2 + 2\alpha g F + wg - \beta F - \alpha = 0 \tag{5.4-45}$$

$$g = \frac{\alpha + \beta F}{\beta F^2 + 2\alpha F + w} \tag{5.4-46}$$

where $\beta = \hat{u}^H U \hat{u}$.

Prediction (PR) coefficients in active W mode $PR_{[3x1]}$ are computed as $PR_{[3x1]} = g\hat{u}$. These coefficients are further scaled as per Equation (5.4-52).

When $g_{passive} \geq 3$, the downmix matrix with active W prediction coefficients is computed as

$$PRdmx_{[4x4]} = \begin{pmatrix} 1 & (0,0,0) \\ -g\hat{u} & I_3 \end{pmatrix} \begin{pmatrix} 1 & F\hat{u}^H \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & I_3 \end{pmatrix} = \begin{pmatrix} 1 & F\hat{u}^H \\ -g\hat{u} & I_3 - gF\hat{u}\hat{u}^H \end{pmatrix} \tag{5.4-47}$$

where F is the input mixing strength coefficient with $F = gf$ and the computation of f and g is as follows.

First, the post prediction matrix is computed and then the covariance between primary downmix channel and predicted channels is minimized which results in a quadratic equation in f as given in Equation (5.4-49).

$$C_{[4x4]}^{postpred} = PRdmx_{[4x4]} * C_{[4x4]}^{md}(b) * PRdmx_{[4x4]}{}^H = \begin{pmatrix} \cdot & \cdot \\ \hat{r} & \cdot \end{pmatrix} \tag{5.4-48}$$

Then $\hat{r}$ is minimized by setting $\hat{u}^H \hat{r} = 0$ as follows:

$$\hat{u}^H \hat{r} = f^2 \beta g^3 + 2\alpha g^2 f + wg - \beta g f - \alpha = 0 \tag{5.4-49}$$

In Equation (5.4-49), g is set to 3 and $f$is computed by solving the quadratic equation as follows:

$$f = \frac{\left((\beta - 2\alpha g) + \sqrt{4\alpha^2 g^2 + \beta^2 - 4\beta g^2 w}\right)}{\max(2\beta g^2, \varepsilon)} \tag{5.4-50}$$

Prediction (PR) coefficients in active W mode $PR_{[3x1]}$ are computed as $PR_{[3x1]} = g\hat{u}$. These coefficients are further scaled as per Equation (5.4-52).

Computation of downmix scaling factor

In active-W prediction mode, the primary downmix channel W', obtained by using the prediction matrix in Equations (5.4-41) and (5.4-47), is further scaled with an additional gain W_{scale}.

The gain W_{scale} is computed by estimating the W channel energy at the upmixer output (as described in clause 6.4.6.4), where the upmixing gain is computed as a function of prediction coefficients transmitted from encoder to decoder, and then equating the estimated energy of W channel with the energy of W channel input to SPAR and solving the closed form solution of the resulting polynomial. This method attempts to minimize the energy difference between the upmixed signal at the decoder and input signal at the encoder and results in W_{scale} value as given below.

$$W_{scale} = \frac{g_w + \sqrt{g_w{}^2 + 4 f_s g^2}}{2} \tag{5.4-51}$$

where $f_s = \begin{cases} 0.25 & if\ ivas\ bitrate \leq 16.4\ kbps\ OR\ W_{vad} = 0 \\ 0.5 & otherwise \end{cases}$, $w = C_{[1,1]}^{md}(b)$, $dmx_{[1x4]} = (1 \quad F\hat{u}^H)$, $w' = dmx_{[1x4]} C_{[4x4]}^{md}(b) dmx_{[1x4]}{}^H$, and $g_w = \sqrt{\frac{w}{w'}}$.

The unquantized PR coefficients in active W mode are then computed as follows:

$$PR_{[3x1]}(b) = \frac{g\hat{u}}{W_{scale}} \tag{5.4-52}$$

The downmixing gains s_w, s_y, s_z, s_x mentioned in Equation (5.4-38) are computed as $s_{[1x4]}$ as follows.

$$s_{[1x4]} = dmx_{[1x4]} W_{scale} \tag{5.4-53}$$

5.4.3.7.7.3 Active W prediction coefficients in 2 and 3 channel downmix mode

In 2- and 3-channel downmix mode, the downmix matrix with active W prediction coefficients is computed as per Equation (5.4-47). In this mode f and g are set to 1 and the unit vector is computed as

$$\hat{u}_i = \begin{cases} 0 & if\ i\ != Res_{ind} - 1 \\ 1 & if\ i\ == Res_{ind} - 1 \end{cases}$$

where Res_{ind} is computed in Equation (5.4-32). W_{scale} is set to 1 and Prediction coefficients and downmix matrix is computed as

$$PR_{[3x1]}(b) = \hat{u}. \tag{5.4-54}$$

5.4.3.7.8 Cross-Prediction coefficients computation

To compute cross-prediction coefficients, first, downmix matrix is computed from the prediction coefficients, as

$$PRdmx_{[N_{spar\_ana}xN_{spar\_ana}]} = \begin{pmatrix} 1 & 0 \\ -PR_{[N_{PR}\,x\,1]}(b) & I_{N_{PR}} \end{pmatrix} \begin{pmatrix} W_{scale} & W_{scale} F \hat{u}_{[1xN_{PR}]}{}^{H} \\ 0 & I_{N_{PR}} \end{pmatrix} \tag{5.4-55}$$

where $PR_{[N_{PR}\,x\,1]}(b)$ is the column vector of prediction coefficients with $N_{PR} = N_{spar\_ana} - 1$, $I_{N_{PR}}$ is an identity matrix and $\hat{u}_{[1xN_{PR}]}$ is the unit vector as given in equations (5.4-41) and (5.4-47). Then,

$$F = \begin{cases} F & if\ Flag_{activeW} = 1 \\ 0 & if\ Flag_{activeW} = 0 \end{cases}$$

and

$$W_{scale} = \begin{cases} W_{scale} & if\ Flag_{activeW} = 1 \\ 1 & if\ Flag_{activeW} = 0 \end{cases}$$

Then the downmix matrix is remixed as per a remix matrix. The remix matrix rearranges downmix channels as per the priority order which is decided based on perceptual importance of ambisonics channels. Priority order for FOA signal is given below.

$$W > Y > X > Z \tag{5.4-56}$$

Based on this priority order a remix matrix $remix_{[N_{spar\_ana}xN_{spar\_ana}]}$ is created as given below.

$$remix_{[i,j]} = \begin{cases} I_{[i,j]} & , i > 4, j > 4 \\ remixFOA_{[i,j]}, & \text{otherwise} \end{cases} \tag{5.4-57}$$

where $1 \leq i \leq N_{spar\_ana}\ and\ 1 \leq j \leq N_{spar\_ana}$ and

$$remixFOA_{[4,x4]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{5.4-58}$$

The remixed downmix matrix is computed as follows:

$$PRdmx_{[N_{spar\_ana}xN_{spar\_ana}]} = remix_{[N_{spar\_ana}xN_{spar\_ana}]} PRdmx_{[N_{spar\_ana}xN_{spar\_ana}]} \tag{5.4-59}$$

With the remixed downmix matrix, post-prediction matrix is computed as follows.

$$C^{postpred}_{[N_{spar\_ana}xN_{spar\_ana}]} = PRdmx_{[N_{spar\_ana}xN_{spar\_ana}]}\ C^{md}_{[N_{spar\_ana}\,x\,N_{spar\_ana}]}(b)\ PRdmx_{[N_{spar\_ana}xN_{spar\_ana}]}{}^{H} \tag{5.4-60}$$

CP coefficients cross-predict any remaining portion of the channels that are not part of the downmix signal (also referred to as fully parametric channels) from the downmix signal. From the post-predicted matrix, CP coefficients are computed as follows.

$$CP_{[(N_{spar\_ana}-N_{dmx})\ x\ (N_{dmx}-1)]}(\mathrm{b}) = \begin{cases} 0, & if\ N_{dmx} = 1 \\ 0, & if\ N_{dmx} = N_{spar\_ana} \\ 0, & if\ W_{vad} = 0 \\ 0, & if\ tr(R_{dd})0.005 < \varepsilon \\ R_{ud}(R_{dd} + I * max(\varepsilon, tr(R_{dd})0.005))^{-1}, & \text{otherwise} \end{cases} \quad (5.4\text{-}61)$$

where $R_{du} = R_{ud}{}^{H} = C^{postpred}_{[2:N_{dmx},N_{dmx}+1:N_{spar\_ana}]}$, $R_{dd} = C^{postpred}_{[2:N_{dmx},2:N_{dmx}]}$ and I is the identity matrix with same dimensions as R_{dd}. When $N_{dmx} < 4$, CP coefficients are set to 0 if the determinant of matrix $(R_{dd} + I * max(\varepsilon, tr(R_{dd})0.005))$ is less than ε.

5.4.3.7.9 Decorrelation coefficients computation

The decorrelation coefficients ($D_{[(N_{spar\_ana}-N_{dmx})\ x\ 1]}$) are computed by normalizing the residual energy in fully parametric channels as given below.

$$D_{[(N_{spar\_ana}-N_{dmx})\ x\ 1]}(\mathrm{b}) = \mathrm{diag}(\sqrt{\max\left(0, real\left(diag(NRes_{uu})\right)\right)}) \quad (5.4\text{-}62)$$

where

$$NRes_{uu} = \frac{Res_{uu}}{\max(\varepsilon, R_{WW}, \mathrm{scale} * tr(|Res_{uu}|))},$$

$$Res_{uu} = \begin{cases} R_{uu} - (CP(b) * R_{dd} * CP(b)^{H}) & if\ 1 < N_{dmx} < N_{spar\_ana} \\ R_{uu} & \text{otherwise} \end{cases},$$

$$\mathrm{scale} = \begin{cases} 0.75, & if\ W_{vad} == 0\ AND\ N_{dmx} == 1 \\ 1, & otherwise \end{cases},$$

$$R_{WW} = C^{postpred}_{[1,1]},$$

$$R_{uu} = C^{postpred}_{[N_{dmx}+1:N_{spar\_ana},\ N_{dmx}+1:N_{spar\_ana}]},$$

$$R_{dd} = C^{postpred}_{[2:N_{dmx},2:N_{dmx}]},$$

and $CP(b)$ are the cross prediction coefficients computed in Equation (5.4-61).

5.4.3.7.10 Overview of quantization, bitrate distribution and coding of SPAR parameters

In SPAR mode, the SPAR metadata bitrate (to encode SPAR metadata parameters) and the audio downmix signal bitrate (to encode the downmix signal) are computed on frame-by frame basis using a bitrate distribution process and a quantization and coding process as described below.

SPAR parameters are encoded by a quantization and coding loop which includes iterating over multiple quantization strategies from fine to coarse quantization as described in clause 5.4.3.7.12. Then for each quantization strategy, calculating and quantizing the interrelated SPAR parameters, that is PR, CP and D coefficients, in a specific ordering as described in clause 5.4.3.7.12 and then iterating over multiple coding strategies to encode the quantized parameters. The first quantization strategy, with which SPAR metadata bitrate is less than or equal to the bitrate threshold, is selected to quantize the SPAR parameters, where the bitrate threshold is defined by target metadata bits $MDbits_{tar}$ and maximum metadata bits $MDbits_{max}$ which are computed based on the core-coder bitrates read from the SPAR bitrate distribution table as described in clause 5.4.3.7.11. The audio downmix signal bitstream is then generated by coding the downmix signal using the core coding tool, as per Table 5.4-1, and the downmix signal bitrate, where the downmix signal bitrate is obtained by subtracting metadata bitrate and header bits from total IVAS bitrate.

The IVAS bitstream contains quantized and coded SPAR metadata parameters, coded quantization strategy which indicates set of quantization levels with which SPAR metadata is quantized, and audio downmix bitstream.

5.4.3.7.11 SPAR bitrate distribution table

The Table 5.4-8 is the SPAR bitrate distribution table which contains an entry corresponding to each SBA bitrate. Row entry is selected based on IVAS bitrate, bandwidth and SBA analysis order. In addition to various bitrate dependent settings, each table row contains:

a) Target, minimum and maximum bitrate, $core_{brs}$, for each downmix signal.
b) Maximum of 3 quantization strategies (from fine to coarse quantization). Each quantization strategy contains number of quantization points for PR, CP and D coefficients $\{\{qlvl1_{pr}, qlvl1_{cp}, qlvl1_{d}, 1\}, \{qlvl2_{pr}, qlvl2_{cp}, qlvl2_{d}, 1\}, \{qlvl3_{pr}, qlvl3_{cp}, qlvl3_{d}, 1\}\}$.

From the row entry, target metadata bits $MDbits_{tar}$ and maximum metadata bits $MDbits_{max}$ are computed as follows.

$$MDbits_{tar} = \left(\left(ivas_{bits} - ivas_{headerbits} - sba_{headerbits} - \left(\frac{\sum_{n=1}^{n=N_{dmx}} core_{brs[n][1]}}{50}\right) - spar_{CSbits} - spar_{qbits}\right)\frac{nB_{md}}{12}\right) + spar_{CSbits} + spar_{qbits} \quad (5.4\text{-}63)$$

and

$$MDbits_{max} = \left(\left(ivas_{bits} - ivas_{headerBits} - sba_{headerbits} - \left(\frac{\sum_{n=1}^{n=N_{dmx}} core_{brs[n][2]}}{50}\right) - spar_{CSbits} - spar_{qbits}\right)\frac{nB_{md}}{12}\right) + spar_{CSbits} + spar_{qbits} \quad (5.4\text{-}64)$$

where $ivas_{bits} = \frac{ivas_{bitrate}}{50}$, $ivas_{headerBits} = 3$, $sba_{headerbits} = 3$, $spar_{CSbits} = 3$ and $spar_{qbits} = 2$.

Table 5.4-8: SPAR bitrate distribution table

IVAS bitrate [kbps]	SBA analysis order	bwidth	Number of downmix channels	Init time active_w	Core-coder bitrates {target, min, max} for each downmix channel	Quantization levels	TD ducking	AGC bits channel index
13.2	1	3	1	1	{ 10000, 8150, 13150 }	{ { 15, 1, 5, 1 },{ 15, 1, 3, 1 },{ 7, 1, 3, 1 } }	0	0
16.4	1	3	1	1	{ 13200, 11350, 16350 }	{ { 15, 1, 5, 1 },{ 15, 1, 3, 1 },{ 7, 1, 3, 1 } }	0	0
24.4	1	3	1	1	{ 16400, 14850, 24350 }	{ { 15, 1, 5, 1 },{ 15, 1, 3, 1 },{ 7, 1, 3, 1 } }	0	0
32	1	3	1	1	{ 24000, 20450, 31950 }	{ { 21, 1, 5, 1 },{ 15, 1, 5, 1 },{ 15, 1, 3, 1 } }	0	0
48	1	3	2	0	{ { 24000, 21000, 31950 },{ 16000, 15000, 20400 } }	{ { 15, 7, 5, 1 },{ 15, 7, 3, 1 },{ 7, 7, 3, 1 } }	1	0
64	1	3	2	0	{ { 38000, 34050, 56000 },{ 16000, 15600, 20400 } }	{ { 21, 7, 5, 1 },{ 15, 7, 5, 1 },{ 15, 7, 3, 1 } }	1	1
80	1	3	2	0	{ { 46000, 43000, 56000 },{ 24000, 23000, 31950 } }	{ { 21, 7, 5, 1 },{ 15, 7, 5, 1 },{ 15, 7, 3, 1 } }	1	0
96	1	3	3	0	{ { 47000, 42600, 56000 },{ 23000, 22600, 31950 },{ 16000, 15600, 20400 } }	{ { 21, 9, 9, 1 },{ 21, 7, 5, 1 },{ 21, 7, 5, 1 } }	1	0
128	1	3	3	0	{ { 55000, 50000, 56000 },{ 36000, 36000, 56000 },{ 27000, 27000, 31950 } }	{ { 21, 11, 9, 1 },{ 21, 9, 7, 1 },{ 21, 7, 7, 1 } }	1	0
160	1	3	3	0	{ { 74000, 70900, 112000 },{ 41000, 40050, 56000 },{ 35000, 34050, 56000 } }	{ { 21, 11, 11, 1 },{ 21, 9, 9, 1 },{ 21, 7, 7, 1 } }	1	0
192	1	3	3	0	{ { 90000, 87900, 112000 },{ 50000, 48050, 56000 },{ 42000, 41050, 56000 } }	{ { 21, 11, 11, 1 },{ 21, 9, 9, 1 },{ 21, 7, 7, 1 } }	1	0
256	1	3	4	0	{ { 90000, 85000, 112000 },{ 70000, 69000, 112000 },{ 50000, 48950, 56000 },{ 36400, 35600, 56000 } }	{ { 31, 1, 1, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
256	2	3	4	0	{ { 84650, 83000, 112000 },{ 65850, 64550, 56000 },{ 47000, 46100, 48000 },{ 28200, 27650, 40000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
256	3	3	4	0	{ { 76300, 73550, 112000 },{ 59350, 57200, 56000 },{ 42400, 40850, 48000 },{ 25450, 24500, 40000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 }, { 1, 1, 1, 1 } }	1	2

384	1	3	4	0	{ { 128000, 128000, 128000 },{ 100000, 100000, 128000 },{ 79850, 79850, 104000 },{ 66600, 66600, 104000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
384	2	3	4	0	{ { 128000, 128000, 128000 },{ 105350, 103300, 112000 },{ 75200, 73750, 96000 },{ 45100, 44250, 48000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
384	3	3	4	0	{ { 124300, 121550, 128000 },{ 96700, 94550, 112000 },{ 69050, 67500, 96000 },{ 41450, 40500, 48000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
512	1	3	4	0	{ { 128000, 128000, 128000 },{ 128000, 128000, 128000 },{ 128000, 128000, 128000 }, {118450, 118450, 128000 } }	{ { 31, 1, 1, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
512	2	3	4	0	{ { 124000, 124000, 128000 },{ 124000, 124000, 128000 },{ 125200, 118450, 128000 },{ 76300, 73000, 128000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2
512	3	3	4	0	{ { 118000, 118000, 128000 },{ 118000, 118000, 128000 },{ 117200, 109250, 128000 },{ 72300, 69000, 128000 } }	{ { 31, 11, 11, 1 },{ 1, 1, 1, 1 },{ 1, 1, 1, 1 } }	1	2

5.4.3.7.12 Quantization and coding process

5.4.3.7.12.1 Overview

When DTX is OFF or $W_{vad} = 1$, quantization strategies are read from the SPAR bitrate distribution table entry corresponding to the given IVAS bitrate, bandwidth value 3 and SBA analysis order. Then the number of quantization points for PR, CP and D coefficients in each of the quantization strategy is read. All SPAR parameters are uniformly quantized within the quantization range of parameters. The quantization range of PR coefficients (PR_{min}, PR_{max}), CP coefficients (CP_{min}, CP_{max}) and D coefficients (D_{min}, D_{max}) is computed as follows.

$$PR_{min} = \begin{cases} -1.2 & if\ active_w == 1 \\ -1 & if\ active_w == 0 \end{cases}, \quad PR_{max} = \begin{cases} 1.2 & if\ active_w == 1 \\ 1 & if\ active_w == 0 \end{cases} \tag{5.4-65}$$

$$CP_{min} = \begin{cases} -0.8 & if\ active_w == 1 \\ -2 & if\ active_w == 0 \end{cases}, \quad CP_{max} = \begin{cases} 0.8 & if\ active_w == 1 \\ 2 & if\ active_w == 0 \end{cases} \tag{5.4-66}$$

$$D_{min} = \begin{cases} 0 & if\ active_w == 1 \\ 0 & if\ active_w == 0 \end{cases}, \quad D_{max} = \begin{cases} 0.8 & if\ active_w == 1 \\ 1 & if\ active_w == 0 \end{cases} \tag{5.4-67}$$

where $active_w$ is the active W flag that is read from the bitrate distribution table.

The quantized coefficient value and the corresponding index to be coded in metadata bitstream is computed as follows.

$$A^{index} = round\left(\frac{((qlvl_A - 1)\, max(A_{min}, min(A_{max}, A)))}{A_{max} - A_{min}}\right) \tag{5.4-68}$$

$$A^{quantized} = A^{index}\left(\frac{A_{max} - A_{min}}{(qlvl_A - 1)}\right) \tag{5.4-69}$$

where A is a SPAR coefficient (either PR or CP or D) and $qlvl_A$ is the number of quantization points as per the quantization strategy read from SPAR bitrate distribution table.

The quantization and coding of SPAR parameters is then done in the quantization and coding loop and the steps performed in the loop are bitrate dependent as described below.

5.4.3.7.12.2 Quantization and coding loop at all bitrates with FOA input and at bitrates 192 kbps and below with HOA2 and HOA3 input

At these bitrates, the quantization and coding are done in following steps.

Step 1: The number of quantization points, for finest quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_D$.

Step 2: All SPAR parameters, that is PR, CP and D coefficients, are calculated without quantization as described clauses 5.4.3.7.6 - 5.4.3.7.9.

Step 3: D coefficients are quantized as per $qlvl_D$ and Equations (5.4-68) and (5.4-69). This is done so that D coefficients are computed as per unquantized PR and CP coefficients and do not make up for the quantization noise in PR and CP coefficients.

Step 4: PR coefficients are quantized as per $qlvl_{PR}$ and Equations (5.4-68) and (5.4-69). At IVAS bitrates 13.2 kbps and 16.4 kbps, the quantized PR coefficients are further modified as per Equations (5.4-70) and (5.4-71).

Step 5: CP coefficients are recomputed, as described in clause 5.4.3.7.8, by substituting quantized PR coefficients instead of unquantized PR coefficients in Equation (5.4-55). This is done so that CP coefficients are computed based on the quantized PR coefficients that are used to generate downmix signals.

Step 6: CP coefficients are quantized as per $qlvl_{CP}$ and Equations (5.4-68) and (5.4-69).

Step 7: All the quantized indices are coded non-differentially with Arithmetic coder as described in clause 5.4.3.7.16 and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{nd\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}$$

where $spar_{CSbits} = 3$, $spar_{qbits} = \begin{cases} 2 & if\ IVAS\ bitrate < 256 \text{ kbps} \\ 1 & otherwise \end{cases}$.

This non-differential arithmetic coding scheme is also referred to as BASE coding scheme.

Step 8: If $MDbits_{act}^{nd\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{nd\_ent} > MDbits_{tar}$, then loop continues with step 9.

Step 9: All the quantized indices are coded non-differentially with base2 coding, that is implemented using Huffman coder with a near flat probability distribution model, as described in clause 5.4.3.7.17 and the actual SPAR MD bits are computed based on Huffman coder coded bits, $bits_{base2}$, as

$$MDbits_{act}^{base2} = spar_{CSbits} + spar_{qbits} + bits_{base2}.$$

This non-differential Huffman coding scheme is also referred to as BASE_NOEC coding scheme.

Step 10: If $MDbits_{act}^{base2} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{base2} > MDbits_{tar}$, then loop continues with step 11. Both non-differential coding schemes are first tried before time-differential coding so that the loop can exit if the bitrate from non-differential schemes fits within $MDbits_{tar}$ and time differential coding can be avoided as that improves quality during packet loss concealment at decoder. At bitrates 256 kbps and above, with FOA input, SPAR bitrate distribution table is tuned such that $MDbits_{act}^{base2} \leq MDbits_{tar}$ is always true.

Step 11: Processing at this step is bitrate dependent. At IVAS bitrates 24.4 kbps and above, all the quantized indices are coded using a time differential coding scheme (as described in clause 5.4.3.7.14) with Arithmetic coder as described in clause 5.4.3.7.16 and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{td\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}.$$

At IVAS bitrates 13.2 kbps and 16.4 kbps, all the quantized indices are coded using band interleaving and 40 ms MD update rate coding scheme (as described in clause 5.4.3.7.15) with Arithmetic coder and the actual SPAR MD bits are computed based on arithmetic coder coded bits, $bits_{arithmetic}$, as

$$MDbits_{act}^{td\_ent} = spar_{CSbits} + spar_{qbits} + bits_{arithmetic}.$$

Step 12: If $MDbits_{act}^{td\_ent} \leq MDbits_{tar}$, then the quantization and coding loop is exited, outputting the MD bitstream. If $MDbits_{act}^{td\_ent} > MDbits_{tar}$, then loop continues with step 13.

Step 13: Bitstream with minimum MD bitrate is selected as follows.

$$MDbits_{act} = min(MDbits_{act}^{nd\_ent}, MDbits_{act}^{base2}, MDbits_{act}^{td\_ent})$$

If $MDbits_{act} \leq MDbits_{max}$, then the quantization and coding loop is exited, outputting the MD bitstream corresponding to $MDbits_{act}$. If $MDbits_{act} > MDbits_{max}$, then the number of quantization points for next quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_{D}$ and the control goes back to Step 2. At 256 kbps and above, SPAR bitrate distribution table is tuned such that the quantization loop exits at this step.

5.4.3.7.12.3 Quantization and coding loop at bitrates 256 kbps and 384 kbps with HOA2 and HOA3 input

At IVAS bitrates 256 kbps and 384 kbps with HOA2 and HOA3 input, SBA coding architecture is as described in clauses 5.4.2.3 and 5.4.2.4. At these bitrates, SPAR MD corresponding to FOA channels is coded for 1 to nB_{foa} frequency bands, where $nB_{foa} = 8$, whereas SPAR MD corresponding to HOA2 and HOA3 channels is coded for 1 to nB_{md} frequency bands. SPAR MD corresponding to FOA channels, for frequency bands $nB_{foa} + 1$ to nB_{md} is computed from DirAC MD. The quantization and coding are done in following steps:

Step 1: The number of quantization points, for finest quantization strategy are read as $qlvl_{PR}, qlvl_{CP}, qlvl_{D}$.

Step 2: All SPAR parameters, including PR (prediction), CP (cross prediction) and D (decorrelation) coefficients, are calculated without quantization.

Step 3: D coefficients are quantized as per $qlvl_{D}$ and Equations (5.4-68) and (5.4-69). This is done so that D coefficients are computed as per unquantized PR and CP coefficients and do not make up for the quantization noise in PR and CP coefficients.

Step 4: For frequency bands 1 to nB_{foa}, PR coefficients corresponding to FOA channels are quantized as per $qlvl_{PR}$ and Equations (5.4-68) and (5.4-69).

For frequency bands $nB_{foa} + 1$ to nB_{md}, PR coefficients corresponding to FOA channels are neither quantized nor coded as they are generated from DirAC MD.

For frequency bands 1 to nB_{md}, PR coefficients corresponding to HOA2 and HOA3 channels are quantized as per $qlvl_{PR}$ and Equations (5.4-68) and (5.4-69).

Step 5: For frequency bands 1 to nB_{foa}, CP coefficients are recomputed by substituting quantized PR coefficients instead of unquantized PR coefficients in Equation 5.4-55. This is done so that CP coefficients are computed based on the quantized PR coefficients that are used to generate downmix signals.

For frequency bands $nB_{foa} + 1$ to nB_{md}, CP coefficients are recomputed from a combination of DirAC metadata and input covariance C^{md} given in Equation (5.4-36). This is done by using DirAC to SPAR converted PR coefficients in Equation (5.4-55).

Step 6: CP coefficients are quantized as per $qlvl_{CP}$ and Equations (5.4-68) and (5.4-69).

Step 7 to step 13 are same as clause 5.4.3.7.12.2.

If $MDbits_{act} \leq MDbits_{max}$, then the quantization and coding loop is exited, outputting the MD bitstream corresponding to $MDbits_{act}$. At 256 kbps and above, SPAR bitrate distribution table is tuned such that the quantization loop exits at this step.

5.4.3.7.12.4 Quantization and coding loop at bitrates 512 kbps with HOA2 and HOA3 input

At IVAS bitrates 512 kbps with HOA2 and HOA3 input, SBA coding architecture is as described in clause 5.4.2.5. At this bitrate, DirAC MD is not converted to SPAR MD at any frequency band and SPAR MD corresponding to all the SBA analysis channels, for frequency bands 1 to nB_{md} is computed, quantized and coded in metadata bitstream. The quantization and coding loop is same as described in clause 5.4.3.7.12.2.

5.4.3.7.13 Spectral filling in the side channels at 13.2 kbps and 16.4 kbps

At 13.2 and 16.4 kbps, PR and D coefficients may be quantized to 0 even when unquantized PR coefficients are non-zero and the variance of corresponding side channel input to SPAR is also non-zero. This happens due to fewer number of quantization points and large quantization steps and can lead to spectral holes in the side channels at the output of up-mixer at decoder. To avoid spectral holes due to quantization of PR and D coefficients, quantized PR coefficients are further modified as follows.

For each b in range 1 to nB_{md} and for each j in range 2 to N_{spar_ana}, if $PR_{j-1}^{quantized}(b)$ and the corresponding $D^{quantized}(b)$ are zero then $PR_{j-1}^{quantized}(b)$ and $PR_{j-1}^{index}(b)$ are recomputed as given below.

$$PR_{j-1}^{quantized}(b) = \begin{cases} q_{step} \quad if\ PR_{j-1}(b) > 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ -q_{step}\ \ if\ PR_{j-1}(b) < 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ PR_{j-1}^{quantized}(b), \qquad\qquad otherwise \end{cases} \tag{5.4-70}$$

where $q_{step} = \frac{PR_{max} - PR_{min}}{(qlvl_{PR} - 1)}$ and

$$PR_{j-1}^{index}(b) = \begin{cases} 1 \quad if\ PR_{j-1}(b) > 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ -1\ \ if\ PR_{j-1}(b) < 0\ , C_{[1,j]}^{md}(b)\ != \ 0 \\ PR_{j-1}^{index}(b), \qquad\qquad otherwise \end{cases} \tag{5.4-71}$$

5.4.3.7.14 Time differential coding

5.4.3.7.14.1 Overview

SPAR uses time differential coding scheme to encode quantized indices of SPAR coefficients. Table 5.4-9 shows the 4 modes of time differential coding scheme, that is 4a, 4b, 4c, 4d, here "base" refers to non-differential coding schemes. Each mode codes 3/4th of the total bands differentially while remaining bands are coded non-differentially. More specifically, the band specified by 0 is coded non time differentially and the band specified by 1 is coded time differentially. The time differential index is generated by taking the difference between quantized parameter index in current frame and quantized parameter index of the previous frame, applying modulo $qlvl_A$ operation on the difference value as described in Equations (5.4-72) and (5.4-73), here $qlvl_A$ is the number of quantization points as per the quantization strategy read from SPAR bitrate distribution table.

Set of bands to be coded time differentially changes on frame-by-frame basis, Table 5.4-10 shows the mapping of time differential coding mode between previous frame and current frame. All 4 modes of time differential coding are such that complete recovery is possible within 4 frames on the decoder side in the event of packet loss. Coding scheme is set to "base" for the first frame.

Table 5.4-9: Interleaved time differential coding schemes

Coding Scheme	Time Diff Coding, Bands 1-12	Time Diff Coding, Bands 1-8
base or base_noec	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
4a	0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1
4b	1 0 1 1 1 0 1 1 1 0 1 1	1 0 1 1 1 0 1 1
4c	1 1 0 1 1 1 0 1 1 1 0 1	1 1 0 1 1 1 0 1
4d	1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0

Table 5.4-10: Mapping of the time differential coding schemes

Previous frame's coding Scheme	Current frame's time differential coding scheme
base or base_noec	4a
4a	4b
4b	4c
4c	4d
4d	4a

5.4.3.7.14.2 Time differential coding across the same quantization strategy

Time differential index is computed in two steps. First the difference between current frame's quantized parameter index and previous frame's quantized parameter index is computed as follows:

$$A_{TD}^{index} = A_{current}^{index} - A_{previous}^{index} \quad (5.4\text{-}72)$$

Then modulo $qlvl_A$ is performed to keep the index range as $\{A_{min}^{index}, A_{max}^{index}\}$

$$A_{TD}^{index} = \begin{cases} A_{TD}^{index} + qlvl_A & if\ A_{TD}^{index} < A_{min}^{index} \\ A_{TD}^{index} - qlvl_A & if\ A_{TD}^{index} > A_{max}^{index} \\ A_{TD}^{index} & otherwise \end{cases}, \quad (5.4\text{-}73)$$

where $A_{min}^{index} = round\left(\frac{((qlvl_A-1)*A_{min})}{A_{max}-A_{min}}\right)$, $A_{max}^{index} = round\left(\frac{((qlvl_A-1)*A_{max})}{A_{max}-A_{min}}\right)$, A refers to PR, CP and D coefficients and $qlvl_A$ is the number of quantization points as per the current frame's quantization strategy read from SPAR bitrate distribution table. A_{min} and A_{max} are the minimum and maximum quantized values as per the current frame's quantization strategy as per Equations (5.4-65), (5.4-66) and (5.4-67). A_{TD}^{index} is then coded with arithmetic coder.

5.4.3.7.14.3 Time differential coding across different quantization strategies

If, for the same IVAS bitrate, the quantization strategy in the previous frame is different than the quantization strategy in the current frame then the quantized indices from previous frame are first mapped to the quantization strategy of the current frame. This allows time-differential coding between frames without resorting to having to send a non-time-differential frame each time the quantization scheme is changed. The mapping of the indices is done as follows.

$$A_{previous,mapped}^{index} = round\left(\frac{A_{previous}^{index}(qlvl_{A,current}-1)}{qlvl_{A,previous}-1}\right) \quad (5.4\text{-}74)$$

Then the difference between current frame's quantized parameter index and previous frame's mapped quantized parameter index is computed as follows.

$$A_{TD}^{index} = A_{current}^{index} - A_{previous,mapped}^{index} \quad (5.4\text{-}75)$$

Then modulo $qlvl_A$ is performed as per Equation (5.4-73). After the modulo operation, A_{TD}^{index} is coded with arithmetic coder.

5.4.3.7.15 Band interleaving and 40ms MD update rate at 13.2 kbps and 16.4 kbps

At 13.2 and 16.4 kbps, SPAR operates at a 40ms metadata time resolution, with occasional metadata updates at a 20ms time resolution where permissible by bitrate limitations. For the first frame, or frames where the metadata can be coded within the target metadata bitrate budget, all bands are coded with one of the two non-differential coding schemes described in clause 5.4.3.7.12.2. For frames where this limitation is not met, metadata corresponding to half of the bands is sent in the current frame, followed by the other half in the next frame, and so on. These interleaved bands are coded with non-differential arithmetic coding scheme. The choice of which bands to send or omit in each frame is interleaved as shown in Table 5.4-11. With the 40 ms metadata time resolution, bands labelled as B are sent in one frame whereas bands labelled as A are sent in the next frame. Coding of band-interleaved modes in done by reusing time-differential coding modes. The mapping between time-differential coding modes and band interleaved modes is shown in Table 5.4-12. The band group (A or B) that is being sent on the current frame is therefore indicated to the decoder using the $spar_{CSbits}$, which indicate the coding scheme as described in clause 5.4.3.7.11.

Table 5.4-11: Frequency band interleaved 40ms metadata update rate coding scheme

	SPAR, Bands 1 to nB_{md}
20 ms update rate frames	Y Y Y Y
40 ms update rate frames	B A B A

Table 5.4-12: Mapping of time differential coding modes to band interleaved coding modes

Coding Scheme	SPAR, Bands 1 to nB_{md}
4a	Arithmetic coding of only the A bands.
4b	Arithmetic coding of only the B bands.
4c	Arithmetic coding of only the A bands.
4d	Arithmetic coding of only the B bands.

5.4.3.7.16 Entropy coding with Arithmetic coders

SPAR uses arithmetic coders to entropy-code the quantized indices. Non-differential indices are coded with different probability distribution models as compared to time differential indices. Table 5.4-13, Table 5.4-14 and Table 5.4-15 contains the probability distribution models for PR, CP and D coefficients in time differential and non-differential mode. There are four probability distribution models corresponding to each $qlvl$ that is read from SPAR bitrate distribution table. One out of the four probability distribution models are chosen dynamically on frame-by-frame basis based on the actual symbol distribution in the quantized indices. First, an estimate of coded bits is generated with each probability distribution model as shown below.

For each i in range $1 \leq i \leq 4$

$$bit_i = -1 * \sum_{k=1}^{k=length_{indices}} \left(dist_{curr,k} \log_2 \left(\frac{max(\varepsilon, dist_{model,i,k+1})}{\sum_{j=1}^{j=length_{indices}} dist_{model,i,j+1}} \right) \right) \quad (5.4\text{-}76)$$

where $length_{indices}$ is the total number of indices to code as per Table 5.4-13, Table 5.4-14 and Table 5.4-15, $dist_{curr,k}$ is the number of occurrences of the kth index in the symbol distribution of quantized SPAR parameter indices in current frame, $dist_{model,i,j+1}$ is the value of (j+1)th index of ith probability distribution model.

Then, the probability distribution model with minimum value of bit_i is selected to code the quantized indices and i is coded with 2 bits in the bitstream as part of Arithmetic coder bits.

Table 5.4-13: Probability distribution models for PR coefficients indices

qlvls from SPAR bitrate distribution table	Active w from SPAR bitrate distribution table	Models for non-differential entropy coding	Models for time differential entropy coding	Indices to code
7	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-3,-2,-1,0,1,2,3
15	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7
21	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10
31	0	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-15,-14,-13,-12,-11,-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
7	1	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-3,-2,-1,0,1,2,3
15	1	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7
21	1	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10

Table 5.4-14: Probability distribution models for CP coefficients indices

qlvls from SPAR bitrate distribution table	Models for non-differential entropy coding	Models for time differential entropy coding	Indices to code
7	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-3,-2,-1,0,1,2,3
9	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-4,-3,-2,-1,0,1,2,3,4
11	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	-5,-4,-3,-2,-1,0,1,2,3,4,5

Table 5.4-15: Probability distribution models for D coefficients indices

qlvls from SPAR bitrate distribution table	Models for non-differential entropy coding	Models for time differential entropy coding	Indices to code in non-differential mode	Indices to code in differential mode
3	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2	-1,0,1
5	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4	-2,-1,0,1,2
7	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4,5,6	-3,-2,-1,0,1,2,3
9	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4,5,6,7,8	-4,-3,-2,-1,0,1,2,3,4
11	{model 1}, {model 2}, {model 3}, {model 4}	{model 1}, {model 2}, {model 3}, {model 4}	0,1,2,3,4,5,6,7,8,9,10	-5,-4,-3,-2,-1,0,1,2,3,4,5

5.4.3.7.17 Base2 coding with Huffman coder

Huffman coding is done with a near flat probability distribution model such that bits coded with Huffman coder are always less than OR equal to bits coded with base2 coding. Table 5.4-16, Table 5.4-17 and Table 5.4-18 contains the Huffman codebook for every $qlvl$ value that can be read from SPAR bitrate distribution table. Every entry in codebook has three fields {quantized index value, bits for the index, code for the index}.

Table 5.4-16: Huffman codebook for PR coefficients indices

qlvls from SPAR bitrate distribution table	codebook
7	{Huffman codebook}
15	{Huffman codebook}
21	{Huffman codebook}
31	{Huffman codebook}

Table 5.4-17: Huffman codebook for CP coefficients indices

qlvls from SPAR bitrate distribution table	codebook
7	{Huffman codebook}
9	{Huffman codebook}
11	{Huffman codebook}

Table 5.4-18: Huffman codebook for D coefficients indices

qIvls from SPAR bitrate distribution table	codebook
3	{Huffman codebook}
5	{Huffman codebook}
7	{Huffman codebook}
9	{Huffman codebook}
11	{Huffman codebook}

5.4.3.8 DirAC and SPAR parameter merge

5.4.3.8.1 General

SBA coding uses a combination of DirAC and SPAR coding as described in Figure 5.4-2, Figure 5.4-3, Figure 5.4-4 and Figure 5.4-5. For FOA channels, SPAR and DirAC parameters are integrated based on a frequency-split-based approach wherein SPAR metadata is generated for low frequency parameter band index 1 to 8 whereas DirAC metadata is generated for high frequency parameter band index 9 to 12. In order to generate SPAR downmix and upmix matrix in DirAC bands, DirAC metadata is converted to SPAR metadata as described in clause 5.4.3.8.2.

At 13.2 kbps and 16.4 kbps SPAR bands are merged to 4 bands and DirAC bands are merged to 2 bands to reduce the overall metadata bitrate as described in clauses 5.4.3.4 and clause 5.4.3.7.

When DTX is ON and $W_{vad} = 0$, SPAR bands are merged to 2 frequency bands and DirAC bands are merged to 2 frequency bands to reduce the overall metadata bitrate as described in clause 5.4.9.

At 256 kbps and 384 kbps some HOA channels are part of the analysed signal vector $s^{sba\_ana}(m, n)$. In this case SPAR and DirAC parameters are integrated based on a channel-split approach. SPAR metadata is generated for planar HOA2 and planar HOA3 channels and all parameter bands, whereas DirAC metadata is generated to enable the reconstruction of the remaining channels on the decoder side.

At 512 kbps, SPAR metadata is generated for FOA, HOA2 and planar HOA3 channels and all parameter bands. For the reconstruction of the remaining channels, HO-DirAC metadata is generated from the FOA and HOA2 input channels. Input channel indices for metadata analysis in SPAR and DirAC are given in Table 5.4-2.

5.4.3.8.2 DirAC to SPAR parameter conversion

For the 4 high frequency parameter bands, a model-based approach is employed to derive the SPAR parameters and the downmix matrix from the psycho-acoustic model parameters of DirAC. This enables the re-use of these parameters from the DirAC encoder according to clause 5.4.3.4. They are transmitted in the bitstream for these bands in order to enable the reconstruction of the HOA channels on the decoder side. The need to transmit additional metadata for the upmix in these bands can be avoided.

The calculation of the SPAR parameters (prediction, cross-prediction, and decorrelation coefficients) from the covariance matrix is performed independently on the encoder and decoder side. First, the input covariance matrix is estimated from quantized DirAC parameters and quantized SPAR decorrelation (D) coefficients depending on the number of downmix channels. Specifically, the calculation of the covariance matrix is based on the real spherical harmonics. and is detailed in the following clauses. Then SPAR MD (PR, CP, and D coefficients) is computed from covariance matrix as per clause 5.4.3.7.5.

5.4.3.8.3 Covariance estimation from DirAC parameters

5.4.3.8.3.1 Covariance estimation when $N_{dmx} > 2$

For the FOA channels w, x, y, z the matrix elements in the DirAC bands are given by

$$C_{x;y;z,w} = E^{\mathrm{dir}} Y_{0,0}(\theta_D) Y_{1,-1;0;1}(\theta_D) \quad (5.4\text{-}77)$$

and

$$C_{w,w} = E^{\mathrm{dir}} Y_{0,0}(\theta_D) Y_{0,0}(\theta_D) + \ E_w^{\mathrm{diff}}. \tag{5.4-78}$$

This result can be understood from the model of a single point source panned in the ambisonics domain to the DoA given by the angle θ_D. The inter-channel covariance is then given by

$$C_{x;y;z,w} = \int dt\ s^2(t) Y_{0,0}(\theta_D) Y_{1,-1;0;1}(\theta_D) \tag{5.4-79}$$

where s(t) is the time-dependent scalar-valued sound signal emitted by the point source. The signal energy appears due to the integrals over the audio signal s(t). Using the diffuseness parameter Ψ, these energies can be eliminated and expressed as $E^{dir} = (1 - \Psi)E$ and $E_w^{\mathrm{diff}} = \frac{\Psi}{f_{norm}}$ with the normalization factor f_{norm} that depend on the degree of the spherical harmonic. This yields:

$$C_{w,w} = (1 - \Psi)\ E\ Y_{0,0}(\theta_D) Y_{0,0}(\theta_D) + \Psi\ E \tag{5.4-80}$$

and for the channels for the degree l = 1

$$C_{x;y;z,x;y;z} = (1 - \Psi)\ E\ Y_{1,-1;0;1}(\theta_D) Y_{1,-1;0;1}(\theta_D) + \frac{\Psi}{3}\ E, \tag{5.4-81}$$

and for the channels for the degree l = 2

$$C_{j,j} = (1 - \Psi)\ E\ Y_j(\theta_D) Y_j(\theta_D) + \frac{\Psi}{5}\ E, \tag{5.4-82}$$

and for those of the degree l = 3

$$C_{j,j} = (1 - \Psi)\ E\ Y_j(\theta_D) Y_j(\theta_D) + \frac{\Psi}{7}\ E, \tag{5.4-83}$$

The index j is a combined index of the real spherical harmonic with and degree l and index m.

The covariance matrix, hence, depends on these model parameters and the downmix matrix, which is employed to transform the audio channels. It is calculated in a signal-adaptive way based on model parameters derived from the input audio channels.

Note that the metadata of the acoustic model are quantized prior to the application of Equations (5.4-77) to (5.4-83). This is necessary to ensure exact agreement of the covariance matrices calculated on the encoder and decoder side. They are also encoded in the metadata encoder according to clause 5.2.4.

The calculation of the covariance and downmix matrices is performed individually on each parameter band of the filterbank-domain representation of the input signal, $\boldsymbol{s}^{sba_{ana},mdft}(m,k)$, which is obtained from the analysis filterbank according to clause 5.4.3.3. The responses ($Y_{lm}(\theta_D)$) in Equations (5.4-77) to (5.4-83) are averaged over all subframes in each frame. This is necessary because the time resolution of the DoA angles is one subframe (5 ms) while the covariance matrix is evaluated with a resolution of one frame (20 ms) in the calculation of the SPAR parameters.

The above processing is applied to the high-frequency parameter bands (DirAC). For the low-frequency parameter bands (SPAR) the mixing matrix is derived differently: the prediction coefficients, cross-prediction coefficients and decorrelation coefficients are quantized and written to the bitstream directly.

For the low-order operation mode, only a single DoA angle θ_D is estimated by a single parameter estimator. This is then used in Equations (5.4-77) to (5.4-83). In the high-order operation mode, two sector DoA angles are estimated. In this case, the simple model of a single point source is still employed in the covariance estimation. For a single point source, the two sector DoAs are equal. Hence, the DoA of the zeroth sector is used in Equations (5.4-77) to (5.4-83).

For configurations with 2 or less channels in the first set of transport channels ($N_{dmx} \leq 2$), the formulas are modified to account for the missing diffuse energy. This results in the formulas in clauses 5.4.3.8.3.2 and 5.4.3.8.3.3.

5.4.3.8.3.2 Covariance estimation when $N_{dmx} \leq 2$ and $W_{vad} = 1$

For each DirAC band, covariance is estimated as

$$C_{i,j}^{DiRAC} = \begin{cases} (1 - \psi)^2 \gamma_i \gamma_j + (1 - (1 - \psi)^2) D_{i-N_{dmx}}^{norm} & if\ i = j\ , i > N_{dmx} \\ (1 - \psi)^2 \gamma_i \gamma_j + \frac{(1-(1-\psi)^2)}{3} & if\ i = j\ , 1 < i \leq N_{dmx} \\ 1 & if\ i = j = 1 \\ (1 - \psi)\gamma_i \gamma_j & if\ i \neq j \end{cases} \tag{5.4-84}$$

Computation of γ_i is as follows.

$$\gamma_i = \frac{\sum_{m=1}^{m=N_{sf}} \gamma_{m,i}}{N_{sf}} \quad (5.4\text{-}85)$$

where $\gamma_{m,i}$ is the spherical harmonic response for first order ambisonics with ACN ordering and SN3D normalization such that $\gamma_{m,1} = 1$, $\gamma_{m,2} = \sin\theta_m \cos\emptyset_m$, $\gamma_{m,3} = \sin\emptyset_m$, $\gamma_{m,4} = \cos\theta_m \cos\emptyset_m$, m is the subframe index with in a frame, N_{sf} is the number of subframes corresponding to which DirAC metadata is quantized and coded in IVAS bitstream, θ_m is quantized DirAC azimuth angle in subframe m, and $\emptyset_m$ is quantized DirAC elevation angle in subframe m.

Normalized D coefficient D_k^{norm} is computed as follows.

$$D_k^{norm} = \frac{D_k^2(nB_{md})}{\max\left(\varepsilon, \left(\sum_{k=1}^{k=(4-N_{dmx})} D_k^2(nB_{md})\right) * \frac{3}{\min(3, \max(0, 4-N_{dmx}))}\right)}, 1 \le k \le (4 - N_{dmx}) \quad (5.4\text{-}86)$$

where $D_k(nB_{md})$ are the SPAR D coefficients in the last SPAR band nB_{md} as computed in Equation (5.4-62). Ψ is the quantized DirAC diffuseness parameter.

5.4.3.8.3.3 Covariance estimation when $N_{dmx} \le 2$ and $W_{vad} = 0$

For each DirAC band, covariance is estimated as

$$C_{i,j}^{DiRAC} = \begin{cases} (1 - \psi)^2 \gamma_i \gamma_j + \frac{(1-(1-\psi)^2)}{3} & if\ i = j\ , i > 1 \\ 1 & if\ i = j\ = 1 \\ (1 - \psi)\gamma_i\ \gamma_j & if\ i \ne j \end{cases} \quad (5.4\text{-}87)$$

where i and j are FOA channel indices ranging from 1 to 4, Ψ is the quantized DirAC diffuseness parameter.

5.4.3.8.4 Directional diffuseness

The diagonal elements in the covariance matrix corresponding to the side channels of FOA signal, as given in Equation (5.4-81), are the sum of directional component, that is computed as a product of energy ratio and spherical harmonics response, and diffuse component in DirAC MD, which is the normalized DirAC diffuseness parameter.

At low bitrates where $N_{dmx} \le 2$, the diffuse component is computed by normalizing the diffused energy from DirAC with normalized SPAR D coefficients D_k^{norm} and the covariance matrix is estimated based on quantized DirAC parameters and quantized SPAR D coefficients as given in Equation (5.4-86).

5.4.3.8.5 SPAR parameter estimation from DirAC parameters

PR, CP and D coefficients corresponding to FOA channels and DirAC frequency bands, also referred to as PR^{DirAC}, CP^{DirAC} and D^{DirAC}, are computed by replacing C^{md} matrix with C^{DirAC} matrix in clauses 5.4.3.7.6 - 5.4.3.7.9. DirAC to SPAR converted parameters are not quantized as they are computed based on quantized DirAC parameters that are transmitted to the decoder.

5.4.3.8.6 Active W prediction from DirAC parameters

In DirAC frequency bands, when active W prediction mode is enabled, primary downmix channel W' is generated by scaling the FOA signal and adding them together as per Equation (5.4-38). The active W gains ($s_{w,}\ s_y, s_z,\ s_x$) are computed from quantized DirAC metadata parameters by computing the prediction downmix matrix given in Equation (5.4-41) from the covariance matrix, that is computed in Equation (5.4-84) and Equation (5.4-87). The parameters of prediction downmix matrix given in Equation (5.4-41) are estimated from DirAC parameters as given follows.

$$\hat{u} = \begin{pmatrix} \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix}, \alpha = (1 - \psi), U = C_{[2:4,2:4]}^{DiRAC}$$

5.4.3.8.7 Passive W prediction from DirAC parameters

In the case that $N_{dmx} = N_{sba\_ana}$, the downmix generation simplifies strongly. Specifically, the downmix matrix only compresses the audio channels by prediction. The prediction matrix is formulated from the elements of covariance matrix as

$$\boldsymbol{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -C_{x,w}/C_{w,w} & 1 & 0 & 0 \\ -C_{y,w}/C_{w,w} & 0 & 1 & 0 \\ -C_{z,w}/C_{w,w} & 0 & 0 & 1 \end{pmatrix}, \quad (5.4\text{-}88)$$

which removes the correlations between the X, Y, Z (or higher order) ambisonics channels and the W channel. Additional matrices can be constructed to model the uncorrelated parts with decorrelators on the decoder side. These matrices together comprise the downmix matrix in the encoder.

At low bitrates, the processing of the ambisonics input channels is limited to the subset of first-order channels. At higher bitrates (≥ 256 kbps), the set of channels transformed includes the planar channels of the second and third order in addition.

5.4.4 Downmix matrix calculation

5.4.4.1 SPAR parameters to downmix matrix conversion

In SPAR bands, that is band index 1 to nB_{md} , the downmix matrix, $dmx_{[N_{dmx}\ x\ 4]}$, is generated as follows. First, the prediction downmix matrix is generated as per Equation (5.4-55) but with quantized PR coefficients, $PR^{quantized}_{[N_{PR}\ x\ 1]}(b)$, as follows.

$$PRdmx_{[N_{spar\_ana} x N_{spar\_ana}]}(b) = \begin{pmatrix} 1 & Zero_{[1x\ N_{PR}]} \\ -PR^{quantized}_{[N_{PR}\ x\ 1]}(b) & I_{[N_{PR} x\ N_{PR}]} \end{pmatrix} \begin{pmatrix} W_{scale} & W_{scale} F \hat{u}_{[1xN_{PR}]}{}^{H} \\ Zero_{[N_{PR}\ x\ 1]} & I_{[N_{PR} x\ N_{PR}]} \end{pmatrix} \quad (5.4\text{-}89)$$

At 13.2 kbps and 16.4 kbps, in band interleaving mode, half of the frequency bands of $PRdmx$ are generated from previous frame's matrix as described in clause 5.4.4.3.

Then $PRdmx$ is remixed as per Equation (5.4-59). After remixing, $PRdmx$ is limited to actual number of downmix channels N_{dmx} as shown below:

$$dmx''_{[i,j]} = PRdmx_{[i,j]} \quad (5.4\text{-}90)$$

where $1 \le i \le N_{dmx}$ and $1 \le j \le 4$.

Then band unmixing is performed on dmx'' as follows:

$$dmx(i + (b1 - 1)bw_{bands}) = dmx''(b1) \quad (5.4\text{-}91)$$

where $1 \le b1 < nB_{md}$ and $1 \le i \le bw_{bands}$.

In DirAC bands, the downmix matrix, $dmx_{[N_{dmx}\ x\ 4]}$, is generated as follows. First, the prediction downmix matrix is generated as per Equation (5.4-55) but with PR coefficients, $PR^{DirAC}_{[N_{PR}\ x\ 1]}(b)$, as follows.

$$PRdmx_{[4x4]}(b) = \begin{pmatrix} 1 & Zero_{[1x\ 3]} \\ -PR^{DirAC}_{[3\ x\ 1]}(b) & I_{[3x\ 3]} \end{pmatrix} \begin{pmatrix} W_{scale} & W_{scale} F \hat{u}_{[1x3]}{}^{H} \\ Zero_{[3\ x\ 1]} & I_{[3x\ 3]} \end{pmatrix} \quad (5.4\text{-}92)$$

where $PR^{DirAC}_{[N_{PR}\ x\ 1]}(b)$, W_{scale}, F and $\hat{u}_{[1x3]}$ are computed by replacing C^{md} matrix with C^{DirAC} matrix in clauses 5.4.3.7.6 - 5.4.3.7.9.

Then $PRdmx$ is remixed as per Equation (5.4-59). After remixing, $PRdmx$ is limited to actual number of downmix channels N_{dmx} as shown in Equation (5.4-90) to generate , $dmx(b)$ in DirAC bands.

In one channel downmix mode, that is $N_{dmx} = 1$, the $PRdmx$ that is computed in Equation (5.4-92) is further scaled to compensate for the energy difference between C^{md} and C^{DirAC} as described in clause 5.4.4.2.

5.4.4.2 Energy compensation in DirAC bands

In DirAC bands, when number of downmix channels is one ($N_{dmx} = 1$), primary downmix channel W' is further scaled with an energy compensation factor to prevent spatial collapse by scaling the downmix signal such that the upmixed signal is better energy matched with respect to the input. The scaling is computed in each DirAC band index b as shown below.

$$scale(b) = \max\left(1, min\left(2, \sqrt{tr(C_{norm}{}^{in}_{[4x4]}(b))/max(\varepsilon, tr(C^{DirAC}_{[4x4]}(b)))}\right)\right) \quad (5.4\text{-}93)$$

where $C_{norm}{}^{in}_{[4x4]} = C^{in}_{[4x4]}/\max(\varepsilon, C^{in}_{[1,1]})$ and $C^{in}_{[4x4]}$is the input covariance matrix as computed in Equation (5.4-21). The prediction downmix matrix, that is computed in Equation (5.4-92), is then scaled as follows.

$$PRdmx_{[1,i]}(b) = PRdmx_{[1,i]}(b)\, scale(b) \quad (5.4\text{-}94)$$

where $1 \leq i \leq 4$.

5.4.4.3 Band Interleaving of downmix matrix at 13.2 and 16.4 kbps

When band interleaving mode or 40 ms metadata update rate mode is triggered at 13.2 and 16.4 kbps, the downmix matrix is also generated in band-interleaved fashion. The downmix matrix corresponding to the bands that are coded in the metadata bitstream in the current frame is generated using the quantized prediction coefficients as described in Equation (5.4-89). The downmix matrix corresponding to the bands that are not coded in the metadata bitstream in the current frame is generated using the previous frame's downmix matrix as follows:

$$PRdmx_{[N_{spar\_ana} x N_{spar\_ana}]}(b) = PRdmx^{prev}_{[N_{spar\_ana} x N_{spar\_ana}]}(b)\,, \quad (5.4\text{-}95)$$

where b is the band index corresponding to SPAR parameters not coded in the bitstream for the current frame and $PRdmx^{prev}_{[N_{spar\_ana} x N_{spar\_ana}]}$ is the previous frame's downmix matrix.

5.4.5 Downmix generation

5.4.5.1 Overview

Input signals which need to be modified in the parametric downmix process are run through the SPAR filterbank which employs the MDFT for fast convolution (overlap-save) with the SPAR filters. In case of active downmix processing with N_{dmx} equal to 1, all downmix input signals are transformed into the MDFT domain. Otherwise only the W signal is MDFT transformed to compute the prediction signals corresponding to the side channels, that is Y, X and Z channels. Note that the pass-through components in the downmix matrix are implemented as pure delays and the residual computation happens in the time domain for computational efficiency. Filterbank processing is done in the MDFT domain by multiplication of the audio signals with the (parameter) weighted SPAR filters frequency responses. The filterbank synthesis is realized by accumulation of all weighted SPAR filters frequency responses before the multiplication. To ensure smooth transition between frames, the parametric downmix processing is done twice, once with the previous frame parameters and second time with the current frame parameters and then the two signals are crossfaded in the time domain.

5.4.5.2 Time domain to MDFT conversion

Every signal which requires filterbank processing is arranged into a buffer of $2L_{frame}$ samples (40ms) where L_{frame} is IVAS frame length in samples. The first L_{frame}samples correspond to the previous frame input data and the second L_{frame} samples correspond to the current frame input data. This buffer is MDFT transformed resulting in L_{frame} complex-valued coefficients. Filter bank processing is done by multiplying with the accumulated 12 SPAR Filter responses, each response weighted by the corresponding parameter. Note that the accumulation across the 12 SPAR filters already constitutes the SPAR filter bank synthesis step. In case of active downmix processing, the downmixing is entirely done in the MDFT domain. Otherwise, residual signals are computed in the time domain.

5.4.5.3 Filterbank mixing

The filterbank downmixing, when number of downmix channels is equal to 1, is performed as follows., using the downmix matrix calculated according to clause 5.4.4.

$$s_{dmx}(k) = (\sum_{b=1}^{b=12} dmx_{[N_{dmx}\ x\ N_{spar_{ana}}]}(b)\, H_b(k))\ S^{sba_{ana},40ms}_{[N_{spar_{ana}}\ x\ 1]}(k) \quad (5.4\text{-}96)$$

Where k is the frequency bin index with $1 <= k <= L_{frame}$, b is the band index, $s_{dmx}(k)$ is the downmix signal in MDFT domain with N_{dmx} channels, $S^{40ms,sba\_ana}_{[N_{spar_{ana}}\ x\ 1]}(k)$ is the 40ms buffered SPAR analysis signal in the MDFT domain, $H_b(k)$ is the filterbank response as computed in clause 5.2.5.3, $dmx_{[N_{dmx}\ x\ N_{spar_{ana}}]}(b)$ is the downmix matrix calculated according to clause 5.4.4.

When number of downmix channels is greater than 1, filterbank mixing generates side channel predictions as follows.

$$s_{i,predictions}(k) = (\sum_{b=1}^{b=12} dmx_{i+1\ ,1}(b)\, H_b(k))\ W(k) \quad (5.4\text{-}97)$$

Where $1 \leq i \leq N_{dmx} - 1$, $W(k)$ is the 40ms buffered W channel input in MDFT domain.

5.4.5.4 MDFT to Time domain conversion

The MDFT domain processed signals, that is $s_{dmx}(k)$ or $s_{i,predictions}(k)$, are converted to the time domain by performing inverse MDFT and extracting the second half (960 samples with 48 kHz sampling rate) of the output buffer. Note that the mixing process and conversion from MDFT to time domain is done twice. Once with the previous frame parameters and a second time with the current frame parameters.

5.4.5.5 Windowing and crossfading

The two processed time domain signals are then cross-faded to get a smooth transition. The cross-fade length corresponds to 4ms (192 samples with 48 kHz sampling rate) and is computed from a squared sine wave section.

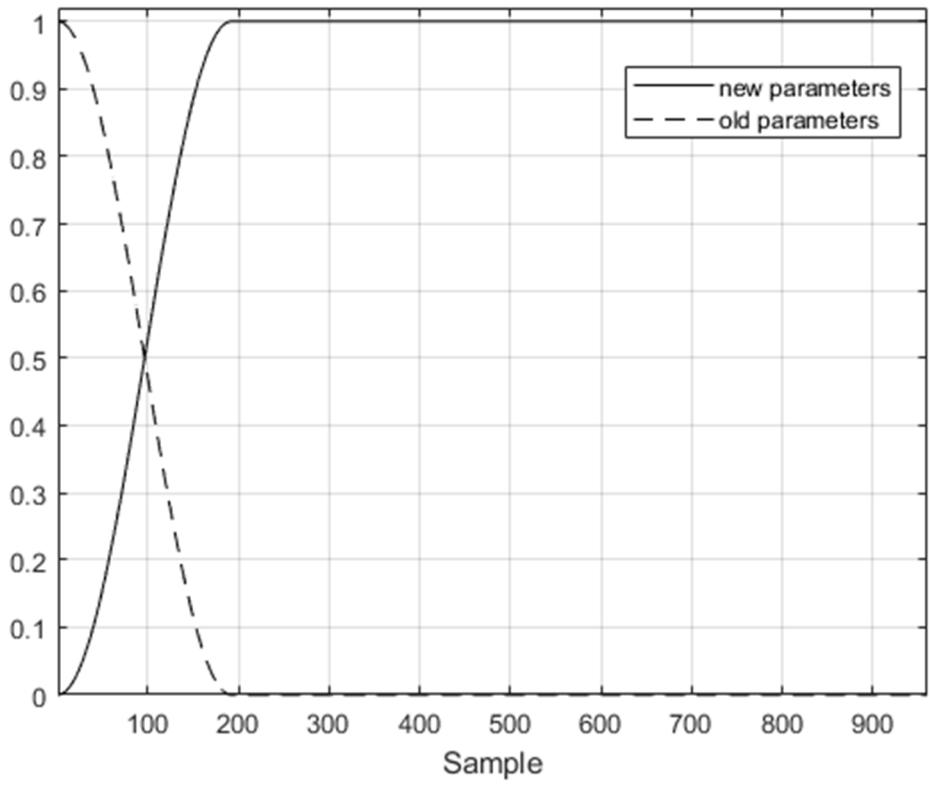


Figure 5.4-11:Cross-fading windows

5.4.5.6 Time domain mixing

When the number of downmix channels is greater than 1, the filterbank mixing and crossfading output corresponds to the side channels predictions. The final downmix is computed in the time domain by delaying side channels of the ambisonics input by the filterbank delay of 1ms (48 samples with 48 kHz sampling rate), computing the residuals by taking the difference between the delayed signals and the corresponding prediction signals.

When the number of downmix channels is two or three, the primary downmix channel, $W'(n)$, is computed based on the value of $Dyn_{activeW}$ and Res_{ind}, as computed in clause 5.4.3.7.3, for the previous frame and the current frame. If the value of $Dyn_{activeW}$ changes from 0 to 1 between the previous frame and current frame, then the computation of $W'(n)$ is as follows.

$$W'(n) = \begin{cases} c(n)S_{Res_{ind}}^{del,sba_{ana}}(n) + \big(1 - c(n)\big)W^{del}(n), & 1 \le n \le L_{crossfade} \\ S_{Res_{ind}}^{del,sba_{ana}}(n), & L_{crossfade} < n \le L_{frame} \end{cases} \quad (5.4\text{-}98)$$

Where, n is the time domain sample index, $c(n)$ is the crossfade as described in clause 5.4.5.5, $L_{crossfade}$ is the corssfade length (4ms), $W^{del}(n)$ is the W channel input delayed by the delay of SPAR MDFT filterbank, that is 1ms, $S_{Res_{ind}}^{del,sba_{ana}}(n)$ is the SPAR analysis channel corresponding to channel index Res_{ind} delayed by the delay of SPAR MDFT filterbank.

If the value of $Dyn_{activeW}$ changes from 1 to 0 between the previous frame and current frame, then the computation of $W'(n)$ is as follows.

$$W'(n) = \begin{cases} c(n)W^{del}(n) + \big(1 - c(n)\big)\, S_{Res_{ind}}^{del,sba_{ana}}(n), & 1 \le n \le L_{crossfade} \\ W^{del}(n), & L_{crossfade} < n \le L_{frame} \end{cases} \quad (5.4\text{-}99)$$

If the value of $Dyn_{activeW}$ is unchanged between the previous frame and current frame then the computation of $W'(n)$ is as follows.

$$W'(n) = \begin{cases} S_{Res_{ind}}^{del,sba_{ana}}(n), & Dyn_{activeW} = 1 \\ W^{del}(n), & Dyn_{activeW} = 0 \end{cases} \quad (5.4\text{-}100)$$

5.4.6 Principal Component Analysis (PCA)

5.4.6.1 Overview

PCA operates on the generated downmixed signals, $s_i^{DMX}(n), 1 \le i \le N_{dmx}$; PCA is restricted to the case of FOA input signal and the bit rate of 256 kbit/s where $N_{dmx} = 4$. The use of PCA is defined by a configuration parameter (`Opt_PCA_ON`) which may be either 1 (PCA on) or 0 (PCA off).

If PCA is configured to be disabled (i.e., when `Opt_PCA_ON=0`), 1 bit with value 0 corresponding to the "PCA by-pass indicator" is written in the SBA metadata and and the PCA processing described in the following subsections is skipped; in this case the output signals of the PCA processing, $s_i^{DMX}(n), 1 \le i \le N_{dmx}$, are identical to input signals, i.e. the four channels are left unchanged. The value 0 of the PCA by-pass indicator means that PCA operates in inactive mode (denoted by `PCA_MODE_INACTIVE`).

5.4.6.2 Handling of bit rate switching and other configurations than FOA

If the bitrate in the current frame is different from 256 kbit/s, or if the bitrate is 256 kbit/s but the number of input channels is higher than 4, the PCA processing is restricted to the following steps:

- The current bypass decision is set to `PCA_MODE_INACTIVE`,
- The PCA encoder state is reset, where $\boldsymbol{V}^{prev}$ is set to an identity matrix of dimension 4, the unit quaternions are set to $\widehat{\boldsymbol{q}}_i^{prev} = (1,0,0,0)$, the eigenvalues are set to $\boldsymbol{\Lambda}^{prev} = diag(1/4, \dots, 1/4)$, and the smoothed covariance $\boldsymbol{R}_{sm}$ is reset to zero.
- If the bit rate in the previous frame is different from 256 kbit/s, the output frame is identical to the input frame; otherwise, the PCA matrixing described in clause 5.4.6.9 with quaternions $\widehat{\boldsymbol{q}}_i$ set to $(1,0,0,0)$ for interpolation.
- The PCA encoder state is updated (see clause 5.4.6.10)

5.4.6.3 Covariance matrix estimation

In the current frame, the smoothed covariance matrix $\boldsymbol{R}_{sm}$ is determined based on the 4-dimensional input $s_i^{DMX}(n), 1 \le i \le 4$. First, the sample covariance $\boldsymbol{R}^{[h]}$ is estimated by half frame indexed by $h = 0,1$:

$$\boldsymbol{R}^{[h]}(i,j) = \sum_{n=0}^{L/2-1} s_i^{DMX}(n)\, s_j^{DMX}(n + L/2) \tag{5.4-101}$$

The matrix $\boldsymbol{R}_{sm}$ is updated by half-frame, for $h = 0,1$:

$$\boldsymbol{R}_{sm} \leftarrow \alpha_{sm}\boldsymbol{R}^{[h]} + (1 - \alpha_{sm})\boldsymbol{R}_{sm} \tag{5.4-102}$$

where $\alpha_{sm} = 0.0234375$. The diagonal of this matrix is further conditioned as follows:

$$\boldsymbol{R}_{sm}(i,i) \leftarrow \max(\boldsymbol{R}_{sm}(i,i), 0.00003) \tag{5.4-103}$$

5.4.6.4 Eigenvalue decomposition

The covariance matrix $\boldsymbol{R}_{sm}$ is factorized by eigenvalue decomposition as

$$\boldsymbol{R}_{sm} = \boldsymbol{V}\,\boldsymbol{\Lambda}\,\boldsymbol{V}^T \tag{5.4-104}$$

where $\boldsymbol{V}$ is the eigenvector matrix (with eigenvectors as columns) and $\boldsymbol{\Lambda} = diag(\lambda_0, \ldots, \lambda_3)$ is the diagonal matrix whose coefficients are the eigenvalue. The eigenvalue decomposition is implemented using the QR method (Householder method) with a maximum of 10 iterations.

5.4.6.5 Realignment of eigenvectors

From frame to frame, eigenvectors in $\boldsymbol{V}$ might change significantly. These modifications would create discontinuities in the signal after PCA operation. To improve signal continuity between successive frames, a signed permutation is applied to the eigenvector matrix $\boldsymbol{V}$ in the current frame based on the eigenvector matrix $\boldsymbol{V}^{prev}$ in the previous frame. To simplify notations, the successive transformations applied to $\boldsymbol{V}$ are denoted by keeping the same notation $\boldsymbol{V}$ for the transformed matrix:

1. A permutation is found by matching eigenvectors in the current and previous frames according to the axes (not directions) of each basis vector. This problem can be seen as an assignment problem where the goal is to find the closest eigenvector in the current frame for each eigenvector in the previous frame. The optimal solution is found using the Hungarian algorithm. After this step, eigenvectors are permuted. This allows to maximize similarity between the two successive bases, where the similarity is defined based on $tr(\boldsymbol{V}^t\boldsymbol{V}^{prev})$, where $tr(.)$ being the trace of a matrix, with some weighting based on the eigenvalues $\boldsymbol{\Lambda}$ and $\boldsymbol{\Lambda}^{prev}$ in the current and previous frame (respectively). The matrix of eigenvectors $\boldsymbol{V}$ is therefore transformed by applying the optimal permutation.
2. The direction of each eigenvector is determined based on the permuted eigenvector matrix in the current frame $\boldsymbol{V}$ and the rotation matrix $\boldsymbol{V}^{prev}$ in the previous frame $\boldsymbol{V}^t\boldsymbol{V}^{prev}$. A negative diagonal value in $\boldsymbol{V}^t\boldsymbol{V}^{prev}$ indicates a direction inversion between two frames. The sign of the respective columns of $\boldsymbol{V}$ is inverted to compensate for this change of direction.
3. In general, the matrix $\boldsymbol{V}$ is orthogonal and it may be either a rotation matrix ($\det(\boldsymbol{V}) = +1$) or a reflection matrix ($\det(\boldsymbol{V}) = -1$). The eigenvector matrix $\boldsymbol{V}$ is forced to define a rotation matrix by swapping its last two columns.

5.4.6.6 Conversion to double quaternion

The 4x4 rotation matrix $\boldsymbol{V}$ is factorized using the so-called Cayley transform:

$$\boldsymbol{V} = \boldsymbol{Q}_1\boldsymbol{Q}_2^{\star} \tag{5.4-105}$$

where

$$\boldsymbol{Q}_1 = \begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ -b_1 & a_1 & -d_1 & c_1 \\ -c_1 & d_1 & a_1 & -b_1 \\ -d_1 & -c_1 & b_1 & a_1 \end{pmatrix} \tag{5.4-106}$$

and

$$\boldsymbol{Q}_2^{\star} = \begin{pmatrix} a_2 & -b_2 & -c_2 & -d_2 \\ b_2 & a_2 & -d_2 & c_2 \\ c_2 & d_2 & a_2 & -b_2 \\ d_2 & -c_2 & b_2 & a_2 \end{pmatrix} \tag{5.4-107}$$

are the left and right matrix representations of two unit quaternions $\boldsymbol{q}_1 = (a_1, b_1, c_1, d_1)$ and $\boldsymbol{q}_2 = (a_2, b_2, c_2, d_2)$ (respectively). This factorization is unique up to sign. A detailed implementation of this factorization is described in Appendix of [22].

These two quaternions form a double quaternion $(\boldsymbol{q}_1, \boldsymbol{q}_2)$ that is used for quantization and interpolation purposes.

5.4.6.7 Adaptive PCA mode decision

The decision to activate PCA in the current frame is made based on the following criteria: amount of decorrelation (`dist_alt`), interframe stability of unit quaternions (`min_dot, min_dot2`).

Given $\boldsymbol{\Lambda} = diag(\lambda_0, \ldots, \lambda_3)$, eigenvalues sorted in descending order. The amount of decorrelation is derived from the sorted and normalized eigenvalues λ_i:

$$\texttt{dist\_alt} = \sum_{i=0}^{3} \left(\log \left[\frac{1}{\lambda_0 + \cdots + \lambda_3} \boldsymbol{R}_{sm}(i,i) + \delta \right] - \log \left[\frac{1}{\lambda_0 + \cdots + \lambda_3} \lambda_i + + \delta \right] \right) \tag{5.4-108}$$

with $\delta = 10^{-8}$.
The interframe stability of unit quaternions is defined as:

$$\texttt{min\_dot} = \boldsymbol{q}_1 \cdot \boldsymbol{q}_1^{prev} \tag{5.4-109}$$

$$\texttt{min\_dot2} = \boldsymbol{q}_2 \cdot \boldsymbol{q}_2^{prev} \tag{5.4-110}$$

where $\boldsymbol{q}_i^{prev}$ is the unit quaternion $\boldsymbol{q}_i$ in the previous frame and $\cdot$ is the vector dot product.

By default, the PCA by-pass indicator is set to active mode (denoted by `PCA_MODE_ACTIVE`). This decision is changed to inactive mode (denoted by `PCA_MODE_INACTIVE`) when one of the following conditions is met:

```
dist_alt < IVAS_PCA_THRES_DIST_ALT
min_dot < IVAS_PCA_THRES_MIN_DOT
min_dot2 < IVAS_PCA_THRES_MIN_DOT2
```

If the PCA by-pass indicator is set to PCA_MODE_INACTIVE, the matrix $\boldsymbol{V}$ is reset to an identity matrix of dimension 4, the unit quaternions are set to $\widehat{\boldsymbol{q}}_i^{prev} = (1,0,0,0)$ and one bit with value 0 corresponding to the "PCA by-pass indicator" is written in the SBA metadata; in this case the output signals of the PCA processing, $s_i^{DMX}(n), 1 \le i \le N_{dmx}$, are identical to input signals, i.e. the four channels are left unchanged, if the PCA by-indicator was also set to PCA_MODE_INACTIVE, otherwise double quaternion and interpolation and PCA matrixing described in clause 5.4.6.9 are applied and PCA processing in the current frame is terminated.

Otherwise, when PCA by-pass indicator is set to PCA_MODE_ACTIVE, one bit with value 0 corresponding to the "PCA by-pass indicator" is written in the SBA metadata to indicate active mode.

5.4.6.8 Quantization of double quaternion

5.4.6.8.1 Pre-processing

If the double quaternion $(\boldsymbol{q}_1, \boldsymbol{q}_2)$, where $\boldsymbol{q}_1 = (a_1, b_1, c_1, d_1)$ and $\boldsymbol{q}_2 = (a_2, b_2, c_2, d_2)$, represent the rotation matrix $\widehat{\boldsymbol{V}}_t$, the opposite $(-\boldsymbol{q}_1, -\boldsymbol{q}_2)$ also represents the same matrix. This redundancy is exploited by pre-processing $(\boldsymbol{q}_1, \boldsymbol{q}_2)$ to force a first component with positive sign in $\boldsymbol{q}_1$: if $a_1 < 0$, $\boldsymbol{q}_1 \leftarrow -\boldsymbol{q}_1$ and $\boldsymbol{q}_2 \leftarrow -\boldsymbol{q}_2$.

Then, the double quaternion is quantized by 4D spherical coding.

5.4.6.8.2 Spherical grid definition

A unit quaternion $\boldsymbol{q} = (a, b, c, d)$, is converted to spherical coordinates $(1, \phi_1, \phi_2, \phi_3)$ where 1 is the unit radius and other coordinates are angles verifying:

$$\begin{cases} a = \cos\ \phi_1 \\ b = \sin\ \phi_1 \cos\ \phi_2 \\ c = \sin\ \phi_1 \sin\ \phi_2 \cos\ \phi_3 \\ d = \sin\ \phi_1 \sin\ \phi_2 \sin\ \phi_3 \end{cases} \quad (5.4\text{-}111)$$

The spherical grid is defined by a sequential scalar quantization of angles ϕ_1, ϕ_2, ϕ_3. The angle ϕ_1 is coded by uniform scalar quantization on N_1 levels on the interval $[0, \pi]$ (including 0 and π as reconstruction levels), and the resulting coded value is:

$$\hat{\phi}_1(i) = \frac{i}{N_1 - 1}\pi, i = 0, \dots, N_1 - 1 \quad (5.4\text{-}112)$$

where

$$N_1 = 2\left[\frac{90}{\alpha}\right] + 1 \quad (5.4\text{-}113)$$

and $[.]$ is the rounding to the nearest integer, $\alpha = 2$ degrees. Note that this gives a value $N_1 = 91$ that can be pre-computed.

The angle ϕ_2 is coded by uniform scalar quantization on $N_2(i)$ levels on the interval $[0, \pi]$ (including 0 and π as reconstruction levels), and the resulting coded value is:

$$\hat{\phi}_2(i, j) = \frac{j}{N_2(i) - 1}\pi, \quad j = 0, \dots, N_2(i) - 1 \quad (5.4\text{-}114)$$

where

$$N_2(i) = \max\left(1, \left[\frac{180}{\alpha}\sin\hat{\phi}_1(i)\right]\right) \quad (5.4\text{-}115)$$

Note that $N_2(0) = 1$ and $N_2(N_1 - 1) = 1$. The angle ϕ_3 is coded by uniform scalar quantization on $N_3(i, j)$ levels on the interval $[0, 2\pi]$ by exploiting angle wrapping:

$$\hat{\phi}_3(i, j, k) = \delta(i, j) + \frac{k}{N_3(i,j)}2\pi, \quad k = 0, \dots, N_3(i, j) - 1 \quad (5.4\text{-}116)$$

where

$$N_3(i, j) = \max\left(1, \left[\frac{360}{\alpha}\sin\hat{\phi}_1(i)\sin\hat{\phi}_2(i, j)\right]\right) \quad (5.4\text{-}117)$$

Note that $N_3(0,0) = 1$ and $N_3(N_1 - 1, 0) = 1$.

The spherical grid is therefore defined as a codebook with entries defined in terms of spherical coordinates as:

$$\left\{\left(1, \hat{\phi}_1(i), \hat{\phi}_2(i, j), \hat{\phi}_3(i, j, k)\right) | i = 0, \dots, N_1 - 1; j = 0, \dots, N_2(i) - 1, k = 0, \dots, N_3(i, j) - 1\right\} \quad (5.4\text{-}118)$$

and the total number of points in this grid is given by:

$$N_{tot} = \sum_{i=0}^{N_1 - 1}\sum_{j=0}^{N_2(i) - 1} N_3(i, j) \quad (5.4\text{-}119)$$

which gives $N_{tot} = 463456$.

For indexing purposes, cumulative cardinalities are defined as:

$$offset_1(0) = 0 \quad (5.4\text{-}120)$$

$$offset_1(i) = \left[\left(\frac{i - 0.5}{N_1 - 1}\pi - \frac{1}{2}\sin\left(\frac{i - 0.5}{N_1 - 1}2\pi\right)\right)\frac{N_{tot}}{\pi}\right], i = 1, \dots, N_1 - 1 \quad (5.4\text{-}121)$$

and

$$offset_2(i,0) = 0 \tag{5.4-122}$$

$$offset_2(i,j) = \left[\left(1 - \cos\left(\frac{j-0.5}{N_2(i)-1}\pi\right)\right)\frac{N_{subtot}(i)}{2}\right], \tag{5.4-123}$$

where

$$N_{subtot}(i) = \left[\left(\frac{i+0.5}{N_1-1}\pi - \frac{1}{2}\sin\left(\frac{i+0.5}{N_1-1}2\pi\right)\right)\frac{N_{tot}}{\pi}\right] - \left[\left(\frac{i-0.5}{N_1-1}\pi - \frac{1}{2}\sin\left(\frac{i-0.5}{N_1-1}2\pi\right)\right)\frac{N_{tot}}{\pi}\right] \tag{5.4-124}$$

for $i = 1, \ldots, N_1 - 2$.

The cumulative cardinality, $offset_1(i)$, is pre-stored and listed below for $i = 0, \ldots, N_1$:

$offset_1(i)$ = {

0, 1, 9, 61, 163, 359, 685, 1125, 1747,
2519, 3521, 4713, 6183, 7883, 9809, 12093, 14633, 17575,
20807, 24343, 28181, 32487, 37121, 42097, 47405, 53057, 59061,
65421, 72137, 79205, 86625, 94415, 102345, 110629, 119263, 128017,
137097, 146515, 156043, 165637, 175551, 185515, 195535, 205837, 216183,
226545, 236911, 247273, 257619, 267921, 277941, 287905, 297819, 307413,
316941, 326359, 335439, 344193, 352827, 361111, 369041, 376831, 384251,
391319, 398035, 404395, 410399, 416051, 421359, 426335, 430969, 435275,
439113, 442649, 445881, 448823, 451363, 453647, 455573, 457273, 458743,
459935, 460937, 461709, 462331, 462771, 463097, 463293, 463395, 463447,
463455, 463456

} (5.4-125)

Similarly, the cumulative cardinality, $offset_2(i,j)$, is also pre-stored. For the sake of conciseness, it is not listed here.

5.4.6.8.3 Encoding of a unit quaternion

The encoding of a unit quaternion, $\boldsymbol{q} = (a, b, c, d)$, is done sequentially by spherical coordinates $(1, \phi_1, \phi_2, \phi_3)$.

If $\left||a| - 1\right| < \varepsilon$, with $\varepsilon = 10^{-7}$, ϕ_1 is directly given by 0 ($i = 0$) if a>0 or π ($i = N_1 - 1$) if a<0; the reconstructed unit quaternion is obtained by converting $(1, \phi_1, 0, 0)$ to Cartesian coordinated, and the index, $index$, is set to $offset_1(i)$. Encoding of $\boldsymbol{q}$ is then terminated.

Otherwise, if $\left||a| - 1\right| \geq \varepsilon$, the spherical coordinate ϕ_1 is computed:

$$\phi_1 = \arccos a \tag{5.4-126}$$

and two candidate indices are obtained for optimal search:

$$i_1(0) = \arg \min_{i=0,\ldots,N_1-1} \left(\phi_1 - \hat{\phi}_1(i)\right)^2 \tag{5.4-127}$$

$$i_1(1) = \arg \min_{\substack{i=0,\ldots,N_1-1 \\ i \neq i_1(0)}} (\phi_1 - \hat{\phi}_1(i))^2 \tag{5.4-128}$$

If $\left|\left|b/\sqrt{b^2 + c^2 + d^2}\right| - 1\right| < \varepsilon$, , with $\varepsilon = 10^{-7}$, ϕ_2 is directly given by 0 ($i_2(0) = i_2(1) = 0$) if b>0 or π ($i_2(0) = N_1(i_1(0)) - 1, i_2(1) = N_2(i_1(1)) - 1$) if b <0. The optimal candidate is selected after converting $\left(\hat{\phi}_1(i_1(0)), \hat{\phi}_2(i_1(0), i_2(0)), 0\right)$ and $\left(\hat{\phi}_1(i_1(1)), \hat{\phi}_2(i_1(1), i_2(1)), 0\right)$ into Cartesian coordinates and minimizing the distance to $\boldsymbol{q} = (a, b, c, d)$. The corresponding quantization indices are: $(i_1(m), i_2(m), 0)$, m = 0 or 1. The index, $index$, is set to $offset_1(i_1(m)) + offset_2(i_1(m), i_2(m))$. Encoding of $\boldsymbol{q}$ is then terminated.

Otherwise, if $\left|\left|b/\sqrt{b^2+c^2+d^2}\right|-1\right| \geq \varepsilon$, the spherical coordinate ϕ_2 is computed:

$$\phi_2 = \arccos b/\sqrt{b^2+c^2+d^2} \tag{5.4-129}$$

Four candidates $\hat{\phi}_2(i_1(m) \gg 1, i_2(m))$, $m = 0, \ldots, 3$ are defined with

$$i_2(0) = \arg \min_{i=0,\ldots,N_2(i_1(0))-1} (\phi_2 - \hat{\phi}_2(i_1(0), i))^2 \tag{5.4-130}$$

$$i_2(1) = \arg \min_{\substack{i=0,\ldots,N_2(i_1(0))-1 \\ j \neq i_2(0)}} (\phi_2 - \hat{\phi}_2(i_1(0), i))^2 \tag{5.4-131}$$

and

$$i_2(2) = \arg \min_{i=0,\ldots,N_2(i_1(1))-1} (\phi_2 - \hat{\phi}_2(i_1(1), i))^2 \tag{5.4-132}$$

$$i_2(3) = \arg \min_{\substack{i=0,\ldots,N_2(i_1(1))-1 \\ j \neq i_2(2)}} (\phi_2 - \hat{\phi}_2(i_1(1), i))^2 \tag{5.4-133}$$

The spherical coordinate ϕ_3 is computed:

$$\phi_3 = \begin{cases} \arccos y/\sqrt{y^2+z^2} & z \geq 0 \\ 2\pi - \arccos y/\sqrt{y^2+z^2} & z < 0 \end{cases} \tag{5.4-134}$$

This angle is coded four times by uniform scalar quantization with an adaptive number of levels $N_3(i,j)$ where $i = i_1(m \gg 1)$ and $j = i_2(m)$, $m = 0,..,3$, to obtain 4 values $\hat{\phi}_3(i_1(m \gg 1), i_2(m), i_3(m))$, $m = 0,..,3$.

The optimal candidate is selected after converting $\left(\hat{\phi}_1(i_1(m)), \hat{\phi}_2(i_1(m), i_2(m)), \hat{\phi}_3(i_1(m \gg 1), i_2(m), i_3(m))\right)$ into Cartesian coordinates and minimizing the distance to $\boldsymbol{q} = (a, b, c, d)$. The corresponding quantization indices are: $(i_1(m), i_2(m), i_3(m))$, $m = 0$ or 1. The index, $index$, is set to $index = offset_1(i) + offset_2(i,j) + k$, where $i = i_1(m \gg 1)$, $j = i_2(m)$, and $k = i_3(m)$.

5.4.6.8.4 Encoding of double quaternion

The bit rate allocated to $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ is 18 and 19 bits, respectively. The encoding described in clause 5.4.6.8.3 is repeated for $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$. One bit is saved in the coding of $\boldsymbol{q}_1$ because this unit quaternion is coded on a hemi-hypersphere, due to the forced positive first component a_1. The two indices of 18 and 19 bits are written to the bitstream corresponding to the SBA metadata. The reconstructed double quaternion is denoted $(\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2)$.

5.4.6.9 Interpolation of double quaternions, conversion to rotation matrices and PCA matrixing

The rotation matrices are interpolated by subframes to smooth variations of PCA rotation matrices and avoid signal discontinuities. The current frame of length L is divided into $K = 40$ blocks (slots).. For each block of index $0 \leq k < K$ in the current frame, the left $(\hat{\boldsymbol{q}}_1^{prev}, \hat{\boldsymbol{q}}_1)$ and right quantized quaternions $(\hat{\boldsymbol{q}}_2^{prev}, \hat{\boldsymbol{q}}_2)$ are pre-processed to ensure that the shortest path is used for interpolation; the left $(\hat{\boldsymbol{q}}_1^{prev}, \hat{\boldsymbol{q}}_1)$ and right quantized quaternions $(\hat{\boldsymbol{q}}_2^{prev}, \hat{\boldsymbol{q}}_2)$ are then interpolated by the NLERP (normalized linear interpolation) algorithm:

$$\hat{\boldsymbol{q}}_i^{[k]} = \frac{\alpha_k \hat{\boldsymbol{q}}_i^{prev} + (1-\alpha_k)\hat{\boldsymbol{q}}_i}{\left\|\alpha_k \hat{\boldsymbol{q}}_i^{prev} + (1-\alpha_k)\hat{\boldsymbol{q}}_i\right\|} \tag{5.4-135}$$

where $\|.\|$ is the norm of the quaternion and

$$\alpha_k = 1 - \frac{1}{2}\left(1 - \cos\frac{\pi k}{K-1}\right) \tag{5.4-136}$$

The interpolated double quaternion $\left(\widehat{\boldsymbol{q}}_{\boldsymbol{1}}^{[k]}, \widehat{\boldsymbol{q}}_{\boldsymbol{2}}^{[k]}\right)$ is converted to a rotation matrix $\boldsymbol{V}^{[k]}$ using the factorization (see Eq. 5.4-105) into the corresponding left and right quaternion matrix representations (see Eqs. 5.4-106 and 5.4-107). In each block of index k, the FOA signal, $s_i^{DMX}(n), 1 \le i \le N_{dmx}, n = \frac{kL}{K}, \ldots, \frac{(k+1)L}{K} - 1$, is transformed into principal components by applying the interpolated rotation matrix in PCA matrixing:

$$\begin{bmatrix} s_1^{DMX}(n) \\ s_2^{DMX}(n) \\ s_3^{DMX}(n) \\ s_4^{DMX}(n) \end{bmatrix} \leftarrow \boldsymbol{V}^{[k]} \cdot \begin{bmatrix} s_1^{DMX}(n) \\ s_2^{DMX}(n) \\ s_3^{DMX}(n) \\ s_4^{DMX}(n) \end{bmatrix} \tag{5.4-137}$$

5.4.6.10 Encoder state update

The quantized quaternions $\widehat{\boldsymbol{q}}_{\boldsymbol{1}}^{prev}$ and $\widehat{\boldsymbol{q}}_{\boldsymbol{2}}^{prev}$ in the previous frame are replaced by $\widehat{\boldsymbol{q}}_1$ and $\widehat{\boldsymbol{q}}_2$. The matrix $\boldsymbol{V}^{prev}$ is replace by $\boldsymbol{V}$. The PCA by-pass decision in the current frame is also saved for the next frame.

5.4.7 Automatic Gain Control (AGC)

AGC aims to reduce audio artifacts caused by an overload condition of an audio signal to be encoded by a target audio codec. An overload condition exists when the audio signals exceed the range expected by the target audio encoder. As SPAR downmixing of Ambisonics input channels may lead to some or more of the downmix channels exceeding the range of the input channels, the IVAS SBA coding mode provides an AGC mechanism for handling of potential overload cases. AGC specified here is a perceptually motivated overload handling without the need for an additional processing delay, i.e., zero delay AGC. AGC for IVAS SBA operates on the generated downmixed signals, $s_i^{DMX\_enc}(n), 1 \le i \le N_{dmx}$, described in clause 5.4.5, which are to be encoded by the core encoder described in clause 5.4.8, acting as a target audio encoder. Note that AGC is only applied to a single downmixed channel case, $N_{dmx} = 1$. The audio signals expected by the core encoder range within the interval $[-32768, 32767]$.

If an overload condition exists, AGC determines a gain parameter $e(j)$, of the current frame j to be encoded, and a gain transition function to be applied to the current frame samples/signals. The gain transition function is constructed comprising two portions, i.e., a transitory portion and a steady-state portion. The transitory portion is constructed by scaling a prototype smoothing function $p(\cdot)$ by a gain scaling factor obtained from the current and previous frame gain parameters, $e(j) - e(j-1)$. This portion corresponds to a transition from the gain parameter of the previous frame to the current frame. When gain control is applied, it either takes the form of gain increases or decreases over a portion of samples of the current frame. The steady-state portion is merely a constant gain value specified by the last index (right-most) of the transitory portion.

To achieve zero additional delay processing, AGC defines the length of transitory portion based on the overall core codec delay. Given the overall core codec delay $D = 12$ ms, corresponding to the total core encoder and decoder delays, 8.75 ms and 3.25 ms, respectively, and the frame length of $L = 20ms$, the transitory portion length L_p is set to $L - D = 8$ ms. This is equal to setting $L = 960$, $D = 576$, and $L_p = 384$ samples at 48 kHz sampling frequency.

The gain transition function $t(l, j)$ at a sample index l in frame j is thus defined as

$$t(l,j) = \begin{cases} p(l,j)^{(e(j)-e(j-1))}, & l = 0 \ldots L_p - 1 \\ p\left(L_p - 1, j\right)^{(e(j)-e(j-1))}, & l = L_p \ldots L - 1. \end{cases}$$

The prototype smoothing function $p(\cdot)$ is defined below depending on a gain transition step size $DBSTEP$ in dB:

$$p(DBSTEP, l, j) = 1 + a(DBSTEP) \cdot \left(\cos\left(\pi \cdot \frac{l}{L_p - 1}\right) - 1\right),$$

where

$$a(DBSTEP) = 0.5 \cdot \left(1 - 10^{\frac{DBSTEP}{20}}\right).$$

$DBSTEP$ has been determined based on subjective and objective quality criteria; thus, it is directly related to the SCE/CPE and MCT core codecs of the IVAS codec. It is set to a predefined value of $DBSTEP = -6\,dB$. Figure 5.4-12 depicts the gain transition function at $48kHz$ sampling frequency having positive gain scaling factors.

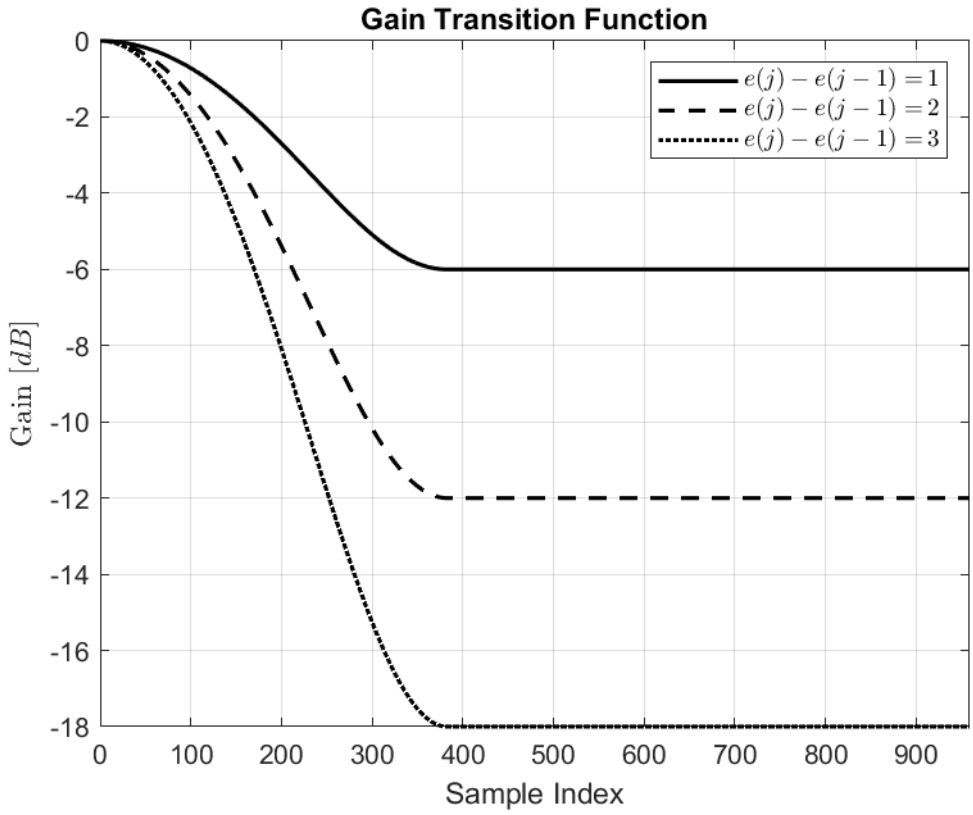


Figure 5.4-12: Gain transition function having the scaling factor $e(j) - e(j-1)$ set to 1, 2 and 3.

The following equation defines the relevant relation for constructing the current gain transition function based on the gain transition function applied to the preceding frame $t(DBSTEP, L-1, j-1)$, and where the gain transition step size plays the determining role in adjusting the transition.

$$t(DBSTEP, l, j) = t(l, j) \cdot t(DBSTEP, L-1, j-1), \quad l = 0 \cdots L-1\,,$$

where $t(DBSTEP, L-1, j-1)$ is initialized to 1.

Consequently, applying the gain transition function results in the gain adjusted frame $s_i^{AGC\_enc}(n)$, of the downmixed signal $s_i^{DMX\_enc}(n)$. Gain attenuation or amplification results in an attenuated or amplified frame of input signals, respectively. Finally, setting the sample index l to n, and introducing the downmix channel index $i = 1$, indicating a single downmixed channel, gives the following formulation:

$$s_{i=1}^{AGC\_enc}(n) = t(DBSTEP, n, j) \cdot s_{i=1}^{DMX_{enc}}(n),$$

where the gain adjusted frame $s_{i=1}^{AGC\_enc}(n)$ is then encoded by the core encoder.

AGC encodes gain control information in the dedicated AGC metadata bitstream consisting of several bit fields. Firstly, AGC always encodes one bit field indicating the presence or absence of further AGC metadata in the bitstream. If gain control is applied in the current frame, this bit is set to 1 and further AGC metadata is written to the bitstream as described below. If the gain control is not applied, it is set to 0, corresponding to applying a unity gain to the downmix channel, and no further bits are written.

Secondly, based on the information that a gain control is applied in the current frame, AGC encodes the gain parameter $e(j)$ using a field of $\beta_E = ceil(\log_2(E_{MIN} + E_{MAX} + 1))$ bits, where $E_{MAX} = 0$ and $E_{MIN} = \max\big(3, ceil(\log_2(1.5 \cdot N_{HOA}))\big)$, representing the maximum and minimum gain parameter attenuation range, respectively. The term N_{HOA} ideally signifies the amount of HOA components, from which the downmixed signal is calculated. It is set to an FOA case having $N_{HOA} = 4$ components, resulting in β_E set to 2 bits. This gain parameter is encoded using a binary encoding scheme. For a given $\beta_E = 2$ bits, $e(j)$ takes a value from the set $\{0,1,2,3\}$ corresponding to the binary code $\{(00), (01), (10), (11)\}$.

5.4.8 Core-coder encoding

5.4.8.1 General

The downmix signal $s_i^{DMX}(n)$ is coded with either SCE (clause 5.2.3.2), CPE (clause 5.2.3.3) or MCT (clause 5.2.3.4) core-coding tool as per Table 5.4-1.

5.4.8.2 MCT bitrate distribution in SBA high bitrate mode

In SBA format, for bitrates greater than or equal to 256 kbps, the bitrate distribution in MCT is further modified as follows. The ratio $\hat{r_i}$ as computed in Equation (5.2-259) is further modified based on a bitrate distribution process and $bits_{remaining}$ as computed in clause 5.2.3.4.6. The bitrate distribution process is described as follows. First the $core_{brs}$ are read from SPAR bitrate distribution table as described in clause 5.4.3.7.11. Then the residual bits are computed by subtracting total core-coder target bits from available bits as given below.

$$bits_{residual} = bits_{remaining} - \frac{\left(\sum_{n=1}^{n=N_{dmx}} core_{brs[n][1]}\right)}{50} \quad (5.4\text{-}101)$$

If the $bits_{residual} \geq 0$, then bits are added to target core coding bits as per the priority order which is decided based on perceptual importance of downmix channels as indicated in Equation (5.4-56).

$$bits_i = \frac{\left(core_{brs[i][1]}\right)}{50} + min(bits_{residual},\ bits_{range})$$

where $1 \leq i \leq N_{dmx}$ and $bits_{range}$ is computed as given below.

$$bits_{range} = \frac{\left(\sum_{n=1}^{n=N_{dmx}} core_{brs[n][3]}\right)}{50} - \frac{\left(\sum_{n=1}^{n=N_{dmx}} core_{brs[n][1]}\right)}{50}$$

After computing $bits_i$ corresponding to downmix channel index i, the $bits_{residual}$ is updated as

$$bits_{residual} = bits_{residual} - min(bits_{residual},\ bits_{range})$$

If the $bits_{residual} < 0$, then bits are taken from target core coding bits as per the priority order.

$$bits_i = \frac{\left(core_{brs[i][1]}\right)}{50} - min(bits_{overflow},\ bits_{range})$$

where $1 \leq i \leq N_{dmx}, bits_{overflow} = -bits_{residual}$ and $bits_{range}$ is computed as given below

$$bits_{range} = \frac{\left(\sum_{n=1}^{n=N_{dmx}} core_{brs[n][1]}\right)}{50} - \frac{\left(\sum_{n=1}^{n=N_{dmx}} core_{brs[n][2]}\right)}{50}$$

After computing $bits_i$ corresponding to downmix channel index i, the $bits_{overflow}$ is updated as

$$bits_{overflow} = bits_{overflow} - min(bits_{overflow},\ bits_{range})$$

After iterating over all downmix channels, if $bits_{overflow}$ is still greater than 0 then $bits_i$ is further updated as

$$bits_i = bits_i - \left(floor\left(\frac{bits_{overflow}}{N_{dmx}}\right)\right)$$

$$bits_{N_{dmx}-1} = bits_{N_{dmx}-1} - mod(bits_{overflow}, N_{dmx})$$

Then target ratio is computed from $bits_i$ as given below.

$$tr_i = \frac{bits_i}{\sum_{n=1}^{n=N_{dmx}} bits_n}$$

$$tr_i = \max(1,\ \min(r_{RANGE} - 1, \lfloor r_{RANGE}\ tr_i + 0.5 \rfloor))$$

where r_{RANGE} is as given in Equation (5.2-260).

The target ratio is then further modified as follows.

$$tr_4 = tr_4 - ratio_{diff}$$

$$tr_2 = tr_2 + ratio_{diff}$$

where $ratio_{diff} = tr_4 - min\left(\frac{tr_4}{\sum_{n=1}^{n=N_{dmx}} tr_n}, \frac{\hat{r}_4}{\sum_{n=1}^{n=N_{dmx}} \hat{r}_n}\right) * \left(\sum_{n=1}^{n=N_{dmx}} tr_n\right)$.

Then the ratio $\hat{r}_i$ is computed as

$$\hat{r}_i = tr_i$$

$$\hat{r}_i = \max(1, \min(r_{RANGE} - 1, \lfloor r_{RANGE}\, \hat{r}_i + 0.5 \rfloor))$$

where r_{RANGE} is as given in Equation (5.2-260).

5.4.9 DTX operation

5.4.9.1 Overview

In the SBA format, the DTX operation mode is supported from 13.2 kbps to 80 kbps while the DTX operation is not supported at bitrates equal or higher than 96 kbps.

In SBA format, the metadata frame is flagged as inactive frame if W_{vad} is set to 0. The VAD of core-coder and front-end VAD, described in clause 5.4.3.1, are tuned such that if W_{vad} is set to 1, then core-coder VAD will be set to 1 as well. However, if W_{vad} is set to 0, then core-coder VAD depends on downmix signal and is independent of W_{vad}. As a result, in some frames, SPAR MD may be coded in inactive mode while the frame (MD + core-coder) may be coded in active mode. W_{vad} is coded with 1 bit, in SBA metadata bitstream, to indicate if MD is coded in active mode or inactive mode.

A frame is coded as SID or NO_DATA frame if W_{vad} is set to 0 and core-coder is operating in SID or NO_DATA frame mode.

In inactive mode, SBA MD is coded in four frequency bands with SPAR MD in two low frequency bands and DirAC MD in two high frequency bands.

5.4.9.2 DirAC parameter estimation

The audio scene analysis done in DirAC is performed for both active and inactive frames and produce two different sets of spatial parameters. Active and inactive frames are categorized using the Voice Activity Detection (VAD) performed on the transport channels. However, the spatial DirAC parameters for the SID frames of the inactive phases are fewer and quantized coarser than the spatial parameters of the active phase.

Once, the DirAC parameters are derived as for the active frame for a given parameter band partition, the grouping, the quantization, and the coding of the DirAC parameters (diffuseness and directions of arrival (DoAs)) are done as described in clause 5.2.4.

5.4.9.3 SPAR parameter estimation

5.4.9.3.1 General

In SPAR inactive coding, the current MD frame is set as active frame or inactive frame based on output of VAD detector W_{vad}, described in clause 5.4.3.1. The metadata frame is flagged as inactive frame if W_{vad} is set to 0, then a downmix signal and upmix metadata is computed from the FOA input signal to SPAR, where the downmix signal has either one channel or two channels and the upmix metadata is such that the downmix signal can be upmixed into FOA signal at the output of the upmixer at decoder. In inactive mode, the computation of upmix metadata is different from active coding in terms of band grouping, covariance smoothing, and quantization and coding. The upmix metadata includes PR and D coefficients that are computed, quantized and coded in IVAS bitstream as described below.

5.4.9.3.2 Banded covariance smoothing

The SPAR metadata is computed based on an inter-channel covariance matrix which is obtained by performing temporal smoothing on the banded covariance described in Equation (5.4-20). The covariance matrix is tracked separately for inactive frames and the temporal smoothing is performed as per Equation (5.4-21), where the temporal smoothing in inactive frames is higher than that of active frames. The forgetting factor for inactive frames is given in Equation (5.4-102). If a transient is detected in a frame, then the covariance of that frame is removed from the temporal smoothing calculation by setting $\alpha(b)$ to zero, as per Equation (5.4-102).

$$\alpha(b) = \begin{cases} 1, & frameIdx_{prev} = -1 \\ 0, & T_0 = 1 \; OR \; T_1 = 1 \\ \alpha_{sm}(b), & otherwise \end{cases} \quad (5.4\text{-}102)$$

where,

$$N_b = \sum_{k=k_0^b}^{k=k_1^b} H_{abs,5ms}^{mdft}(b,k) \quad (5.4\text{-}103)$$

$$N_{min} = \begin{cases} \frac{40}{0.5b}, & if \; ivas \; bit \; rate < 24.4kbps \\ \frac{40}{0.75b}, & if \; ivas \; bit \; rate \geq 24.4kbps \end{cases} \quad (5.4\text{-}104)$$

$$\alpha_{sm}(b) = \min\left(0.4, \frac{N_b}{N_{min}}\right), \quad (5.4\text{-}105)$$

The smoothed covariance is then band-remixed as per clause 5.4.3.7.4.

5.4.9.3.3 Quantization and coding in VAD inactive frames

SPAR coefficients are computed as follows. First PR coefficients are computed as per clauses 5.4.3.7.6 and 5.4.3.7.7. CP coefficients are set to 0 as mentioned in clause 5.4.3.7.8. Then D coefficients are computed as per clause 5.4.3.7.9. When number of downmix channels N_{dmx} is set to1 then D coefficients are further boosted as follows.

$$D_i(b) = \begin{cases} 0.4, & if \; D_i(\mathrm{b}) > 0.1, D_i(\mathrm{b}) < 0.4, N_{dmx} = 1 \\ D_i(\mathrm{b}), & otherwise \end{cases}$$

Then D coefficients are quantized using uniform quantization given in Equations (5.4-68) and (5.4-69) using the parameters given below.

$$D_{min} = 0, D_{max} = 1.6, qlvl_D = 7$$

Then PR coefficients are quantized as per PR_{min} and PR_{max} given in Equation (5.4-65), and the $qlvl_{PR}$ is set based on number of downmix channels and channel priority as follows.

$$qlvl_{PR}^i = \begin{cases} 7, & if \; N_{dmx} = 2, i = 3 \\ 9, & otherwise \end{cases}$$

where i is the channel index and $i = 3$ corresponds to Z channel as per ACN ordering.

PR coefficients are then quantized using uniform quantization given in Equations (5.4-68) “(5.4-68)” and (5.4-69) “(5.4-69)”.

SPAR coefficients are coded in pairs where each PR coefficients that is quantized with 9 quantization points is paired with D coefficient that is quantized with 7 quantization points. Together they form 63 quantization points that are coded with base2 coding using 6 bits. When $N_{dmx} = 2$, PR coefficients corresponding to Z channel are coded separately with 3 bits.

5.4.9.4 SPAR and DirAC parameter merge

In inactive MD frames, SBA MD is coded for 4 frequency bands out of which SPAR MD is computed for band index b with $1 \leq b \leq 2$ and DirAC MD is computed for band index b with $3 \leq b \leq 4$.

In the high-frequency parameter bands, the SPAR MD are derived from the DirAC parameters according to clauses 5.4.3.8.2 and 5.4.3.8.5

5.4.9.5 CNG parameters encoding

Core encoding of the CNG parameters is based on the FD-CNG encoding procedure from EVS with the modifications described in 5.2.2.3.5. If only one transport channel is used, the to-be-encoded noise parameter vector $N^{dB}_{SBA,FD-CNG}$ is calculated for the downmix signal. If two transport channels are used, the noise parameter vectoris calculated as the mean of the noise parameter vectors of the two transport channels:

$$N^{dB}_{SBA,FD-CNG} = 0.5\left(N^{dB}_{1,FD-CNG}(i) + N^{dB}_{2,FD-CNg}(i)\right).$$

MSVQ encoding is then done as described in 5.2.2.3.5.

5.4.10 SBA bitrate switching

Bitrate switching is supported from any IVAS bitrate to any other IVAS bitrate. As the configurations of all IVAS components depend strongly on the bitrate, these configurations must be checked and modified accordingly when a bitrate switch is requested. At every frame, it is checked whether the current bitrate R equals that of the previous frame R_{old}. If this is not the case, a re-configuration is triggered.

Then the new SBA analysis order (cf. clause 5.4.3.4) is computed first. If it differs from the previous one, the number of SBA analysis channels, N_{sba_ana}, changes and HP20 filter buffers are allocated or deallocated to match the new number of channels.

Then the SPAR encoder is re-configured. Specifically, the number of SPAR analysis channels and the number of transport channels in the first set, N_{dmx}, (cf. clauses 5.4.4 and 5.4.5) is re-computed according to Table 5.4-1 and Table 5.4-2. Then the number of CPEs or SCEs are set based on the new IVAS bitrate. The table row index for SPAR bitrate distribution table is computed as per the new IVAS bitrate and SBA analysis order. For the case of a single transport channel and one SCE, the core nominal bitrate is set depending on the total IVAS bitrate. When either the number of SPAR analysis channels or the number of transport channels in the first set changes on the bitrate switch, more changes are made to the SPAR encoder configuration: The number of MDFT filterbank buffers in both the parameter-estimation and signal path and the dimensions of the downmix matrix are adjusted (cf. clauses 5.4.4 and 5.4.5). The size of the buffers for the covariance matrix and the SPAR coefficients is also adapted to the new number of channels. The covariance smoothing factor is changed depending on the bitrate.

If the number of transport channels changes to 1 with bitrate switching, then AGC is initialized otherwise AGC is deallocated if the number of transport channels changes from 1 to a value greater than 1. If PCA is enabled with the new but not the old bitrate, the necessary initialization is performed.

The DirAC encoder is reconfigured too. First, the decision between the high-order and low-order operation mode (cf. clause 5.4.3.4) is made based on the new bitrate. The encoding unit can switch between a low-order and a high-order operation mode. The low-order mode is active for bitrates less than 384 kbps. The high-order mode is switched on for the bitrates 384 and 512 kbps.

In the low-bitrate mode, the encoding path for the second sector is deactivated and no information for the second sector is written to the bitstream as side information. The first sector then has an omni-directional spatial filter. In the high-bitrate mode, the encoder paths for both sectors are switched on. The parameters for both sectors are contained in the metadata. This is realized by setting a variable to indicate whether the encoder runs in the high- or low-order operation mode.

In addition, the number of parameter bands and DoAs is re-computed. The buffer sizes for the DirAC metadata are adjusted to these new values. If necessary, the buffers are freed and re-allocated with the correct dimensions. It is determined whether the parameter estimation will be performed on the encoder side for all bands (high-order operation mode) or only for the high bands (low-order operation mode). The grouping of the parameter bands and the time resolution of the parameters is set up according to the bitrate. If the number of parameters bands or DoAs changes, the buffers for the quantized energy ratio indices are initialized with zeros.

Finally, the core-coder is reconfigured to reflect the changes in the number of transport channels in the first set and the bitrate.

5.5 Metadata-assisted spatial audio (MASA) operation

5.5.1 MASA format overview

The metadata-assisted spatial audio (MASA) operation encodes the IVAS encoder inputs that use the MASA format. This is a parametric spatial audio format that can be used with any multi-microphone array with suitable capture analysis. The MASA format is optimised for immersive audio capture by smartphones and other form factors that may utilize irregular microphone arrays.

The MASA format is based on audio channels and an associated set of metadata parameters. The audio signals can be one or two, i.e., mono or stereo. These can be denoted as mono-MASA (MASA1) and stereo-MASA (MASA2), respectively. The metadata parameters include spatial metadata parameters providing information about the captured spatial audio scene for transmission and reproduction of the spatial audio, and descriptive metadata parameters providing further description about the capture configuration and source format of the spatial audio content represented by the MASA format.

Each MASA metadata frame, corresponding to 20 ms of audio, includes the descriptive metadata (consisting of a format descriptor and a channel audio format field that further defines the number of directions described by the spatial metadata, number of audio channels, the source format configuration, and a variable description depending on the previous information) and the spatial metadata parameters that are: direction index, direct-to-total energy ratio, diffuse-to-total energy ratio, remainder-to-total energy ratio, spread coherence, and surround coherence.

The direction index (decodable with an elevation and an azimuth component) provides an efficient representation of the multitude of possible spatial directions with about 1-degree accuracy in any arbitrary direction. The direction indices define a spherical grid that covers a sphere with several smaller spheres with centres of the spheres giving the points corresponding with the directions.

Each spatial metadata parameter is provided (through capture, analysis, or creation) for each of 96 time-frequency (TF) tiles corresponding to 4 temporal (or time) subframes and 24 frequency bands. The MASA frequency band borders as CLDFB frequency bins are shown in table 5.5-1.

Table 5.5-1: MASA frequency band borders as CLDFB frequency bins

Band	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
First bin	0	1	2	3	4	5	6	7	8	9	10	11
Last bin	0	1	2	3	4	5	6	7	8	9	10	11
Band	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
First bin	12	13	14	15	16	17	18	19	20	25	30	40
Last bin	12	13	14	15	16	17	18	19	24	29	39	59

The direct-to-total energy ratio and spread coherence parameters are associated with the direction (parameter). The direction index, direct-to-total energy ratio, and spread coherence parameters are therefore given for each direction described per TF tile (as given by the number of directions descriptive metadata parameter). For each TF tile, the sum of the different energy ratio parameters is 1.0.

The metadata for MASA format inputs shall be provided as defined in detail in Annex A of [12]. The uncompressed MASA metadata size according to definitions in [12] is between 272.8 and 426.4 kbps (268.8 and 422.4 kbps for spatial metadata only), depending on the number of directions in spatial metadata. As IVAS supports encoding of both mono-MASA and stereo-MASA at IVAS bitrates between 13.2 and 512 kbps, as described in Table 4.2-1, significant compression of the MASA metadata is performed during the encoding process. The compression is based on several overall strategies, including a bitrate dependent configuration for MASA metadata, simplification of the spatial metadata based on combining of spatial metadata parameters in time or frequency, decreasing the number of directions from 2 to 1, and selecting the method of compression based on the configuration parameter indicating the source format.

Figure 5.5-1 presents the MASA encoder. The details of the encoding algorithm are provided in the following clauses.

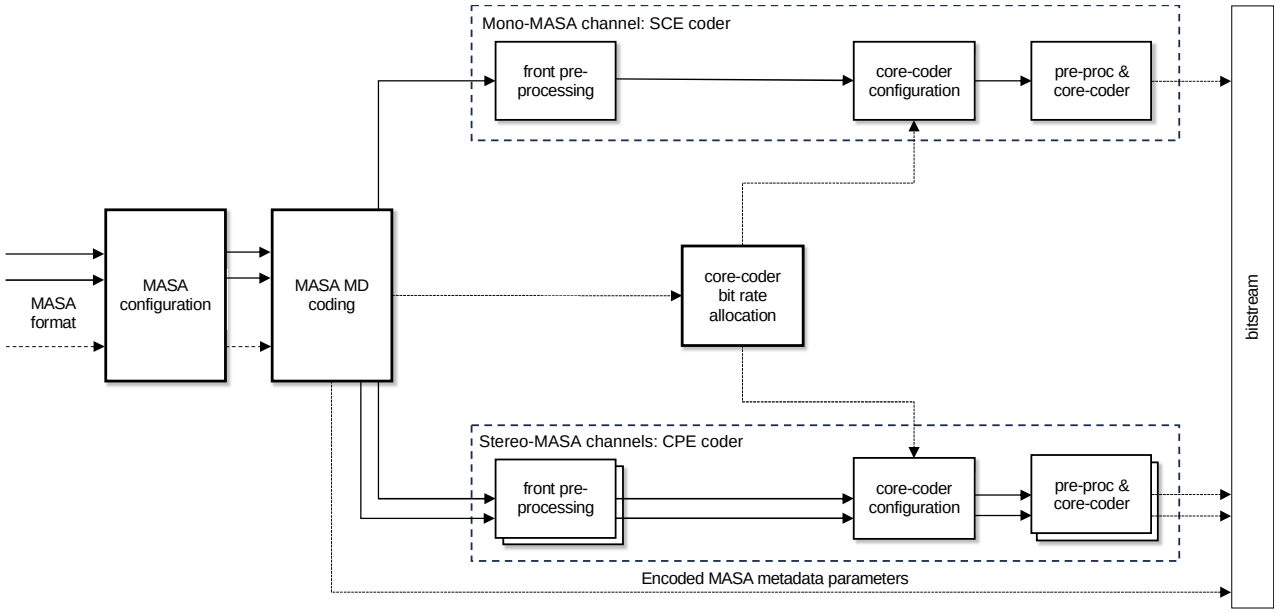


Figure 5.5-1: Block diagram of MASA encoder

5.5.2 MASA format metadata input

5.5.2.1 Obtaining the MASA format metadata

The initial part of MASA format encoding is obtaining the MASA format input at the encoder. The 1 or 2 transport audio signals and the associated MASA format metadata (comprising of spatial metadata and descriptive metadata) are received into the encoder (e.g., by reading from a file, retrieved from a stream, etc.).

The transport audio signals are handled with the common audio signal encoding path (see clauses 5.1 and 5.2.2). The MASA metadata are provided with a structure interface where the structure comprises: a structure for MASA descriptive metadata (descriptive_meta), 1 to 2 structures for respective directive parameter sets of spatial metadata (directional_meta), and a structure for common spatial metadata (common_meta). These structures are populated with the MASA metadata parameters read from the MASA format metadata bitstream of each input MASA format frame. As part of MASA metadata bitstream decoding, a verification is done against the format descriptor field in descriptive metadata that it corresponds to bytes signifying "IVASMASA". No other type of MASA format is supported by the IVAS encoder. This descriptor also signifies the beginning of each metadata frame.

As part of the population of the MASA metadata structures, the spatial metadata parameter values are transformed to a more suitable format for the encoding process while preserving their accuracy. In practice, direction parameters are deindexed from spherical indexes or direction indexes to azimuth and elevation values in degrees (see clause 5.5.2.2 for details) and ratio values are scaled to the range from 0 to 1. The populated MASA metadata structures are stored, and thus provided for all of the MASA metadata encoder stages described in later subclauses of 5.5.

Finally, energy parameter values are also computed for each time-frequency tile from the associated transport audio signals to be stored together with the metadata parameter values of the corresponding each time-frequency tile. This is detailed in clause 5.5.2.3. The computed energy is stored into a structure and provided for the further MASA metadata encoder stages together with the metadata.

5.5.2.2 Direction index deindexing

The index corresponding to the direction information is deindexed to the azimuth and elevation values. These values correspond to a point in the spherical grid described in Annex A of [12]. The deindexing procedure is described in this section. The deindexing consists first in estimating the spherical grid circle index value *id*, by solving a polynomial function of the spherical grid circle index value *id* that corresponds to the input spherical direction index value that needs to be deindexed. The estimated spherical grid circle index value ($estim$) (which is the estimate of the spherical grid circle index value *id)* is used to border by a lower value and a higher value the potential elevation index value. Once the elevation index is determined, using the offset indexes for the points on the circle corresponding to that elevation index, the azimuth index is derived as well. The angle values are finally derived.

The polynomial approximation which defines the cumulative index value as function of the grid circle index value, *id* is given by:

$$sphIndex = p_0\, id^2 + p_1 id + p_2 \qquad (5.5\text{-}1)$$

However, the above polynomial is not actually solved for the grid circle index value, *id*. Instead, the estimated spherical grid circle index, $estim$, for the input spherical index direction $sphIndex$ is calculated as the minimum of 121 and the solution of the above equation by:

$$estim = \min(121, ba + div * \sqrt{del + a4 * sphIndex})$$

The parameter sets used to obtain the estimated solution of the above second order polynomial are given in table 5.5-2 below and they are defined for three different ranges of the input direction index.

Table 5.5-2: Polynomial parameters

$sphIndex$	$sphIndex \geq 64964$	$47870 \leq sphIndex < 64964$	$sphIndex \leq 47870$
ba	1.228408647140870e+02	1.244854404591542e+02	2.137991118026424e+02
del	1.424072242426373e+06	1.300883976959332e+06	7.998262115303199e+05
div	-0.092050209205032	-0.100938185496887	-0.237662341081474
$a4$	-21.727272727270197	-19.814106922515204	-8.415300425381099

The procedures of adjusting the lower and higher limits for the elevation index and its final determination are detailed below and they use the array $n_\theta[l], l = 0{:}\,121$ holding the number of azimuth values for each elevation value corresponding circle on the sphere. The values of $n_\theta[l], l = 0{:}\,121$ are calculated as follows:

$n_\theta[0] = 430$
$n_\theta[121] = 1$

```
cum_n_prev = 0;
for i=1:120
{
    theta = 0.012894427382667 * (I + 0.5);
    if (i == 1)
    {
        cum_n = 2 * ceilf(32768.00566947353 * sin(theta)-- 0.0064471690266724975);
    }
    else
    {
       cum_n = 2 * roundf(32768.00566947353 * sin(theta)-- 0.0064471690266724975);
    }
    n_θ[i] = (int)((cum_n—cum_n_prev)/2);
    cum_n_prev = cum_n;
}
id_th = round(estim)-- 1;
```

The estimate of the elevation index is obtained by rounding of $estim$ to the nearest integer.

Similarly to the derivation of the n_θ value, the cumulated index offset corresponding to `id_th`, is obtained using the functions from the previous pseudo-code and assigned to the lower limit *base_low*. The upper limit is obtained as

```
base_up = base_low + 2 * n_θ [id_th];
```

After these initializations, the deindexing procedure proceeds as below:

```
sign_theta = 1;

n_crt = n[id_th];
if ( sphIndex < base_low )
{
    id_th--;
    n_crt = n[id_th];
    if ( id_th == 0 )
    {
        base_low = 0;
        base_up = n_crt;
    }
    else
    {
        base_up = base_low;
        base_low -= 2 * n[id_th];
```

```
    }
}
else if ( sphIndex >= base_up )
{
    id_th++;
    n_crt = n[id_th];
    base_low = base_up;
    base_up += 2 * n_crt;
}
```

The resulting deindexed elevation value is $\theta = id_{th} * 0.738796268264740 * sign\_theta$ and the azimuth is $\phi = id\_phi * 360 / n\_crt - 180$ if *id_th* is even and $\phi = (id_{phi} + 0.5) * 360 / n\_crt - 180$ if *id_th* is odd.

5.5.2.3 TF tile based energy calculation

The MASA format encoding algorithms take time-frequency domain transport audio signal energy as an input. This the computation of the energy values is practical to do once as a pre-processing step. This computation is done in time-frequency domain (namely CLDFB) and provides an energy parameter value for each of the 96 time-frequency tiles corresponding to the time-frequency resolution of the associated MASA spatial metadata parameters. The computation of the energy is carried out with the equation

$$E(b,m) = \sum_{k=k_{low}(b)}^{k_{high}(b)} \sum_{i} \left| S_i^{CLDFB}(k,m) \right|^2$$

where $S_i^{CLDFB}(k,m)$ is the CLDFB-domain transformed (see clause 6.2.5.1 for description of the CLDFB transform) received transport audio signal $s(m)$, b is the MASA frequency band, m is the time subframe, i is the transport audio signal channel index, k is the frequency bin index of the CLDFB-domain, and $k_{low}(b)$ and $k_{high}(b)$ set the range of CLDFB-domain frequency bins that belong to each MASA frequency band.

The computed energy parameter values are stored into an internal encoder structure to be available for retrieval by further encoding processes.

5.5.3 Coding of MASA format input data

5.5.3.1 Overview

The coding of MASA input format data starts with the encoding of the metadata. The number of bits of the metadata is variable, even within one operation point, but it is limited to a highest allowed value, set for each operational mode. In order to respect the maximum number of bits allowed, the metadata parameters are reduced before encoding. Some of the parameter reductions are dependent on the operation mode, others are explicitly signalled. For each frame the encoded metadata, including the signalling bits, is written at the end of the bitstream. The remaining bits are used for the transport channels encoded as a CPE (clause 5.2.3.3) for stereo-MASA or as a SCE (clause 5.2.3.2) for mono-MASA, as described in clause 5.5.4.

5.5.3.2 MASA metadata pre-encoding configuration, processing, and signalling

5.5.3.2.1 Overview

The input MASA metadata parameters are specified for 96 TF tiles (24 frequency bands and 4 subframes) for 1 or 2 directions. Before quantization and encoding they are grouped or reduced to a lower number of same type equivalent parameters (that is, parameter sets for each TF-tile). This way they correspond to the number of parameters to be encoded which is specified for each codec total bitrate in table 5.5-3 and common codec configuration procedure in clause 5.5.3.2.4. The MASA metadata pre-encoding configuration, processing and signalling is given by the following pseudo-code:

```
Set_encoder_configuration();
Energy_ratio_compensation();
If ivas_bitrate >= 384000
{
    Do_not_merge_ratios_over_subframes = 1;
}
Combine_freqbands_and_subframes();
If num_input_directions == 2 ∧ num_coding_subbands_dir1 != num_coding_subbands_dir2
```

```
{
    Combine_directions();
}
Write_number_of_transport_channels;
Write_reserved_bits();
Write_number_of_directions();
Write_subframe_mode();
If (max_metadata_bits < 100 ∧ joined_subframes== 0)
{
    Reduce_metadata_further(); /* merge through time (nblocks:=1) or through frequency (nbands:=1)*/
    write_metadata_reduction_mode();
}
```

The detailed functionality of the pseudo-code is described in the following sections. MASA encoder configuration (*Set_encoder_configuration()*) is presented in clause 5.5.3.2.2. The energy ratio compensation processing (*Energy_ratio_compensation()*) is presented in clause 5.5.3.2.4. The metadata parameter set reduction methods (*Combine_freqbands_and_subframes()*, *Combine_directions()*, *Reduce_metadata_further()*) are presented in clauses 5.5.3.2.6, 5.5.3.2.7, and 5.5.3.2.8, respectively. The bit signalling writing is described in clause 5.5.3.2.10. This MASA pre-encoding configuration, processing, and signalling is followed by the MASA metadata quantization and encoding in clause 5.5.3.3.

As part of the pseudocode, if bitrate is at least 384 kbps, then it is ensured that energy ratios are not merged over subframes in the reduction functions by disabling the *mergeRatiosOverSubframes* flag.

5.5.3.2.2 Bitrate dependent parameter settings

The MASA encoder is configured for each input frame based on the input MASA spatial metadata (as obtained in clause 5.5.2.1 as part of obtaining the MASA spatial audio content also containing the associated transport audio signals) and current bitrate. This process setups the core coding elements and configures the metadata codec to quantize and encode the input MASA spatial metadata together with the transport audio signals to optimally preserve the quality of the represented spatial audio content. As the same MASA format encoder is used also by McMASA multichannel format encoding module and OMASA combined format encoding module, the MASA encoder configuration is reused by these modules with some adjustments. These specific adjustments are described in clauses 5.7.3.6.1 and 5.9.2.

The MASA encoder configuration first sets the core coding elements to suitable values based on the total bitrate, number of transport channels, the IVAS format, and the details of the format (MC mode, ISM mode, ISM total bitrate). The output is the number of core coding elements, element mode, and nominal bits for the frame of metadata.

Next, the metadata composition is detected, the metadata is possibly aligned if necessary, and validity of metadata is checked. This process is described in clause 5.5.3.2.3. This is followed by the common MASA metadata codec configuration which sets various configuration parameters (that is, encoding configuration parameters) based on the given inputs including detected metadata composition and received configuration parameters (total bitrate, number of transport channels, and IVAS format). This block is shared by the encoder and the decoder and described in detail in clause 5.5.3.2.4.

When metadata codec configuration has been set, then if there are two valid directional parameters sets present and codec is configured to encode at least one band with two directions, an importance weight table is constructed that defines for each coding band an importance weight between 0 and 1 with former being "not important" and latter being "very important". The constructed table is different for each number of coding bands B_{coding} and generally has smaller weight for higher coding bands. Unused bands are set to zero importance and in the special case of all bands having two directional data, importance is set to one for all bands. This importance table is stored in a MASA encoder structure.

In case the metadata composition detection in clause 5.5.3.2.3.4 has set the “joinedSubframes” to true, then the number of temporal subframes is set in the metadata encoder structure (of the encoder detailed in clause 5.5.3.3) to be 1 instead of the default 4. This causes the metadata encoder (in clause 5.5.3.3) to encode just the first time subframe of each frequency band, as all 4 temporal subframes (in this mode) will each have identical spatial audio parameter sets, i.e. that of the first time subframe.

Next, memory is allocated or reallocated for the metadata encoder structure based on the requirements. This is followed by setting common spatial metadata encoding (detailed in clause 5.2.4) configuration parameters based on the MASA encoder configuration parameters. An encoding configuration parameter is also set based on if a received configuration parameter (i.e., IVAS format which is obtained separately to the spatial audio content as an argument) is either MC-format (which indicates McMASA analysis) or MASA input format. This received configuration parameter indicates the source format of the spatial audio content (i.e., the format of the spatial audio content that was used to obtain the

spatial metadata) represented by the MASA spatial metadata to be encoded. Thus, it allows optimising the encoding of the spatial metadata by selecting a method of compression (in clause 5.2.4.3.2.2) optimized for the MASA spatial metadata independently of the metadata content. The encoding configuration parameter “mc_ls_setup” is set and indicates the MC LS setup used and contains "MC_LS_SETUP_INVALID" in the case of MASA input format (or MASA part of OMASA) or indication of the channel layout of the original MC format (5.1, 7.1, 5.1+2, 5.1+4, 7.1+4) which is obtained together with the IVAS format when IVAS format is MC-format. The encoding configuration parameter “mc_ls_setup” is provided for (and obtained within) the metadata encoder in clause 5.2.4.3.2.2.

The next step is to set coherence data configuration parameter “all_coherence_zero” in the metadata encoder structure based on the metadata composition detection in clause 5.5.3.2.3.4. If there are no significant coherence parameter values present in the metadata, then this is signalled with the configuration parameter. This is followed by setting a maximum bit requirement value in the metadata encoder structure based on the IVAS format.

Next, the number of coding bands B_{coding} and the band mapping is adjusted for the transport audio signal bandwidth and signal content as detailed in clause 5.5.3.2.11. If B_{coding} would be less that B_{2dir}, then $B_{2dir} = B_{coding}$.

Finally, the stereo core coder is set to force a mono transmission in the case that the bitrate for MASA format is less than 24.4 kbps and there are two transport channels present in the MASA format input.

5.5.3.2.3 Metadata composition detection, alignment, and check for validity

5.5.3.2.3.1 Overview

As part of the metadata codec configuration, MASA metadata is first inspected for structural and parameter validity properties. The input metadata always contains 24 frequency bands and 4 temporal subframes and either 1 or 2 sets of directional metadata parameters. However, the underlying structure of metadata can vary and detecting it allows improved metadata coding output.

In this step, first, the directional metadata is aligned in the case of two parameter sets of directional metadata. This is described in clause 5.5.3.2.3.2 Then, metadata framing is inspected together with the previous metadata frame to see if the metadata is considered to have same metadata across subframes (called "1sf"-mode) but in asynchrony (i.e., not aligned with the frame borders). If such asynchrony is detected, then the metadata is combined across subframes. This process is described in clause 5.5.3.2.3.3.

As the final step after metadata adjustments, the metadata composition is analyzed for: similarity of subframes, presence of significant coherence parameters, and validity of directional parameter sets. This process is described in clause 5.5.3.2.3.4.

5.5.3.2.3.2 Aligning MASA metadata directions

MASA spatial metadata can contain two ordered directions $i \in [0,1]$ associated with two audio sources within an audio scene. Each direction consists of own directional metadata parameters arranged as a grid of time-frequency tiles for identifying a direction of arrival, i.e., azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$, direct-to-total energy ratio $r_{dir}(b,m,i)$, and spread coherence $\zeta(b,m,i)$ for each time-frequency tile, where m and b are the time axis (sub-frame) and frequency axis (parameter band) grid indices. The original ordering of the spatial metadata, i.e., assignment of a specific source direction into a specific metadata direction i, is not always optimal considering the temporal continuity of the metadata within the direction i. To make the metadata better suitable for the further processing steps, the original spatial metadata tiles (b,m,i) are re-ordered between the two directions $i \in [0,1]$ such that an error measure is minimized as follows.

First, the directional metadata parameters for two directions $i \in [0,1]$ associated with two audio sources within an audio scene are obtained, specifically the parameters identifying a direction of arrival, i.e., azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$. An ordering error is determined for each time-frequency tile. More specifically, the ordering considers temporally consecutive (current m and previous $m-1$ subframe which may also belong to a different metadata parameter frame) time-frequency tiles in the same parameter band b. The ordering error is configured to identify the case that the directional metadata parameters describing the source direction in a neighboring time-frequency tile are more similar and/or less different in another direction order index than in the original order index. In other words, that the sound source directions described by the azimuth and elevation metadata parameters are more similar with the previous time-frequency tile in the same parameter band when the metadata direction field assignment $i \in [0,1]$ is modified by reordering it.

The error measure for not reordering the directions is computed as

$$D_{no\_swap}(b,m) = \arccos\big(\cos\big(\phi(b,m,0)\big)\cos\big(\phi(b,m-1,0)\big)\cos\big(\theta(b,m,0)-\theta(b,m-1,0)\big) + \sin\big(\phi(b,m,0)\big)\sin\big(\phi(b,m-1,0)\big)\big) + \arccos\big(\cos\big(\phi(b,m,1)\big)\cos\big(\phi(b,m-1,1)\big)\cos\big(\theta(b,m,1)-\theta(b,m-1,1)\big) + \sin\big(\phi(b,m,1)\big)\sin\big(\phi(b,m-1,1)\big)\big)$$

The error measure for reordering the directions is computed as

$$D_{swap}(b,m) = \arccos\big(\cos\big(\phi(b,m,0)\big)\cos\big(\phi(b,m-1,1)\big)\cos\big(\theta(b,m,0)-\theta(b,m-1,1)\big) + \sin\big(\phi(b,m,0)\big)\sin\big(\phi(b,m-1,1)\big)\big) + \arccos\big(\cos\big(\phi(b,m,1)\big)\cos\big(\phi(b,m-1,0)\big)\cos\big(\theta(b,m,1)-\theta(b,m-1,0)\big) + \sin\big(\phi(b,m,1)\big)\sin\big(\phi(b,m-1,0)\big)\big)$$

The values two partial error measures are compared and the decision to reorder the directions is made based on the comparison. If the error measure value for the direction parameters when not reordering the direction is larger than the error measure value for the direction parameter when reordering the associated order indices, i.e., $D_{no\_swap}(b,m) > D_{swap}(b,m)$, the directions are reordered to make the metadata parameters of direction of arrival more similar between temporally consecutive neighboring metadata subframes. In other words, the direction ordering indices are reordered by reversing the association of the spatial metadata parameter to them with

$$\theta'(b,m,0) = \theta(b,m,1)$$

$$\theta'(b,m,1) = \theta(b,m,0)$$

$$\phi'(b,m,0) = \phi(b,m,1)$$

$$\phi'(b,m,1) = \phi(b,m,0)$$

$$r'_{dir}(b,m,0) = r_{dir}(b,m,1)$$

$$r'_{dir}(b,m,1) = r_{dir}(b,m,0)$$

$$\zeta'(b,m,0) = \zeta(b,m,1)$$

$$\zeta'(b,m,1) = \zeta(b,m,0)$$

This re-ordered spatial metadata is used in the further processing instead of the original spatial metadata.

If the input MASA spatial metadata contains only one active direction, no re-ordering is done.

5.5.3.2.3.3 MASA metadata framing asynchrony correction

5.5.3.2.3.3.1 Detecting MASA metadata framing asynchrony

The MASA spatial metadata parameters are analyzed for determining if the metadata framing (grouping of four subframes into metadata frames) is in asynchrony compared to the system framing. The analysis obtains the frame of four subframes of spatial metadata parameters associated with the audio signal. Further, the analysis is based on a state structure containing a copy of the MASA spatial metadata parameters of the last subframe of the previous frame, the offset detected (asynchrony information) in the previous frame, and the number of similar subframes from the end of the previous subframe. The analysis returns the asynchrony information consisting of the determined MASA metadata frame mode ("1sf" or "4sf") and framing offset N_{offset} describing the number of subframes containing similar values. The spatial metadata frame processing described in clause 5.5.3.2.3.3.3 obtains the asynchrony information for each frame of four subframes, and based on the asynchrony information processes the spatial metadata parameter values of the frame of four subframes before the encoder processes the frame further. The purpose of the processing in clause 5.5.3.2.3.3.3 is to enable more efficient further processing of the spatial metadata parameter frames in the encoding.

In this context, a MASA spatial metadata subframe $I(m)$ consists of the parameters $I(m) = \{ N_{dir}, \theta(b,m,i), \phi(b,m,i), r_{dir}(b,m,i), \zeta(b,m,i), \gamma(b,m,i)\}$, where N_{dir} is the number of directions in the frame. When the relevant spatial metadata parameters in two subframes are similar, this is denoted with $I(m) \approx I(n)$. If the relevant spatial metadata parameters in two subframes are not similar, this is denoted with $I(m) \not\approx I(n)$.

The number N_{start} of spatial metadata subframes similar to the first subframe in the current frame is determined using the method described in clause 5.5.3.2.3.3.2. The value of N_{start} is the largest in the range $1 \leq N_{start} \leq 4$ such that

$I(0) \approx I(m), \forall m \in [0, N_{start} - 1]$. In other words, N_{start} is the number of consecutive subframes following the current frame start (i.e., the first subframe of the current frame) which have spatial metadata parameter set similar to the spatial metadata parameter set of the first subframe. Similarly, the number N_{stop} of spatial metadata subframes similar to the last subframe in the current frame is determined. The value is the largest in the range $1 \leq N_{stop} \leq 4$ such that $I(3) \approx I(3 - m), \forall m \in [0, N_{stop} - 1]$. In other words, N_{stop} is the number of consecutive subframes preceding the current frame end (i.e., the last subframe of the current frame) which have spatial metadata parameter set similar to the spatial metadata parameter set of the last subframe.

The metadata framing mode is initialized to "4sf", and the offset with

$$N_{offset} = \begin{cases} 0, & \text{if } N_{offset}^{[-1]} > 2 \\ N_{offset}^{[-1]}, & \text{otherwise} \end{cases}$$

The framing mode and offset are determined based on N_{start}, N_{stop}, $N_{offset}^{[-1]}$, and $N_{stop}^{[-1]}$:

If $N_{stop} == 4$ and $N_{start} == 4$, set framing mode to "1sf" and determine the offset with

$$N_{offset} = \begin{cases} N_{stop}^{[-1]}, & \text{if } 0 < N_{stop}^{[-1]} < 3 \text{ and } I(0) \approx I^{[-1]}(3) \\ N_{offset}^{[-1]}, & \text{if } N_{stop}^{[-1]} == 3 \text{ and } I(0) \approx I^{[-1]}(3) \\ 0, & \text{otherwise} \end{cases}$$

and exit the analysis.

If $N_{stop} == 3$, set framing mode to "1sf" and $N_{offset} = 3$, and exit the analysis.

If $N_{stop}^{[-1]} > 0$ and $I(0) \approx I^{[-1]}(3)$ and $N_{start} > 1$ and $N_{start} + N_{stop}^{[-1]} \geq 4$, set framing mode to "1sf" and determine offset with

$$N_{offset} = \begin{cases} min\left(2, N_{stop}^{[-1]}\right), & \text{if } N_{offset}^{[-1]} == 0 \\ N_{offset}^{[-1]}, & \text{otherwise} \end{cases}$$

and exit the analysis. The asynchrony information consists of the determined MASA metadata frame mode ("1sf" or "4sf") and framing offset N_{offset} are provided to further processing as described in clause 5.5.3.2.3.3.3.

5.5.3.2.3.3.2 Detecting mutually similar spatial metadata in MASA subframes

Two values of the MASA spatial metadata parameters in the MASA metadata subframes $A = I(m)$ and $B = I(n)$ are compared and determined if they are similar. A number of checks are performed and if none of them indicates that the subframes are not similar, the subframes are similar. On the subframe level, if the two subframes have a different number of directions, they are not similar. The other comparisons are performed on each parameter band b for each spatial metadata parameter direction i. The absolute value of the difference between the spatial parameter values are compared to a threshold specific for the parameter to determine if the parameter in the two subframes is similar or not similar. When any spatial metadata parameter in any parameter band b in any spatial metadata direction i is determined to be not similar, the whole subframes can be determined to be not similar.

Obtain the azimuth angle spatial metadata parameter values $\theta_A(b, i)$ and (b, i). Determine the difference in the azimuth angles with

$$d_{azi}\left(\theta_A(b, i), \theta_B(b, i)\right) = |\theta_A(b, i) - \theta_B(b, i)|$$

The angle difference is wrapped if $d_{azi}\left(\theta_A(b, i), \theta_B(b, i)\right) > 180$ by

$$d_{azi}\left(\theta_A(b, i), \theta_B(b, i)\right) = 360 - d_{azi}\left(\theta_A(b, i), \theta_B(b, i)\right)$$

Compare the difference in the azimuth angle spatial metadata parameter values with a threshold, and if $d_{azi}\left(\theta_A(b, i), \theta_B(b, i)\right) > 0.5$, the subframes are not similar.

Obtain the elevation angle spatial metadata parameter values $\phi_A(b, i)$ and $\phi_B(b, i)$. Determine the difference in elevation angles with

$$d_{ele}\big(\phi_A(b,i),\phi_B(b,i)\big) = |\phi_A(b,i) - \phi_B(b,i)|$$

Compare the difference in the elevation angle spatial metadata parameter values with a threshold, and if $d_{ele}\big(\phi_A(b,i),\phi_B(b,i)\big) > 0.5$, the subframes are not similar.

Obtain the direct-to-total energy ratio spatial metadata parameter values $r_{dir;A}(b,i)$ and $r_{dir;B}(b,i)$. Determine the difference in direct-to-total energy ratios with

$$d_{dir}\left(r_{dir;A}(b,i),r_{dir;B}(b,i)\right) = |r_{dir;A}(b,i) - r_{dir;B}(b,i)|$$

Compare the difference in the direct-to-total energy ratio spatial metadata parameter values with a threshold, and if $d_{dir}\left(r_{dir;A}(b,i),r_{dir;B}(b,i)\right) > 0.1$, the subframes are not similar.

Obtain the spread coherence spatial metadata parameter values $\zeta_A(b,i)$ and $\zeta_B(b,i)$. Determine the difference in spread coherence parameters with

$$d_{spr}\big(\zeta_A(b,i),\zeta_B(b,i)\big) = |\zeta_A(b,i) - \zeta_B(b,i)|$$

Compare the difference in the spread coherence spatial metadata parameter values with a threshold, and if $d_{spr}\big(\zeta_A(b,i),\zeta_B(b,i)\big) > 0.1$, the subframes are not similar.

Obtain the surround coherence spatial metadata parameter values $\gamma_A(b)$ and $\gamma_B(b)$. Determine the difference in surround coherence parameters with

$$d_{sur}\big(\gamma_A(b),\gamma_B(b)\big) = |\gamma_A(b) - \gamma_B(b)|$$

Compare the difference in the surround coherence spatial metadata parameter values with a threshold, and if $d_{sur}\big(\gamma_A(b),\gamma_B(b)\big) > 0.1$, the subframes are not similar.

5.5.3.2.3.3.3 Combination of MASA metadata over subframes

The spatial metadata framing asynchrony information is obtained as described in clause 5.5.3.2.3.3.1. If the metadata asynchrony information determined by the asynchrony analysis indicates that a non-zero offset $N_{offset} \neq 0$ has been found, indicating that subframes of consecutive subframes possibly in different system frames contain similar spatial metadata parameter values, and a metadata framing mode is "1sf", the spatial metadata parameter values are processed by combining them across the subframes. This processing enables the further processing of the spatial metadata frame of four subframes such that the spatial metadata parameter values can be more efficiently encoded.

The azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$ angles obtained from the spatial metadata are first converted to Cartesian coordinate vectors. The length of the vectors is determined based on the direct-to-total energy ratio $r_{dir}(b,m,i)$ and the energy $E(b,m)$. This is performed by

$$x(b,m,i) = \cos\big(\theta(b,m,i)\big)\cos\big(\phi(b,m,i)\big)\,l(b,m,i)$$

$$y(b,m,i) = \sin\big(\theta(b,m,i)\big)\cos\big(\phi(b,m,i)\big)\,l(b,m,i)$$

$$z(b,m,i) = \sin\big(\phi(b,m,i)\big)\,l(b,m,i)$$

where

$$l(b,m,i) = r_{dir}(b,m,i)E(b,m)$$

where b is the MASA frequency band index, m is the subframe index, and i is the direction index. Then, the vectors are summed over the subframes

$$x_{sum}(b,i) = \sum_{m=0}^{3} x(b,m,i)$$

$$y_{sum}(b,i) = \sum_{m=0}^{3} y(b,m,i)$$

$$z_{sum}(b,i) = \sum_{m=0}^{3} z(b,m,i)$$

The sum of the parameter tile energies is determined with

$$E_{sum}(b) = \sum_{m=0}^{3} E(b,m)$$

The combined direction is determined with

$$\theta'(b,i) = \arctan(y_{sum}(b,i), x_{sum}(b,i))$$

$$\phi'(b,i) = \arctan(z_{sum}(b,i), \sqrt{x_{sum}(b,i)^2 + y_{sum}(b,i)^2})$$

and the combined direct-to-total energy ratio is determined by

$$r'_{dir}(b,i) = \frac{\sqrt{x_{sum}(b,i)^2 + y_{sum}(b,i)^2 + z_{sum}(b,i)^2}}{E_{sum}(b)}$$

The combined direct-to-total energy ratio is used for determining combined diffuse-to-total energy ratio with

$$r'_{diff}(b) = 1 - r'_{dir}(b)$$

for one-direction metadata and by

$$r'_{diff}(b,m) = 1 - \sum_{i=0}^{1} r'_{dir}(b,i)$$

for two-direction metadata.

The spread coherence is combined over subframes by

$$\zeta'(b,i) = \frac{\sum_{m=0}^{3} \zeta(b,m)E(b,m)}{E_{sum}(b)}$$

The surround coherence is combined over subframes by

$$\gamma'(b) = \frac{\sum_{m=0}^{3} \gamma'(b,m)E(b,m)}{E_{sum}(b)}$$

The combined spatial parameter values are assigned to all four subframes in the current system frame

$$\theta'(b,m,i) = \theta'(b,i)\ \forall m \in [0,3]$$

$$\phi'(b,m,i) = \phi'(b,i)\ \forall m \in [0,3]$$

$$r'_{dir}(b,m,i) = r'_{dir}{}'(b,i)\ \forall m \in [0,3]$$

$$\zeta'(b,m,i) = \zeta'(b,i)\ \forall m \in [0,3]$$

$$r'_{diff}(b,m,i) = r'_{diff}(b,i)\ \forall m \in [0,3]$$

$$\gamma'(b,m,i) = \gamma'(b,i)\ \forall m \in [0,3]$$

5.5.3.2.3.4 Detecting metadata composition

The input for metadata composition is the full set of input MASA metadata for the current 20-ms frame. That is, descriptive metadata, one or two sets of directional spatial parameters, and one set of common spatial parameters. The output is a set of detection results which are used in configuration. Namely, subframe composition, presence of coherence parameter values, and validity of data in presence of two directional spatial parameter sets.

First, the validity of the directional spatial parameter sets is checked. The expected number of directions is obtained from input descriptive metadata. If there is only one expected directional spatial metadata parameter set, then the first direction is always considered to be valid. If there are two expected directional spatial metadata parameter sets, then for each direction separately, the direct-to-total ratio (also called energy ratio) is inspected for each time-frequency tile. If the direct-to-total ratio for any time-frequency tile for the given directional parameter set is 0.1 or larger, then that directional parameter set is considered to be valid. Otherwise, the inspected directional parameter set is considered to be invalid.

Next, if the second directional parameter set is invalid, then the MASA metadata is considered to contain only one directional parameter set and no further steps are done for checking directional parameter set validity.

On the other hand, if the second directional parameter set is valid but the first directional parameter set is invalid, then the metadata parameters are moved from the second directional parameter set to the first directional parameter set and the MASA metadata is considered to contain only one valid directional parameter set.

In the end, if both directional parameter sets are considered to be valid, then the output parameter (*isTwoDir*) signifying validity of two directional parameter sets is set to true. Otherwise, it is set to false. Further checks in metadata composition detection are done with the detected number of valid directional parameter sets.

After checking validity of directional parameter sets, the next step is to check for similarity of data through subframes. This is done with the algorithm described in clause 5.5.3.2.3.3.2 where, on two consecutive subframes at a time, each parameter of the two spatial parameter sets are compared to each other for similarity within a defined tolerance threshold. If all subframe pairs are similar, then all the subframes in the metadata are considered to be similar and the output variable (joinedSubframes) is set to true.

Next, the coherence parameter sets (spread coherence and surrounding coherence) are inspected separately to deduce if are significant coherence parameter values present. The presence of spread coherence is checked by inspecting each spread coherence parameter value for each time-frequency tile in each directional parameter set. If any inspected spread coherence parameter value is 0.1 or larger, then coherence parameter values are considered to be significantly present. If coherence is present, then output variable for presence of coherence (*cohPresent*) is set to true.

Finally, if the previous step for checking spread coherence significance results in coherence not being present, then surrounding coherence is also checked for significance. This algorithm is described in clause 5.5.3.2.3.5 and results in truth value if surrounding coherence is significantly present. This value is assigned to the output variable for presence of coherence (*cohPresent*).

5.5.3.2.3.5 Determining the significance of surround coherence values for encoding

In this clause, the encoding of the surround coherence values associated with the frequency bands of a subframe or a frame are dependent on a significance measure calculated for the subframe or the frame, where the significance measure relates to the significance of the surround coherence values in the subframe or the frame. In other words, encoding of the surround coherence values of the subframe or the frame is performed when the significance of the said values is deemed significant enough to warrant encoding.

When there are four subframes m (this is the case when the input format is MASA), the coherence significance measure is determined separately for each subframe m, and using these values it is determined whether to encode the surround coherence values.

When there is only one subframe m (this is the case when the input format is multi-channel, i.e., when operating in the McMASA mode), the coherence significance measure is determined for the whole frame, and using this value it is determined whether to encode the surround coherence values.

The coherence significance measure is calculated by determining a coherent non-directional energy proportion for each frequency band b given by

$$r_{cohnde}(b,m) = r_{diff}(b,m)\gamma(b,m)$$

where $\gamma(b,m)$ is the surround coherence and $r_{diff}(b,m)$ is the diffuse-to-total energy ratio which represents the non-direction energy ratio for the frequency band and is determined using the direct-to-total energy ratio r_{dir} for the band b.

In the case of one direction per sub band the diffuse-to-total energy ratio $r_{diff}(b,m)$ is given by

$$r_{diff}(b,m) = 1 - r_{dir}(b,m)$$

In the case of two directions per sub band the diffuse-to-total energy ratio $r_{diff}(b,m)$ is given by

$$r_{diff}(b,m) = 1 - \sum_{i=0}^{1} r_{dir}(b,m,i)$$

The frequency band b can refer to a MASA frequency band or to a coding band, depending on where coherence significance measure is used.

A first initial coherence significance measure is determined by

$$\xi_1(m) = \frac{\sum_{b=0}^{B-1} r_{cohnde}(b,m)}{B}$$

where B is the number of frequency bands.

A second initial coherence significance measure is determined by

$$\xi_2(m) = 0.4 \frac{\sum_{b=0}^{B-1} r_{cohnde}(b,m) r_{diff}(b,m)}{\sum_{b=0}^{B-1} r_{diff}(b,m)}$$

Using them, the coherence significance measure is determined by

$$\xi(m) = \max\left(\xi_1(m), \xi_2(m)\right)$$

If the number of subframes m is one, the coherence significance measure $\xi(m)$ is then checked against a fixed threshold of 0.1. If the result of the checking indicates that the coherence significance measure $\xi(m)$ is larger than the threshold, then it is determined that the surround coherence values for this frame are significant, and the surround coherence values $\gamma(b,m)$ for the frequency bands b of this frame are encoded. If the coherence significance measure $\xi(m)$ value is not larger than the threshold of 0.1, then it is determined that the surround coherence values are not significant, and the surround coherence values $\gamma(b,m)$ are not encoded for this frame.

If the number of subframes m is four, the coherence significance measure $\xi(m)$ is then checked against a fixed threshold of 0.1 for all subframes m. If the result of the checking indicates that the coherence significance measure $\xi(m)$ is larger than the threshold for any of the subframes m, then it is determined that the surround coherence values for this frame are significant, and the surround coherence values $\gamma(b,m)$ for the frequency bands b of this frame are encoded. If none of the coherence significance measure $\xi(m)$ values for the different subframes m is not larger than the threshold of 0.1, then it is determined that the surround coherence values are not significant, and the surround coherence values $\gamma(b,m)$ are not encoded for this frame.

5.5.3.2.4 Common MASA metadata codec configuration

The common part of MASA metadata codec configuration sets the metadata codec configuration variables based on: detected metadata composition, number of transport channels, MASA metadata source, and total bitrate for coding audio signals and metadata. The output configuration variables contain: number of coding bands B_{coding} (stored in config variable “numCodingBands”), number of coding bands with two sets of directional parameters B_{2dir} (stored in config variable “numTwoDirBands”), bit budget for metadata coding (stored in config variable “max_metadata_bits”), boolean value to instruct merging of ratio values over subframes (direct-to-total ratio and surrounding coherence, stored in config flag “mergeRatiosOverSubframes”), band mapping from coding bands to full MASA metadata frequency bands (stored in array “band_mapping”), and boolean variable to define if coherence parameter values should be transmitted (stored in config variable “useCoherence”). In general, this process sets both encoder and decoder into similar pre-defined configuration state based on the codec operating point.

The configuration process is detailed in the following text and the overview of the resulting outputs are detailed in table 5.5-3. The table details how the metadata bit budget, number of coding bands, and number of bands with two directional parameter sets depends on the bitrate, the detected subframe structure, and the number of transport channels. This table presents an overview of the configuration, and the details are shown in the process described below.

Table 5.5-3: MASA parameters coding configuration

Bitrate [kbps]	Metadata max bits		Nr. Subbands nbands		Nr. subbands with 2 Directions	
		LR stereo		Joined sf mode		Joined sf mode
13.2	50	50	5	5	0	0
16.4	60	50	5	5	0	0
24.4	70	60	5	5	0	0
32	85	70	5	8	0	0
48	140		5	12	0	0
64	180		5	12	1	2
80	220		5	12	1	2
96	256		5	18	1	3
128	350		8	18	3	4
160	432		12	18	4	6
192	528		18	18	6	8
256	832		24	24	6	9
384	1024		24	24	9	12
512	1300		24	24	24	24

The configuration process starts with setting B_{coding} and B_{2dir} to default value of zero. Then, a bit budget table is selected from ROM tables based on the IVAS total bitrate and number of transport channels and assigned to "masa_bits_table". If the given total bitrate is less than 48 kbps and there are two transport channels, then "masa_bits_LR_stereo"-table is selected and assigned. Otherwise, the general "masa_bits"-table is used. Notably, a check is also made for the source format of the MASA metadata used in this encoding in the form of argument "isMcMasa" which is used to select McMASA bit budget table “mcmasa_bits” when it is true. This is described in 5.7.3.6.1. When the value is false, it is assumed that the source format for the metadata is generic MASA format.

With the bit budget table selected, a search is performed to find the bitrate table index i_{BR} which fulfills the condition

$$BR_{total} \leq BR_{table}(i_{BR})$$

Then a check is made if $BR_{total} < 48\ kbps$ and number of tranports channels is 2, and $i_{BR} > 3$. If true, then i_BR is set to 3.

The value i_{BR} is then copied to band table index i_{bands}. If number of valid directional parameter sets is more than 1 (in practice, 2), then number of bands for two directional parameter sets B_{2dir} is set from a ROM table with values presented in table 5.5-3 using the table index i_{BR}. The ROM table for the data is selected based on if the detected subframe structure is "1sf" or if it is "4sf" with the tables being respectively "masa_twodir_bands_joined" and "masa_twodir_bands".

Next, i_{bands} is decreased by one if either of the following conditions is true:

- $BR_{total} > 96\ kbps$ and detected subframe structure is "4sf"
- $BR_{total} > 80\ kbps$ and detected subframe structure is "1sf"

Then, i_{bands} is used to obtain the value B_{coding} for coding bands from a ROM table which is again selected based on if the detected subframe structures is "1sf" or if it is "4sf" with the tables being respectively "masa_joined_nbands" and "masa_nbands".

After setting the bands, the maximum metadata coding bit budget is obtained from the previous selected bit budget table using table index i_{BR}. This forms the base bit budget which is further adjusted as follows:

- If the total bitrate is 64 kbps and there are more than one valid directional parameter sets present, then maximum metadata bit budget is increased by 40 bits.
- If the total bitrate is 32 kbps and there are two transport channels, then maximum metadata bit budget is increased by 10 bits
- If the total bitrate is 48 kbps and detected subframe structure is "1sf", then maximum metadata bit budget is increased by 10 bits

Next, several processing configuration variables are set. First, configuration flag for merging ratio values over subframes is set such that if detected subframe structure is "4sf", then ratio values are merged over subframes, otherwise, no merging is required. This is stored in flag “mergeRatiosOverSubframes”. However, if total bitrate for MASA format is at least 384 kbps, then “mergeRatiosOverSubframes” is always false. Second, a predefined band mapping table is selected from ROM tables based on the value B_{coding} obtained above. The tables present in the ROM are:

```
MASA_band_mapping_24_to_5
MASA_band_mapping_24_to_8
MASA_band_mapping_24_to_12
MASA_band_mapping_24_to_18
```

and the mapping from 24 bands to 24 band is implicit one-to-one mapping.

This band mapping table provides information how the reduced number of wider bandwidth coding bands map to the full number of 24 narrower bandwidth frequency bands in MASA format. In the case of 24 coding, a one-to-one mapping table is populated. The band mapping table is used in further encoding process to combine parameters set from frequency bands to coding bands.

Finally, "useCoherence" configuration variable is set as follows:

- False, if the source format for the metadata is McMASA and bitrate is less than 16.4 kbps
- False, if the source format for the metadata is MASA and bitrate is less than 48 kbps
- True, otherwise

This configuration variable controls if coherence parameters are to be transmitted without considering their presence in the metadata.

5.5.3.2.5 Energy ratio compensation

The MASA format spatial metadata parameter values are sanitized for invalid energy ratio values and the energy ratio values are compensated before further encoding. In practice, metadata transmission is done to a reduced selection of all energy ratios (direct-to-total energy ratios, diffuse-to-total energy ratio, and remainder-to-total energy ratio) to optimize bit usage. The assumed model in input MASA metadata follows

$$r_{dir1} + r_{dir2} + r_{diff} + r_{rem} = 1$$

Where r_{dir1} and r_{dir2} are the direct-to-total (or direct) energy ratios, r_{diff} is the diffuse-to-total (or diffuseness) energy ratio, and r_{rem} is the remainder-to-total (or remainder) energy ratio. These ratios also directly represent the signal energy portions of direct, diffuse, and remainder energy of the total signal energy for each frequency band and time subframe (i.e., time-frequency tile).

The input for this step is all the MASA metadata parameter values (above energy ratios, associated direction parameters, and coherence parameters) obtained from the stored values provided in MASA metadata input (see clause 5.5.2). The output is MASA metadata parameters where energy ratios are modified and reduced to a smaller selection of the energy ratios.

The sanitation step is done for each time-frequency tile separately. First, a check is made if the sum of existing energy ratios is zero. If true, it assumed that the values are not correct and there is no proper data available. In this case, r_{diff} is set to 1 and all other energy ratio values for the TF-tile are set to zero. This is the safest adjustment when considering rendering.

If the energy ratio values were valid in the first check, then a check is made if they sum to a total sum of one as is assumed. If not, then they need to be normalized to sum to one.

Finally, remainder-to-total ratio is selected to not being transmitted and it is set to 0 by default. This can result in the assumed sum of ratios being less than 1 which would mean that energy would be missing from the ratios. Thus, the direct-to-total ratios and diffuse-to-total ratio is normalized with the combination of them, that is,

$$r_{comb} = r_{dir1} + r_{dir2} + r_{diff}$$
$$r_{dir1} := \frac{r_{dir1}}{r_{comb}}$$

$$r_{dir2} := \frac{r_{dir2}}{r_{comb}}$$
$$r_{diff} := \frac{r_{diff}}{r_{comb}}$$

This results in weighted distribution of the remainder portion of energy to directional and diffuse streams in decoding and rendering where these streams are created using energy weights and energy ratio weights (see clauses 6.5, 6.5.7, and 7.2.2.3). As an optimization, the above two normalizations can be done together by simply first setting the remainder-to-total energy ratio value to zero.

5.5.3.2.6 Merging of MASA spatial parameter metadata across MASA frequency bands and subframes

This clause describes the merging of MASA spatial audio parameter sets across frequency bands and temporal subframes.

When merging is performed across MASA frequency bands, a metric is determined as the number of coding frequency bands (B_{coding}, see clause 5.5.3.2.4). This metric is then used to determine whether to merge the spatial audio parameter sets for two or more MASA frequency bands into a merged spatial parameter set. The merged spatial parameter set (over MASA frequency bands) corresponds to a single coding frequency band.

The decision to merge is based is based on whether the metric is smaller than the threshold value of the number of frequency bands in the MASA format (i.e., 24).

The merging is performed for the spatial audio parameter sets of one or more MASA frequency bands, resulting in fewer spatial audio parameter sets than the number of MASA frequency bands. The number of spatial audio parameter sets remaining after the merging process is aligned to the number of coding frequency bands.

The spatial audio parameters sets of consecutive MASA frequency bands are merged by initially converting the azimuth $\theta(b_M, m, i)$ and elevation $\phi(b_M, m, i)$ angles to Cartesian coordinate vectors. The length of the vectors is determined based on the direct-to-total energy ratio $r_{dir}(b_M, m, i)$ and the energy $E(b_M, m)$. This is performed for MASA spatial audio parameter sets which are to be merged by

$$x(b_M, m, i) = \cos\theta(b_M, m, i)\cos\phi(b_M, m, i)\, l(b_M, m, i)$$
$$y(b_M, m, i) = \sin\theta(b_M, m, i)\cos\phi(b_M, m, i)\, l(b_M, m, i)$$
$$z(b_M, m, i) = \sin\phi(b_M, m, i)\, l(b_M, m, i)$$

where

$$l(b_M, m, i) = r_{dir}(b_M, m, i)E(b_M, m)$$

where b_M is the MASA frequency band index, m is the subframe index, and i is the direction index.

The merging process involves mapping a number of MASA frequency bands to a coding frequency band, and then merging the MASA spatial audio parameter sets associated with each of the mapped MASA frequency bands into a single spatial audio parameter set corresponding to the coding frequency band. This is performed, for each mapping of MASA frequency bands to coding frequency band, by summing the vectors and energies corresponding to the MASA frequency bands of the mapping

$$x_{sum}(b, m, i) = \sum_{b_M = b_{M,1}(b)}^{b_{M,2}(b)} x(b_M, m, i)$$
$$y_{sum}(b, m, i) = \sum_{b_M = b_{M,1}(b)}^{b_{M,2}(b)} y(b_M, m, i)$$
$$z_{sum}(b, m, i) = \sum_{b_M = b_{M,1}(b)}^{b_{M,2}(b)} z(b_M, m, i)$$

$$E_{sum}(b,m) = \sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} E(b_M,m)$$

where $b_{M,1}(b)$ is the first mapped MASA frequency band of the coding frequency band b and $b_{M,2}(b)$ the last mapped MASA frequency band.

The merged direction for the coding frequency band b is then determined by

$$\theta(b,m,i) = \arctan(y_{sum}(b,m,i), x_{sum}(b,m,i))$$

$$\phi(b,m,i) = \arctan(z_{sum}(b,m,i), \sqrt{x_{sum}(b,m,i)^2 + y_{sum}(b,m,i)^2})$$

where $\arctan(,)$ is a variant of arcus tangent that can determine the right quadrant.

Then, the initial merged direct-to-total energy ratio corresponding to the coding frequency band b is determined by

$$r'_{dir}(b,m,i) = \frac{\sqrt{x_{sum}(b,m,i)^2 + y_{sum}(b,m,i)^2 + z_{sum}(b,m,i)^2}}{E_{sum}(b,m)}$$

The merged spread coherence corresponding to the coding frequency band b is determined by

$$\zeta(b,m,i) = \frac{\sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} \zeta(b_M,m,i)E(b_M,m)}{E_{sum}(b,m)}$$

The initial merged surround coherence corresponding to the coding frequency band b is determined by

$$\gamma'(b,m) = \frac{\sum_{b_M=b_{M,1}(b)}^{b_{M,2}(b)} \gamma(b_M,m)E(b_M,m)}{E_{sum}(b,m)}$$

In the case the number of coding bands (i.e., numCodingBands) is equal to the number of frequency bands in the MASA format (i.e., 24), the frequency combination need not be performed. The input values are used in the subsequent processing directly, i.e.,

$$\theta(b,m,i) = \theta(b_M,m,i)$$

$$\phi(b,m,i) = \phi(b_M,m,i)$$

$$r'_{dir}(b,m,i) = r_{dir}(b_M,m,i)$$

$$\zeta(b,m,i) = \zeta(b_M,m,i)$$

$$\gamma'(b,m) = \gamma(b_M,m)$$

$$E_{sum}(b,m) = E(b_M,m)$$

It is next checked whether the mergeRatiosOverSubframes flag is enabled (see clause 5.5.3.2.4). In the case mergeRatiosOverSubframes is enabled, the direct-to-total energy ratio and surround coherence values for different subframes (there are four subframes m in the MASA metadata) are merged over subframes, resulting in merged direct-to-total energy ratio and surround coherence values for the frame. I.e., in this case, the frame contains only one coding subframe. The merging of subframes is performed separately for all coding frequency bands b. The other parameters are not merged over subframes.

The merging of spatial parameter values over subframes is performed by first summing the energies over subframes

$$E_{sum}(b) = \sum_{m=0}^{3} E_{sum}(b,m)$$

Then, the surround coherence is merged over subframes by

$$\gamma(b) = \frac{\sum_{m=0}^{3} \gamma'(b,m)E_{sum}(b,m)}{E_{sum}(b)} \quad (5.5\text{-}2)$$

and this value is set to all subframes m yielding the merged surround coherence $\gamma(b,m)$.

The direct-to-total energy ratio is merged over subframes by

$$r_{dir}(b) = \frac{\sum_{m=0}^{3} r'_{dir}(b,m) E_{sum}(b,m)}{E_{sum}(b)} \quad (5.5\text{-}3)$$

and this value is set to all subframes m yielding the merged direct-to-total energy ratio $r_{dir}(b,m)$.

In the case when the subframes are not merged (i.e., when mergeRatiosOverSubframes flag is not enabled), the initial values are used as the output

$$\gamma(b,m) = \gamma'(b,m)$$

$$r_{dir}(b,m) = r'_{dir}(b,m)$$

5.5.3.2.7 Combining of MASA spatial audio metadata across multiple directions

This clause describes the combining of multiple MASA spatial audio parameter metadata of the same type for a frequency band, where each of the MASA spatial audio parameters which are combined correspond to a different direction.

The input to the combining process is the MASA spatial audio parameters corresponding to two directions for each frequency band b. In effect, the combining process receives two-direction MASA spatial metadata, for each frequency band b, consisting of two azimuth $\theta(b,m,i)$, two elevation $\phi(b,m,i)$, and two spread coherence $\zeta(b,m,i)$ spatial audio parameters, where i is the direction index and holds the values of 0 and 1 corresponding to the two different directions. For a frequency band b, the two azimuth values, the two elevation values and the two spread coherence values are then each combined into a single value, giving a combined azimuth value, a combined elevation value, and a combined spread coherence value.

Initially, the azimuth $\theta(b,m,i)$ and elevation $\phi(b,m,i)$ angles are converted to Cartesian coordinate vectors. The length of the vectors is determined based on the direct-to-total energy ratio $r_{dir}(b,m,i)$. This is performed for each direction i (where m and b correspond to the specific subframe and frequency band in which the combining takes place)

$$x(b,m,i) = \cos\theta(b,m,i)\cos\phi(b,m,i)\, r_{dir}(b,m,i)$$

$$y(b,m,i) = \sin\theta(b,m,i)\cos\phi(b,m,i)\, r_{dir}(b,m,i)$$

$$z(b,m,i) = \sin\phi(b,m,i)\, r_{dir}(b,m,i)$$

Then, the vectors are summed over the two directions

$$x_{sum}(b,m) = \sum_{i=0}^{1} x(b,m,i)$$

$$y_{sum}(b,m) = \sum_{i=0}^{1} y(b,m,i)$$

$$z_{sum}(b,m) = \sum_{i=0}^{1} z(b,m,i)$$

and the length of the sum vector is computed by

$$l_{sum}(b,m) = \sqrt{x_{sum}(b,m)^2 + y_{sum}(b,m)^2 + z_{sum}(b,m)^2}$$

Then, an importance metric is determined, specific to the frequency band b, which represents the importance of having two directions instead of one for the frequency band b. It is determined by

$$\lambda(b) = \frac{\sum_{m=0}^{3}\left(r_{dir}(b,m,0) + r_{dir}(b,m,1) - l_{sum}(b,m)\right)}{M}$$

where M is the number of subframes (i.e., 4).

Then, based on the importance metric $\lambda(b)$, it is determined whether the encoded values for the frequency band b comprises the original azimuth, elevation, and spread coherence for each separate direction, or whether the encoded values comprise the combined azimuth, elevation, and spread coherence. The importance metric $\lambda(b)$ is determined for all frequency bands.

Determining whether to encode two separate spatial audio parameter values or combined spatial audio parameter value for a particular frequency band b is performed with the use of a "two-direction frequency band" $\beta(b)$ variable, which has the value of 1 for the frequency bands that use the two directions and the value of 0 for the frequency bands that use a combined (single direction) value. The value of the $\beta(b)$ variable, for each frequency band b, is determined by sorting the values of $\lambda(b)$ for all frequency bands in order of magnitude to determine the B_{2dir} frequency bands that have the largest values of $\lambda(b)$.

B_{2dir} corresponds to the factor twoDirBands determined in clause 5.5.3.2.4. For the B_{2dir} frequency bands b which have the largest values of $\lambda(b)$, $\beta(b)$ is set to 1. For the other frequency bands b, $\beta(b)$ is set to 0.

Then, for the frequency bands b where $\beta(b) = 0$, the directions are combined as follows.

The combined azimuth and elevation angles are determined by

$$\theta(b,m) = \arctan(y_{sum}(b,m), x_{sum}(b,m))$$

$$\phi(b,m) = \arctan(z_{sum}(b,m), \sqrt{x_{sum}(b,m)^2 + y_{sum}(b,m)^2})$$

Then, a ratio sum variable is determined as

$$r_{dir,sum}(b,m) = \sum_{i=0}^{1} r_{dir}(b,m,i)$$

Then, an ambient energy value is determined using the direct-to-total energy ratios by

$$r_{amb,2dir}(b,m) = 1 - \sum_{i=0}^{1} r_{dir}(b,m,i)$$

Then, the combined direct-to-total energy ratio is determined using the ratio sum variable, the ambient energy value, and the length of the sum vector by

$$r_{dir}(b,m) = \frac{l_{sum}(b,m)}{r_{dir,sum}(b,m) + 0.5 r_{amb,2dir}(b,m)}$$

Then, the combined spread coherence is determined by

$$\zeta(b,m) = \frac{\sum_{i=0}^{1} \zeta(b,m,i) r_{dir}(b,m,i)}{r_{dir,sum}(b,m)}$$

Then, a further ambient energy value (corresponding to the combined direct-to-total energy ratio) is determined by

$$r_{amb,comb}(b,m) = 1 - r_{dir}(b,m)$$

Then, original surround coherence energy and new surround coherence energy variables are determined as follows

$$\gamma_{origene}(b,m) = r_{amb}(b,m)\gamma(b,m)$$

$$\gamma_{newene}(b,m) = \max\left(r_{amb,comb}(b,m) - r_{amb,2dir}(b,m), 0\right)\zeta(b,m)$$

The surround coherence is then adjusted to a new value representing the combined surround coherence by using the original surround coherence energy and new surround coherence energy

$$\gamma_{comb}(b,m) = \min\left(1, \frac{\gamma_{origene}(b,m) + \gamma_{newene}(b,m)}{r_{amb,comb}(b,m)}\right)$$

5.5.3.2.8 Metadata reductions for low rate

At the lowest bitrates where the bit budget for MASA metadata coding is low and the detected subframe structure is not "1sf" (i.e., already reduced in time), a further reduction in time-frequency domain is done to the MASA metadata parameter values before encoding the spatial metadata parameter values. In practice, either time-domain or frequency-domain dimension is reduced to one. This is done to reduce the bitrate used by the parameter values in metadata encoding, thus avoiding running out of bits when encoding the parameter values.

The decision on using further reduction is based on the metadata bit budget and the detected subframe structure. If the detected subframe structure is "1sf", then no further reduction can be done. If the detected subframe structure is "4sf" and the configured metadata bit budget is less than a predefined value of MINIMUM_BIT_BUDGET_NORMAL_META bits, then further metadata reduction is done.

The received input to the further metadata reduction are MASA metadata parameter values in already-reduced coding band TF-resolution. In practice, these parameters are always obtained for each of the 5 frequency bands and 4 temporal subframes and the direct-to-total energy ratio and the surrounding coherence ratio has been merged through time. The input and output metadata parameter values at this point contain: direct-to-total energy ratios, direction, spread coherence, and surround coherence. In addition, total transport audio signal energy is provided in the resolution of 24 frequency bands and 4 temporal subframes in the case of MASA format or OMASA combined format, or in the resolution of 5 frequency bands and 4 temporal subframes in the case of McMASA analysed MC-format. The energy has been calculated beforehand from the associated transport audio signals of the input MASA format (see clause 5.5.2.3, where audio signals have been obtained from IVAS encoder common processing, see clause 5.1) or determined as part of McMASA analysis from input multichannel signals (see clause 5.7.3.3). The output from the further metadata reduction is metadata parameter values with time-frequency resolution reduced to either 5 frequency bands and 1 temporal subframe or 1 frequency band and 4 temporal subframes.

In the case that energy is not already in the TF-resolution of 5 frequency bands and 4 temporal subframes, it is summed into correct format using equation

$$E(b,m) = \sum_{b_{24}=b_{map}(b)}^{b_{map}(b+1)-1} E_{24}(b_{24},m)$$

where E_{24} is the energy in the resolution of 24 frequency bands and 4 temporal subframes, b_{24} is band index for the 24 frequency bands, and b_{map} is the band mapping function from 5 frequency coding bands to 24 MASA frequency bands.

In addition, total sum energy is calculated by summing the energy over all time-frequency tiles:

$$E_{tot} = \sum_{m=0}^{M} \sum_{b=0}^{B} E(b,m)$$

Where M is the total number of temporal subframes and B is the total number of frequency coding bands.

Next, onset detection is performed to produce an onset metric which defines the probability that there is a start of a sound event present in the current metadata frame. The onset metric is formed with a relation of a fast and slow envelopes of the audio signal energy. These envelopes and the resulting onset metric are formed as follows:

$$E_\alpha = \max(\alpha E_\alpha^{-1}, E_{tot})$$
$$E_\beta = \gamma \min(E_\alpha, \beta E_\beta^{-1} + (1-\beta)E_\alpha^{-1})$$
$$o = \min\left(1, \frac{E_\beta}{E_\alpha}\right)$$

Here, o is the onset metric, E_α and E_β are the fast and slow signal energy envelopes respectively. α, β, γ are tuning factors for the onset detection.

If the onset metric indicates that there is a start of a sound event present in the current metadata frame, that is, $o < 0.99$, then a merge metric is determined to control if a further merge should be done in time or frequency domain

To determine the merge metric, first, it is necessary to determine a single frequency band to represent all frequency bands. This is done by first determining a further respective parameter energy weight factor $w_E(b)$ for each frequency band with

$$w_E(b) = \frac{E_{band}(b)}{M} * r_{dir}(b)$$

where M is the total number of subframes. Thus, the energy weight factor is determined for each frequency band based on the energy ratio and energy of the band, i.e., it is a product of the total energy of the band $E_{band}(b)$ and the direct-to-total energy ratio $r_{dir}(b)$ of the band. The total energy of a band $E_{band}(b)$ is obtained by summing the energy over subframes, that is,

$$E_{band}(b) = \sum_{m=0}^{M} E(b,m)$$

Next, a threshold value is determined as a weight limit factor w_{thr} based on the averaged energy, that is,

$$w_{thr} = \frac{E_{tot}}{BM} 0.5$$

where the constant value of 0.5 signifies an average energy ratio.

The energy weight factor $w_E(b)$ for each frequency band is then compared to the weight limit factor w_{thr} starting from the highest band. The highest frequency band where the energy weight factor $w_E(b)$ is greater than energy the weight limit factor w_{thr} is selected.

Next, the direct-to-total energy ratio of the selected frequency band is compared to an energy-weighted mean of the direct-to-total energy ratios of all frequency bands in the metadata frame. The mean ratio is obtained as

$$r_{mean} = \frac{1}{E_{tot}} \sum_{m=0}^{M} \sum_{b=0}^{B} r_{dir}(b,m) E(b,m)$$

If $r_{dir}(b) > r_{mean}$ for the selected band, then the merge metric is determined to merge over frequency bands. Otherwise, the merge metric is configured to merge over the time subframes. In practice, merging over frequency bands is selected when there is a probable onset present in the current frame and a single frequency band can sufficiently represent the metadata parameters for all frequency bands.

If the merge metric is configured to merge through time subframes or the onset metric indicates an absence of a sound event, then merging through time is done to direction parameters and spread coherence parameters. As mentioned above, direct-to-total energy ratio parameters and spread coherence parameters have been already merged through time. Direction parameters are merged through time with the method described in clause 5.5.3.2.3.3.3. Spread coherence parameters are merged through time as an energy-weighted average of the spread coherence parameters on the frequency band. Additionally, configuration is set such that the subframe structure is "1sf" and the corresponding configuration parameter in the quantized metadata encoder structure is set to 1. Thus, the time-merged metadata parameters represent the metadata parameters for all time subframes.

If the merge metric is configured to merge through frequency bands and the onset metric indicates a start of a sound event, then the metadata parameters of the selected band (and all of its time subframes) are used to represent the metadata parameters for all frequency bands. In practice, the metadata parameters of the selected band are saved into the space of the metadata parameters of the first frequency band and the metadata parameters of any or all other frequency bands are discarded. The configuration parameter for number of coding bands in the quantized metadata encoder structure is set to 1. Thus, an output is generated for further encoding which contains the parameters of the one selected frequency band and none of the discarded parameters. The parameters of the selected frequency band represent all the respective parameters of the original 5 frequency bands. Encoding only the parameters of the selected frequency band reduces the required bitrate to encode metadata. This further reduced set of metadata parameters is then given for metadata encoding with the common tools presented in clause 5.2.4.

The configured merge metric is also encoded in the bitstream with an indicator bit to indicate if the merging has been done through time or through frequency. This allows correct decoding of the metadata bitstream and reconstruction of the metadata parameter values in the decoder (see clause 6.5.3.7).

5.5.3.2.9 Reordering of directional data with two concurrent directions

In case of MASA format metadata where at least some frequency bands are configured to encode two sets of directional spatial metadata parameters, an analysis and reordering step is performed before quantization and encoding of the metadata parameters. This is done to optimize the encoding of the directional metadata parameters by fulfilling

assumption that for any time frequency tile or frequency band, $r_{dir1} \geq r_{dir2}$, that is, the direct-to-total energy ratio parameter of the first direction is larger than or equal to the direct-to-total energy ratio of the second direction..

In practice, for each frequency coding band containing two sets of directional parameters, a comparison is made to determine if $r_{dir2} > r_{dir1}$, that is, the direct-to-total energy ratio parameter of the second direction is larger than the direct-to-total energy ratio of the first direction. If this is true, then the directional spatial metadata parameters of the first direction are swapped with the directional spatial metadata parameters of the second direction for each time-frequency tile belonging to this frequency coding band. This is swapping done for direct-to-total energy ratio, direction, and spread coherence parameters and is achieved with the following pseudocode algorithm:

```
For subframes in 2dir band
    temp_value = dir1_parameter
    dir1_parameter = dir2_parameter
    dir2_parameter = temp_value
```

5.5.3.2.10 Signalling bits

In the encoding of the MASA input format there are several signalling bits that help transmit the information on the encoded format structure. The information they convey together with the number of bits on which they are represented is presented in the list below:

- The number of MASA transport channels, 1 or 2, written in 1 bit
- Two reserved bits
- The number of audio directions, 1 or 2, written in 1 bits
- The subframe mode, which signals 1 or 4 subframes

The low bit rate encoding mode is signalled with 1 bit only when the maximum allowed number of bits for MASA metadata is lower than 100 and when there are 4 temporal subframes. Under these conditions, the low bitrate mode is activated for IVAS total bitrates of 32 kbps and below.

5.5.3.2.11 Adjustment of highest transmitted MASA metadata band

Next, the configured number of coding bands and the band mapping from coding bands to MASA frequency bands is adjusted according to the sample rate and determined cut off frequency of the transport audio signals. As the spatial audio parameter sets of the MASA metadata are always determined for the 24 frequency sub bands (as obtained in clause 5.5.2.1), there can be spatial audio parameters sets present that can be removed from the transmission as there is no energy present in the associated frequency sub bands of the associated transport audio signals.

This adjustment is done such that first, the expected coding rate associated with the transport audio signals is received (i.e., total IVAS bitrate). As presented above, this coding rate is used in clause 5.5.3.2.4 to provide a band mapping which maps multiple consecutive frequency sub bands to broadened frequency sub bands (i.e., maps MASA frequency bands to coding bands). This is a coding rate adjusted band mapping which contains coding rate adjusted number of frequency sub bands.

Then, the bandwidth value (i.e., sample rate f_s) associated with the transport audio signals is received from encoder configuration. This is used to obtain the highest frequency bin k_{max} representable considering f_s and the bandwidth BW of the frequency resolution. This is

$$k_{max} = \frac{f_s}{BW_{CLDFB}}$$

where $BW = 800\,Hz$ and corresponds also to the bandwidth of the CLDFB filters (see clause 6.2.5). Next, the lowest MASA frequency band is selected which still can contain the maximum frequency bin k_{max}. This value is assigned as the initial maximum frequency band b_{max}. Using the value b_{max} effectively removes, starting from the highest frequency sub band of the coding rate adjusted number of sub bands, a number of frequency sub bands that are above b_{max}. This gives the bandwidth adjusted number of sub bands and accordingly the band mapping. In case the high border of the highest frequency sub band (i.e., b_{max}) extends beyond the k_{max} (maximum bandwidth based on f_s), then the highest frequency sub band is adjusted such that it lies below k_{max}.

Next, if the coding rate is at least 384 kbps, then a cut off frequency sub band is determined. This is done by determining the highest frequency bin which has an energy level greater than a predetermined level (i.e., there is

sufficient energy present) based on the energy determined for each frequency bin of the transport audio signals in clause 5.5.2.3. The cut off frequency sub band is assigned as the one which incorporates the highest frequency bin. The minimum value for the cut off frequency sub band is 5.

If the cut off frequency sub band is less than the highest frequency sub band based on f_s, then b_{max} is reduced further such that the spatial audio parameter sets frequency sub bands containing significant energy are transmitted and the spatial audio parameter sets corresponding to all the higher frequency bands are removed (based on combined effect of the cut off frequency band, bandwidth value, and coding rate). The band mapping is also adjusted based on the cut off frequency band if necessary. This final adjusted number of frequency sub bands and the final adjusted band mapping is provided to the merging function in clause 5.5.3.2.6 where multiple spatial audio parameter sets of the MASA frequency bands are merged to give fewer spatial audio parameter sets over the broadened coding bands. The final adjusted number of frequency sub bands is also given to the MASA metadata encoding functions in clause 5.5.3.3. In practice, B_{coding} in clause 5.5.3.2.2 is set to b_{max} and the corresponding adjusted band mapping is stored for use.

In case the cut off frequency sub band reduction is used (bitrate is at least 384 kbps), then an encoding of the index of the cut off frequency sub band is performed in clause 5.5.3.3 by encoding the number of sub bands with no significant energy (i.e., inactive bands).

5.5.3.3 MASA metadata quantization and encoding

5.5.3.3.1 Overview

The encoding flow of operations in the MASA metadata is described below:

1. If the IVAS bitrate is larger or equal to 384 kbps the number of sub higher bands with no audio content is encoded with a Golomb Rice code of order 1
2. If the bitrate is larger or equal to 48 kbps, one bit is used to signal if coherence is present, based on the coherence presence detection from clause 5.5.3.2.3.5.
3. If there are more than 1 direction in the MASA metadata
    a. The metadata for the two directions is reordered according to clause 5.5.3.2.9.
    b. The indexes of the sub bands for which 2 directions are encoded are transmitted by Golomb Rice of order 0 encoding of the index differences minus 1 unit of the consecutive sub band positions.
4. End if
5. The energy ratios are quantized and encoded according to clause 5.5.3.3.2.
6. The surround coherence, if present, is encoded according to clause 5.5.3.3.5.
7. The direction dependent metadata is encoded as follows:
    a. For each direction d=1:2
        i. If d == 2
            1. Transform the azimuth value according to the procedure described in clause 5.5.3.3.6.1
        ii. End
        iii. Encode spread coherence according to the procedure described in clause 5.5.3.3.4
        iv. Quantize and encode directional metadata (azimuth and elevation)
    b. End for

5.5.3.3.2 Energy ratio encoding

5.5.3.3.2.1 Energy ratio encoding for bitrates up to 256 kbps

The energy ratios of the first direction are quantized and encoded according to clause 5.2.4.4.2. If two directions are present and selected for encoding, the energy ratios of the second direction TF tiles are encoded according to the description from clause 5.2.4.4.4.

5.5.3.3.2.2 Energy ratio encoding for 384 kbps and 512 kbps

For IVAS total bitrates of 384 kbps and 512 kbps the energy ratios are quantized on 4 bits using the corresponding codebook from table 5.2-21. The encoding at 384 kbps is performed similarly to the encoding done for bitrates lower than 384kbps, and the encoding of the energy ratios at 512 kbps is performed independently for each direction.

5.5.3.3.3 Direction encoding

5.5.3.3.3.1 Direction encoding for bitrates up to 256 kbps

The direction encoding consists of the encoding of the azimuth and elevation values, that are defined for each TF tile. The quantization is done according to the bit allocation given by the indexes of the quantized corresponding energy ratios as presented in table 5.2-20.

The encoding flow of the directional parameters at IVAS total bitrates up to 256 kbps is presented below:

1. Quantize directional metadata (elevation and azimuth)
2. Signal using 1 bit if the directional data is planar or not
3. Encode the azimuth and elevation values using the EC1 method presented in clause 5.2.4.5.1 and count the number of bits that would be used for it, *bits_EC1*
4. If *bits_EC1* $\leq$ the allowed number of bits for encoding directional metadata (azimuth and elevation)
    a. Reset bitstream
    b. Try encoding using method EC2 presented in clause 5.2.4.5.2 and calculate the required number of bits *bits_EC2*
    c. If *bits_EC2* $\leq$ the allowed number of bits for encoding directional metadata (azimuth and elevation)
        i. Use EC2 method
    d. Else
        i. Use EC3 with the reduced quantization resolution
    e. End
5. Else
    a. Let the bitstream with the EC1 encoded data
6. End

5.5.3.3.3.2 Direction encoding for 384 kbps and 512 kbps

When the IVAS total bitrate is 384 kbps or 512 kbps the quantization resolution is improved using 11 bits for all TF tiles directional parameters at 384 kbps and 16 bits for all TF tiles (similar to the input format data) at 512 kbps.

5.5.3.3.4 Spread coherence encoding

5.5.3.3.4.1 Spread coherence encoding for bitrates up to 256 kbps

There are *nblocks*×*nbands* spread coherence parameters for MASA format. The number of sub bands *nbands* and number of subframes *nblocks* are given in table 5.5-3. In MASA format the spread coherence is encoded for bitrates starting with 48kbps. For the lower bitrates it is set to zero. For the joined sub frames cases the number of sub frames is *nblocks* = 1 and for the rest *nblocks* = 4. The number of spread coherence parameters to be encoded increases accordingly if the second direction is present. The spread coherence parameters of the two directions are encoded independently. For each direction the encoding follows the procedure described in clause 5.2.4.6.1.

5.5.3.3.4.2 Spread coherence encoding for 384 kbps and 512 kbps

For 384 kbps and 512 kbps codec total bitrate, a higher resolution quantization is performed for the spread coherence. The spread coherence parameters for all sub frames and all sub bands are first uniformly scalar encoded on a predefined number of bits (3 bits for IVAS total bitrate equal to 384 kbps, and 4 bits for IVAS total bitrate equal to 512 kbps). For each sub frame the quantization indexes of all sub bands are encoded with a switched encoding mechanism. The switching is made between minimum removed Golomb Rice encoding of the indexes and average removed Golomb Rice encoding of the indexes. When selected, the average index is fixed rate encoded with the number of bits used for the quantization of the spread coherence parameters (3 or 4 bits, depending on overall codec bitrate). The index difference to the average index is transformed through the function in equation (5.2-269) are encoded using the best of 0 or 1 Golomb Rice order. When the minimum removed approach is selected, the minimum index value is transmitted in fixed rate on the number of bits used in the quantization, and the index differences to the minimum value are encoded with the best Golomb Rice order between 0 and 1.

5.5.3.3.5 Surround coherence encoding

5.5.3.3.5.1 Surround coherence encoding for bitrates up to 256 kbps

There are *nblocks*×*nbands* surround coherence parameters for MASA format. The number of sub bands nbands, the number of subframes nblocks are given in table 5.5-3. Similarly to the spread coherence, the surround coherence is encoded for the MASA format for bitrates starting with 48 kbps. For the joined sub frames cases the number of sub frames is *nblocks* = 1 and for the rest *nblocks* = 4. The number of surround coherence parameters is independent of the number of directions. The surround coherence parameters encoding follows the procedure described in clause 5.2.4.6.2.

5.5.3.3.5.2 Surround coherence encoding for 384 kbps and 512 kbps

At higher bitrates of the codec, a more accurate representation of the surround coherence is expected, and it is described in the following. The surround coherence values are no longer averaged over the sub frames and distinct values are transmitted for each sub frame and each sub band. Similarly to the lower resolution surround coherence encoding in clause 5.2.4.6.2 a significance measure for the surround coherence is calculated, but for each sub band *b=1:nbands* and each subframe *j=1:nblocks*.

$$\epsilon_{hr}(b,j) = 1 - \sum_{i=1}^{nD} energy\_ratio_i(b,j) \quad (5.5\text{-}4)$$

The significance measure is this time quantized using the highest resolution codebook from table 5.2-43. The rest of the encoding proceeds similarly to the procedure described in clause 5.2.4.6.2.

5.5.3.3.6 Coding of the second direction parameters

5.5.3.3.6.1 Joint encoding of parameters associated with first and second sound source directions

Joint encoding of spatial audio parameters of a TF tile associated with a first and second sound source direction is performed with respect to the azimuth direction and the direct-to-total energy ratio only. For both these parameters, there is a degree of dependency with respect to the parameter of the first direction and the parameter of the second direction. Joint encoding of these parameters is only performed when the following criteria is met: spatial audio parameters associated with a second direction are present (in addition to spatial audio parameters associated with a first audio direction); and the spatial audio parameters associated with the second audio direction are selected for encoding based on the overall bitrate and operating mode of the IVAS codec.

The azimuth direction parameter associated with the first and second directions are jointly encoded using the following routine. First, quantization and encoding of the azimuth for the first direction is performed according to one of the methods described in clause 5.2.4.5, where the choice of method used is dependent on the total number of bits available. Then, the azimuth value of the second direction, for the TF tile, is first transformed to an opposite spatial audio direction by rotating the direction value through 180 degrees. The difference between the quantized first direction azimuth and the rotated second direction azimuth value for a TF tile is then quantized and encoded using the same method as used for the quantization and encoding of the first direction azimuth value.

The direct-to-total energy ratio for each of the two directions of a TF tile are jointly quantized and encoded according to the procedure described in 5.2.4.4.4.

5.5.3.3.6.2 Independent encoding of parameters associated with first and second sound source directions

For the cases where the parameters associated with the first and second directions are independently considered for encoding, they are encoded following the one direction case encoding for the first direction, followed by the one direction encoding of the parameters corresponding to the second direction.

5.5.4 Encoding of MASA audio transport channels

One or two audio transport channels are encoded for the MASA format, as shown in figure 5.5-1. The encoding bitrate depends on the bitrate allocated between metadata and audio transport channel encoding.

For the mono-MASA case the input audio transport channel is always encoded using the SCE encoder described in clause 5.2.3.2.

For the stereo-MASA case, for IVAS total bitrates higher than 16.4 kbps the two audio transport channels are encoded using the method presented in clause 5.2.3.3. For IVAS total bitrates lower or equal to 16.4 kbps in the stereo-MASA case, the two channels are encoded using the method described in clause 5.2.3.3, using only the DFT stereo mode (clause 5.3.2.4) without transmitting the stereo parameters.

5.5.5 DTX operation

The DTX processing for MASA is based on the common spatial metadata encoder, which is described in clause 5.2.4.7.

The audio transport channel information is encoded on 48 bits using the method presented in clause 5.3.5. The DTX operation for MASA input format reserves 53 bits per frame for the encoding of the MASA metadata. The total bitrate for the DTX mode is 5.2 kbps.

For the metadata part, only the energy ratios and the directional parameters for only one direction are encoded. The metadata is to 5 sub bands for both directions (if present) and then the obtained data is reduced to one direction only using the methods presented in 5.5.3.2.6 and 5.5.3.2.7, respectively. Each sub band has 4 subframes. The energy ratio values for the sub frames of one sub band are the same. The directional parameter values for the sub frames of a sub band may differ.

One bit is used to signal if the metadata is planar or not. The energy ratios are encoded with 2 bits using the originally 3-bit quantized values but quantizing to the same value and index all the 5 higher energy ratio quantized values. Once the energy ratios are quantized, the number of bits needed to encode the energy ratios (10) and the directional parameters can be estimated. Only one average directional parameter (elevation and azimuth) is sent per sub band. The estimation of the number of bits for directional parameters is done using the information from table 5.2-20. If more bits are needed for encoding the MASA metadata than the allocated bits, the difference is gradually subtracted, bit by bit, sub band by sub band from the average directional parameter quantization, until the allowed value for the number of bits is reached. If fewer bits are needed, the excess bits are distributed to the average directional parameters' quantization. After the bit allocation for the directional parameters is set, the average elevation and azimuth values are calculated for each sub band, quantized and raw encoded.

5.5.6 Bitstream structure

Figure 5.5-1 illustrates the MASA format encoder. The encoding significantly separates transport channel coding and metadata coding. On high level, the transmitted bitstream therefore includes two main components, the compressed transport channel(s) and the compressed metadata.

The overall bitstream structure for MASA input format is the following:

- IVAS format information bits
- Audio transport channel bits
- MASA metadata bits
- Number of transport channels in the MASA input format

5.5.7 MASA bitrate switching

The metadata coding configuration is described in clause 5.5.3.2.2. This configuration depends on bitrate, and the metadata configuration is processed in each frame independently. Thus, the correct configuration is selected during bitrate switching.

When bitrate is changed, the bit allocation changes for the MASA transport channel coding (see clause 5.5.4). This operation follows the SCE (for MASA1) and CPE (for MASA2) bitrate switching operations.

5.6 Object-based audio (ISM) operation

5.6.1 ISM format overview

The object-based audio represents a complex audio auditory scene as a collection of individual elements, also known as audio objects, such as speech, music, or a general audio sound. In IVAS, an audio object is referred to as an independent audio stream with metadata (ISM) where the audio stream represents, in a bitstream, an audio waveform and consists of a mono channel coded by a SCE. Then metadata represents a set of information describing an audio stream and an artistic intension used to translate the original or coded audio objects to a reproduction system. The metadata describes spatial properties of each individual audio diegetic object, such as position, or an orientation and a panning gain for each individual non-diegetic one. In the following text, two sets of metadata are considered:

- input metadata: unquantized metadata representation used as an input to the IVAS codec; and
- coded metadata: quantized and coded metadata forming part of a bitstream transmitted from the encoder to the decoder.

IVAS supports a simultaneous coding of up to four (4) ISMs. The related bitrates and maximum audio bandwidths at which the ISM coding is supported is summarized in Table 5.6-1 while two ISM coding modes are employed: discrete ISM (further referred to as DiscISM) and parametric ISM (further referred to as ParamISM).

Table 5.6-1: Overview of bitrates, bandwidths, and coding modes in ISM format

IVAS bitrate [kbps]	number of ISMs			
	1	2	3	4
13.2	DiscISM – SWB	n/a	n/a	n/a
16.4	DiscISM – FB	DiscISM – WB	n/a	n/a
24.4	DiscISM – FB	DiscISM – SWB	ParamISM – SWB	ParamISM – SWB
32	DiscISM – FB	DiscISM – FB	ParamISM – SWB	ParamISM – SWB
48	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – SWB
64	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
80	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
96	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
128	DiscISM – FB	DiscISM – FB	DiscISM – FB	DiscISM – FB
160	n/a	DiscISM – FB	DiscISM – FB	DiscISM – FB
192	n/a	DiscISM – FB	DiscISM – FB	DiscISM – FB
256	n/a	DiscISM – FB	DiscISM – FB	DiscISM – FB
384	n/a	n/a	DiscISM – FB	DiscISM – FB
512	n/a	n/a	n/a	DiscISM – FB

5.6.2 Discrete ISM coding mode

5.6.2.1 Overview

Figure 5.6-1 is a schematic block diagram of the DiscISM coding mode encoder.

At the coded input, there is buffered for each frame N_{ISM} ISMs, i.e. N_{ISM} audio streams with the associated respective N_{ISM} metadata. In Figure 5.6-1, the audio streams are denoted by a solid line while metadata and parameters in general by a dotted line.

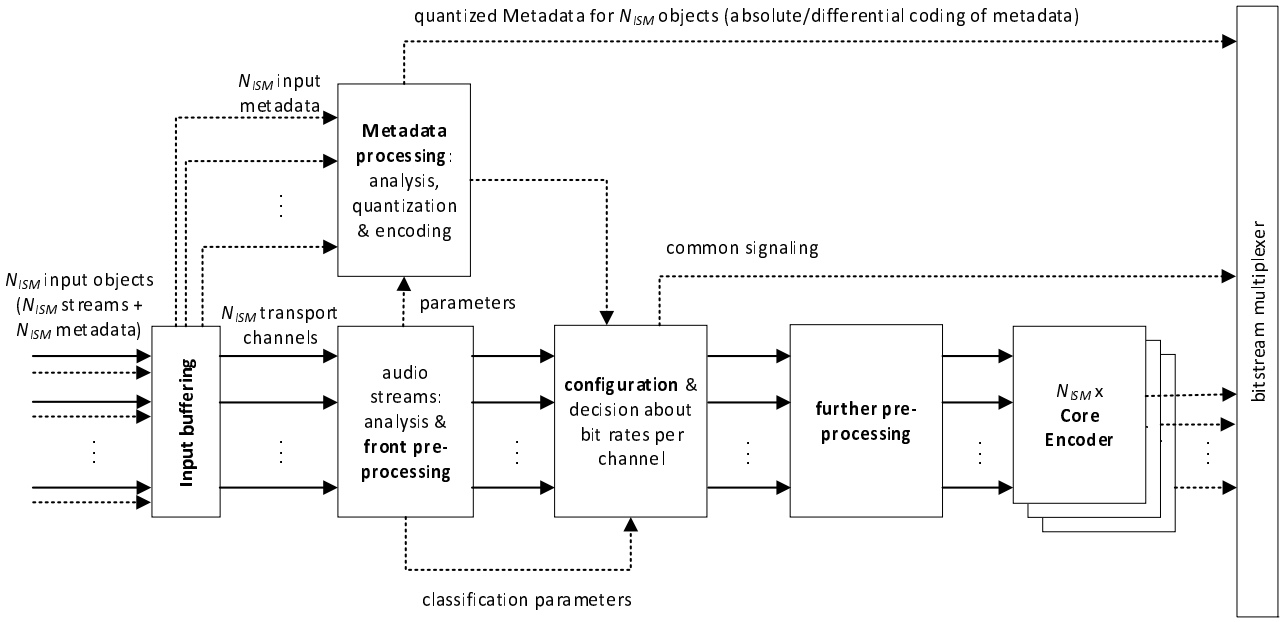


Figure 5.6-1: Block diagram of the DiscISM encoder

5.6.2.2 DiscISM encoding system

The DiscISM encoding system is composed from several functional blocks as shown in Figure 5.6-1.

In the DiscISM mode, the N_{ISM} audio streams represent N_{ISM} transport channels. These transport channels are first analysed and pre-processed in parallel in the front pre-processing (see clause 5.2.2.2) block which further provides information about the audio streams to the metadata processing block while the analysis of audio streams is based on classification of the voice or sound importance of particular streams.

In parallel, the input metadata are analysed, quantized, and encoded in a metadata processing block (see further in clause 5.6.4). The bit-budget used to quantize the metadata then forms an input to the configuration block.

Further, at the configuration block, a decision about bitrates per transport channel is made in a bit-budget allocator. The bit-budget allocator employs a bitrate adaptation algorithm to distribute the available bit-budget for core-encoding the N_{ISM} audio streams in the N_{ISM} transport channels (see further in clause 5.6.2.3). Consequently, a better, more efficient distribution of the available bit-budget between the audio streams is achieved.

Once the configuration and bitrate distribution between the N_{ISM} audio streams is completed by the bit-budget allocator, the pre-processor performs a sequential further pre-processing (see clause 5.2.2.3.1) on each of the N_{ISM} transport channels.

Finally, the N_{ISM} transport channels are sequentially encoded using N_{ISM} fluctuating bitrate core-encoders, specifically by SCE tools (see clause 5.2.3.2), one per transport channel. The bitrate used by each of the N_{ISM} core-encoders is the bitrate selected by the bit-budget allocator for the corresponding audio stream.

5.6.2.3 Bitrates distribution between audio streams

5.6.2.3.1 Bitrate distribution algorithm

The bitrate distribution algorithm comprises the following steps 1-6 performed by the bit-budget allocator.

Step 1: The ISM total bit-budget, $bits_{ISM}$, per frame is calculated from the ISM total bitrate $brate_{ISM}$ (in case of the ISM format it is equal to the IVAS total bitrate $brate_{IVAS}$) using the following relation:

$$bits_{ISM} = \frac{brate_{ISM}}{50} \tag{5.6-1}$$

The denominator, 50, corresponds to the number of frames per second, assuming 20 ms long frames in IVAS.

Step 2: The element bitrate $brate_{element}$ (resulting from a sum of the metadata bitrate and core-coder total bitrate related to one ISM) defined for N_{ISM} streams is constant at a given $brate_{ISM}$, and about the same for the N_{ISM} streams. The corresponding element bit-budget, $bits_{element}$, is computed in the bit-budget allocator for the streams $n = 0, \dots, N_{ISM} - 1$ using the following relation:

$$bits_{element}[n] = \left\lfloor \frac{bits_{ISM}}{N_{ISM}} \right\rfloor \tag{5.6-2}$$

where $\lfloor x \rfloor$ indicates the largest integer smaller than or equal to x. In order to spend all available ISM total bit-budget $bits_{ISM}$ the element bit-budget $bits_{element}$, of the last audio object is eventually adjusted using the following relation:

$$bits_{element}[N_{ISM} - 1] = \left\lfloor \frac{bits_{ISM}}{N_{ISM}} \right\rfloor + bits_{ISM} \bmod N_{ISM} \tag{5.6-3}$$

where "mod" indicates a remainder modulo operation. Finally, the element bit-budget $bits_{element}$ of the N_{ISM} audio objects is used to set the value $element\_brate$ for the ISMs $n = 0, \dots, N_{ISM} - 1$ using the following relation:

$$element\_brate[n] = bits_{element}[n] \cdot 50 \tag{5.6-4}$$

where the number 50 corresponds to the number of frames per second, assuming 20 ms long frames.

Step 3: The metadata bit-budget $bits_{meta}$ per frame, of the N_{ISM} streams is summed, using the following relation:

$$bits_{meta\_all} = \sum_{n=0}^{N_{ISM}-1} bits_{meta}[n] \tag{5.6-5}$$

and the resulting value $bits_{meta\_all}$ is added to an ISM common signalling bit-budget, $bits_{ISM\_signalling}$, resulting in the side bit-budget:

$$bits_{side} = bits_{meta\_all} + bits_{ISM\_signalling} \tag{5.6-6}$$

The metadata bit-budget $bits_{meta}$ is obtained in the metadata processing module as described in clause 5.6.4 while the ISM signalling is described in clause 5.6.5.2.

Step 4: The side bit-budget, $bits_{side}$, per frame, is split in the bit-budget allocator equally between the N_{ISM} streams and used to compute the core-encoder bit-budget, $bits_{CoreCoder}$, for each of the N_{ISM} streams using the following relation:

$$bits_{CoreCoder}[n] = bits_{element}[n] - \left\lfloor \frac{bits_{side}}{N_{ISM}} \right\rfloor \tag{5.6-7}$$

while the core-encoder bit-budget of the last audio stream may eventually be adjusted to spend all the available core-encoding bit-budget using the following relation:

$$bits_{CoreCoder}[N_{ISM} - 1] = bits_{element}[N_{ISM} - 1] - \left\lfloor \frac{bits_{side}}{N_{ISM}} \right\rfloor + bits_{side} \bmod N_{ISM} \tag{5.6-8}$$

The corresponding total bitrate, $brate_{total}$, i.e. the bitrate to code one audio stream in a core-encoder, is then obtained for $n = 0, \dots, N_{ISM} - 1$ using the following relation:

$$brate_{total}[n] = bits_{CoreCoder}[n] \cdot 50 \tag{5.6-9}$$

where the number 50 corresponds to the number of frames per second, giving 20 ms long frames.

Step 5: The total bitrate, $brate_{total}[n]$, in inactive frames may be lowered and set to a constant value in the related audio streams. The so saved bit-budget is then redistributed equally between the audio streams with active content in the frame. Such redistribution of bit-budget is described in the following clause 5.6.2.3.2, Step 5.

Step 6: The total bitrate, $brate_{total}[n]$, in audio streams in active frames is further adjusted between these audio streams based on an ISM importance classification. Such adjustment of bitrate is further described in the following clause 5.6.2.3.2.

When the audio streams are all classified as inactive, the above last two steps 5 and 6 are skipped. Accordingly, the bitrate adaptation algorithm described in the following clause 5.6.2.3.2 is employed when at least one audio stream has an active content.

5.6.2.3.2 Bitrate adaptation based on ISM importance

In the ISM format, there is used a classification of ISM importance based on several core-coder parameters, namely local VAD (VAD_{local}, see clause 5.1.12 in [3]), core-encoder type ($coder\_type$, see clause 5.1.13 in [3]), and core-coder mode ($flag_{tcxonly}$). The ISM importance classification is done as part of the bit-budget allocator for rating the importance of a particular ISM stream based on a metric how critical coding of an audio stream to obtain a given quality of a decoded synthesis is. As a result, four (4) distinct ISM importance classes, $class_{ISM}$, are defined:

- No metadata class, ISM_NO_META: frames without metadata coding, i.e. inactive frames with $VAD_{local} = 0$;
- Low importance class, ISM_LOW_IMP: frames where $coder\_type$ is UNVOICED or INACTIVE;
- Medium importance class, ISM_MEDIUM_IMP: frames where $coder\_type$ is VOICED;
- High importance class ISM_HIGH_IMP: frames where $coder\_type$ is GENERIC.

In case of core-coder employing only the TCX core, i.e. $flag_{tcxonly} = 1$, inactive frames with $VAD_{local} = 0$ are classified as ISM_LOW_IMP and frames with $coder\_type$ being UNVOICED are classified as ISM_HIGH_IMP.

The ISM importance class is then used by the bit-budget allocator, in the bitrate adaptation algorithm (see clause 5.6.2.3, step 6) to assign a higher bit-budget to streams with a higher ISM importance and a lower bit-budget to streams with a lower ISM importance. Thus, for every audio stream n, $n = 0, \ldots, N_{ISM} - 1$, the following bitrate adaptation algorithm is used by the bit-budget allocator:

Step 1: In frames classified as $class_{ISM}$ = ISM_NO_META, a constant low bitrate $B_{VAD0} = 2450$ bps is assigned.

Step 2: In frames classified as $class_{ISM}$ = ISM_LOW_IMP, the core-coder total bitrate, $brate_{total}$, is lowered as:

$$brate'_{total}[n] = \max(\alpha_{low} \cdot brate_{total}[n], B_{low}) \quad (5.6\text{-}10)$$

where the constant $\alpha_{low} = 0.6$. Then the constant B_{low} represents a minimum bitrate threshold supported by the core-coder for a particular configuration, which is dependent on the internal sampling rate of the codec and the coded audio bandwidth.

Step 3: In frames classified as $class_{ISM}$ = ISM_MEDIUM_IMP, the core-coder total bitrate, $total\_brate$, is lowered as

$$brate'_{total}[n] = \max(\alpha_{med} \cdot brate_{total}[n], B_{low}) \quad (5.6\text{-}11)$$

where the constant $\alpha_{med} = 0.8$.

Step 4: In frames classified as $class_{ISM}$ = ISM_HIGH_IMP, no bitrate adaptation is used, i.e.

$$brate'_{total}[n] = brate_{total}[n]. \quad (5.6\text{-}12)$$

Step 5: Finally, the saved bit-budget (a sum of differences between the initial ($brate_{total}$) and new ($brate'_{total}$) total bitrates) is redistributed equally between the streams with active content in the frame. The following bit-budget redistribution logic is used:

First, the saved bit-budget is computed using the following relation:

$$bits_{diff} = \sum_{n=0}^{N_{ISM}-1} \left(bits_{CoreCoder,new}[n] - bits_{CoreCoder}[n] \right) \quad (5.6\text{-}13)$$

where

$$bits_{CoreCoder,new}[n] = \frac{brate'_{total}[n]}{50} \quad . \tag{5.6-14}$$

Next, the saved bit-budget is redistributed equally between the core-coder bit-budgets of the streams with active content in a given frame using the following relation:

$$bits_{CoreCoder,final}[n] = bits_{CoreCoder,new}[n] + \left\lfloor \frac{bits_{diff}}{N_{VAD1}} \right\rfloor \tag{5.6-15}$$

for all n with $VAD_{local} = 1$ where N_{VAD1} is the number of streams with active content. The core-coder bit-budget of the first audio stream with active content is then eventually increased using the following relation:

$$bits_{CoreCoder,final}[n] = bits_{CoreCoder,new}[n] + \left\lfloor \frac{bits_{diff}}{N_{VAD1}} \right\rfloor + bits_{diff} \bmod N_{VAD1} \tag{5.6-16}$$

where n in (5.6-16) corresponds to the first stream with $VAD_{local} = 1$.

The corresponding final core-encoder total bitrate, $brate_{total,final}$, is finally obtained for each audio stream $n,\ n = 0, \ldots, N_{ISM} - 1$, as follows:

$$brate_{total,final}[n] = \quad bits_{CoreCoder,final}[n] \cdot 50. \tag{5.6-17}$$

5.6.3 Parametric ISM coding mode

5.6.3.1 General

In ISM mode, if the total bitrate configured is 24.4 kbps or 32 kbps and if there are three or four input objects/channels, the Parametric ISM (ParamISM) mode is employed (see Table 5.6-1). ParamISM allows for encoding audio objects and their related metadata (if present), where the metadata indicates, among other characteristics, the direction information of each audio object. To reduce data and allow for efficient transmission, ParamISM employs a downmixer for downmixing the audio objects into two transport channels for further encoding. The metadata is used to generate new parameter data that describes two relevant audio objects and is transmitted along with the transport channels. Figure 5.6-2 shows an overview of the ParamISM encoder (band-wise processing not reflected).

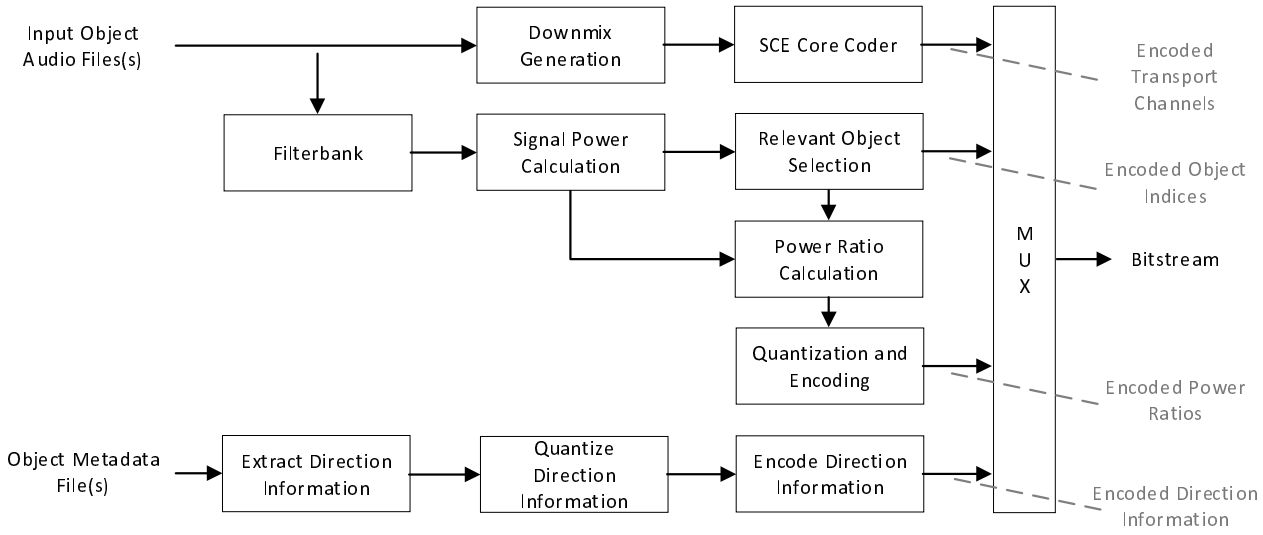


Figure 5.6-2: Parametric ISM encoder overview

5.6.3.2 Parameters

ParamISM identifies the two relevant audio objects in each parameter band, i.e., the two most dominant objects in terms of signal power. To determine the two relevant objects, the audio objects are transformed into a spectral representation by means of an MDFT filterbank as described in clause 5.2.5. Each time frame is divided into 4 subframes, each of which the filterbank is applied to. In the time-frequency representation, each original frame then consists of N-by-K time-frequency (T/F) tiles, where in the temporal direction, 4 time slots, and in the frequency direction, 240, 160, or 80 frequency bins are used, depending on the input sampling rate of 48, 32, or 16 kHz.

The signal power of each object is then calculated and stored for each T/F tile of the current frame:

$$P_i\ (k,n) = |X_i(k,n)|^2, \quad (5.6\text{-}18)$$

where X_i denotes the i-th input object in the spectral domain, k denotes the frequency bin index, and n denotes the time slot index.

To reduce the data load and thus the number of parameters to be transmitted, the T/F files are further grouped into parameter bands. To that end, all 4 time slots are grouped into 1 slot and the K frequency bins are grouped into L bands, where L is either 11 (in case of 48 or 32 kHz input) or 9 (in case of 16 kHz input). Consequently, the signal power of each audio object within each parameter band is calculated by summation over all T/F tiles contained in the parameter band and used as the selection criterion:

$$P_i(l) = \sum_{n=0}^{3} \sum_{k=B(l)}^{B(l+1)} P_i(k,n), \quad (5.6\text{-}19)$$

where l is the parameter band index with $0 \leq l < L$ and $B(l)$ defines the parameter band borders in the frequency direction. $B(l)$ is dependent on the input sampling rate and is given as follows in the case of 48 kHz:

$$B(l) = [0, 4, 8, 12, 16, 24, 32, 44, 60, 84, 124, 240] \quad (5.6\text{-}20)$$

Note: In the case of 32 kHz input sampling rate, $B(11)$ changes from 240 to 160 due to the reduced bandwidth of the input data. In the case of 16 kHz input, the number of bands is reduced to 9 and the borders are given as $B(l) = [0, 4, 8, 12, 16, 24\ 32, 44, 60, 80]$.

Given one value $P_i(l)$ per input object and parameter band, the two objects that comprise the greatest signal power values are then selected as the relevant objects of the considered parameter band and identified by their object indices. The object index of each input object is derived from the input channels fed into the IVAS encoder, i.e., the object occupying the first input channel is identified by an index of 0, the object occupying the second input channel is identified by an index of 1, and so on, up to index 3 in case of the maximum of 4 possible input objects. For each parameter band, the object indices of the two relevant objects are then encoded in the bitstream with 2 bits per index.

The two relevant objects within a parameter band, here represented by their corresponding signal power values $P_1(l)$ and $P_2(l)$, are further characterized by power ratios that signal the contribution of each of the two objects to the sum of the signal powers of both objects:

$$r_1(l) = \frac{P_1(l)}{(P_1(l) + P_2(l))} \quad (5.6\text{-}21)$$

$$r_2(l) = \frac{P_2(l)}{(P_1(l) + P_2(l))} = 1 - r_1(l) \quad (5.6\text{-}22)$$

If both signal power values are 0, the power ratios $r_1(l)$ and $r_2(l)$ are both set to 0.5. Since the relevant objects are always sorted from high-to-lower signal power, the first object index, here denoted with index 1, corresponds to the most dominant object and the corresponding power ratio describes how its signal power compares to that of the second relevant object, here denoted with index 2. As both ratios sum up to 1, only the first power ratio, which always has a value between 0.5 and 1, must be quantized, encoded and transmitted via the bitstream. Specifically, the power ratio associated with the most dominant object is quantized with 3 bits and transmitted as a power ratio index:

$$\lfloor r_{idx}(l) = 2(r_1(l) - 0.5)(2^3 - 1) \rfloor \quad (5.6\text{-}23)$$

As the last part of the ParamISM parametric information to be transmitted, the direction information provided in the input metadata is quantized to a lower resolution. The azimuth values are quantized using 7 bits, while the elevation values are quantized using 6 bits as described in clause 5.6.4.2. The quantized direction information is also used in the generation of the downmix signal as described in clause 5.6.3.4.

5.6.3.3 Noisy Speech

In ParamISM, there is a mechanism that classifies the object input as either noisy speech content or other content. This is done for an entire frame and indicated by a flag $flag\_noisy\_speech$, the calculation of which depends on a noisy speech buffer and another flag $flag\_equal\_energy$ that indicates whether or not all input objects possess roughly the same signal power or energy.

In the context of finding the two relevant objects of a parameter band, the reference power $P_i(l)$ is determined for each object as defined in Eq. (5.6-19) . These values are also made use of to determine $flag\_equal\_energy$. For this, a per-frame signal power value is obtained for each object i by summation over the parameter bands:

$$P_{i,frame} = \sum_{l=0}^{L-1} P_i(l) \quad (5.6\text{-}24)$$

Then, all $P_{i,frame}$ with $i \geq 1$ are compared against $P_{0,frame}$, i.e., the signal power of the first object is compared against the signal power values of all other objects. If it is found that all ratios are in a range

$$0.975 < \frac{P_{0,frame}}{P_{i,frame}} < 1.025, \quad \forall\, i \geq 1 \quad (5.6\text{-}25)$$

then $flag\_equal\_energy$ is set to 1. If the ratio is outside of the specified range for one or more objects, the flag is set to 0.

To determine $flag\_noisy\_speech$, a noisy speech buffer of size 10, all values initalized to 0, is employed. This buffer holds the values of $flag\_noisy\_speech$ of the previous 9 frames plus the current one. If $flag\_equal\_energy$ is 0, $flag\_noisy\_speech$ is also set to 0, a 0 is accordingly written into the noisy speech buffer as the latest entry and the procedure ends.

If $flag\_equal\_energy$ is 1, it is further checked if both core coder flags $flag\_noisy\_speech\_snr$ of the two transport channels are 1. If one or both flags are 0, a 0 is written into the noisy speech buffer. If both flags are 1, it is checked if one or both core coder flags vad_flag (see clause 5.2.2.2) are 1. If true, a 0 is written into the noisy speech buffer, if false, a 1 is written into the noisy speech buffer. Finally, all 10 entries of the noisy speech buffer are evaluated to determine the final value of $flag\_noisy\_speech$: if all 10 entries are 1, then $flag\_noisy\_speech$ = 1, else $flag\_noisy\_speech = 0$. This flag is part of the parametric data transmitted via the bitstream (1 bit per frame) and also part of the SID payload in DTX mode (see table 5.6-2).

5.6.3.4 Downmix

Apart from the set of parameters described above, ParamISM requires a transmission of the actual audio objects. This is done in the form of a downmix, consisting of two transport channels. More concretely, the downmixer is configured to downmix the audio objects in response to the direction information provided in the metadata.

To that end, two virtual microphone signals arranged at the same horizontal position (center) and different orientations are generated. More specifically, two virtual cardioid microphone signals oriented in opposing directions, +90° and -90° with respect to the center line, are employed, forming a left-right pair of virtual microphones.

The positions and orientations of the virtual cardioids are static over all time frames and only the direction information of each object may change from frame to frame according to the associated metadata which gives the direction information for each frame of the corresponding audio object.

For each audio object, the downmixer derives a weighting information for each transport channel using the direction information for the corresponding audio object, thus acquiring the contribution of each object to each transport channel. Prior to obtaining the weights from the virtual cardioids, the direction information provided in the metadata of each object is quantized as described above in the section on the parameter calculation. The left and right cardioids, and thus the weighting information for each object, are defined as:

$$w_{L,i} = 0.5 + 0.5 \cos(\theta_i - \pi/2) \quad (5.6\text{-}26)$$

$$w_{R,i} = 0.5 + 0.5 \cos(\theta_i + \pi/2) = 1 - w_{L,i} \quad (5.6\text{-}27)$$

As elevation is not considered for downmixing, only the azimuth θ_i is used as direction information.

The weights obtained from the virtual cardioids for each time frame and audio object are applied in the time domain in a sample-by-sample manner before combining the weighted samples into the two, left and right, transport channels:

$$DMX_L = \sum_{i \in N_{ISM}} x_i \, w_{L,i} \quad (5.6\text{-}28)$$

$$DMX_R = \sum_{i \in N_{ISM}} x_i \, w_{R,i} \quad (5.6\text{-}29)$$

Here, N_{ISM} denotes the number of input objects and x_i describes the current time frame of the *i*-th object, consisting of, e.g., 960 samples at a sampling rate of 48 kHz. For simplicity, a sample index is omitted.

To avoid sudden directional changes between frames, smoothing is applied to the cardioid signals. This is done starting from the second frame to be processed and makes use of the weighting information of each object of the previous frame, here denoted as $w_{L,i}^{prev}$ and $w_{R,i}^{prev}$, where again $w_{R,i}^{prev} = 1 - w_{L,i}^{prev}$ so that only the weights of one cardioid need to be made available. Smoothing is then realized by calculating

$$w_{L,i}^{smooth} = 0.75\, w_{L,i} + 0.25\, w_{L,i}^{prev} \tag{5.6-30}$$

$$w_{R,i}^{smooth} = 1 - w_{L,i}^{smooth} \tag{5.6-31}$$

and subsequently setting

$$w_{L,i} = w_{L,i}^{smooth} \tag{5.6-32}$$

$$w_{R,i} = w_{R,i}^{smooth} \tag{5.6-33}$$

Furthermore, to ensure a gradual temporal smoothing, a slope is used on the first half of the frame. For this, the following additional weighting is defined:

$$g = \frac{w_{L,i} - w_{L,i}^{prev}}{framesize/2} \tag{5.6-34}$$

The transport channels of the current frame are then obtained by:

$$DMX_{L,i,tmp}(j) = \begin{cases} (w_{L,i}^{prev} + jg)\, x_i(j) & \forall\, j < framesize/2 \\ w_{L,i}\, x_i(j) & \forall\, j \geq framesize/2 \end{cases} \tag{5.6-35}$$

$$DMX_{R,i,tmp}(j) = \begin{cases} (w_{R,i}^{prev} - jg)\, x_i(j) & \forall\, j < framesize/2 \\ w_{R,i}\, x_i(j) & \forall\, j \geq framesize/2 \end{cases} \tag{5.6-36}$$

$$DMX_L(j) = \sum_{i \in N_{ISM}} DMX_{L,i,tmp}(j) \tag{5.6-37}$$

$$DMX_R(j) = \sum_{i \in N_{ISM}} DMX_{R,i,tmp}(j) \tag{5.6-38}$$

Here, j denotes the sample index of the time-domain signals. Finally, the current weighting information of each object $w_{L,i}$ is stored to be used as $w_{L,i}^{prev}$ in the next frame to be processed.

The transport channels and the signal energies contained in each transport channel highly depend on the object positions. As such, the transport channels may contain less energy than the input objects, a fact which may be reflected in the decoded output signal. To mitigate, an energy compensation is employed in the form of a downmix gain that is multiplied in a sample-by-sample manner to the transport channels (sample index omitted):

$$DMX'_L = gain_{DMX}\, DMX_L \tag{5.6-39}$$

$$DMX'_R = gain_{DMX}\, DMX_R \tag{5.6-40}$$

$$gain_{DMX} = \sqrt{\frac{E_{in}}{E_{DMX} + \epsilon}}, \tag{5.6-41}$$

where E_{in} is the combined signal energy of all input objects of the current frame, obtained by

$$E_{in} = \sum_j \sum_{i \in N_{ISM}} x_i^2(j) \tag{5.6-42}$$

and E_{DMX} is the downmix energy, given as

$$E_{DMX} = \sum_j DMX_L{}^2(j) + DMX_R{}^2(j), \tag{5.6-43}$$

with DMX_L and DMX_R the previously calculated transport channels and $\epsilon = 10^{-15}$.

To also avoid any sudden changes between frames, the downmix gain is applied to the transport channels with the same smoothing approach as employed for the cardioid signals. Again, this is done from the second processed frame onwards as it requires access to the gain of the previously processed frame $gain_{DMX}^{prev}$:

$$gain_{DMX}^{smooth} = 0.75\, gain_{DMX} + 0.25\, gain_{DMX}^{prev} \tag{5.6-44}$$

$$g_{nrg} = \frac{gain_{DMX}^{smooth} - gain_{DMX}^{prev}}{framesize/2} \quad (5.6\text{-}45)$$

$$DMX'_L(j) = \begin{cases} (gain_{DMX}^{prev} + j g_{nrg})\, DMX_L(j) & \forall\, j < framesize/2 \\ gain_{DMX}^{smooth}\, DMX_L(j) & \forall\, j \geq framesize/2 \end{cases} \quad (5.6\text{-}46)$$

$$DMX'_R(j) = \begin{cases} (gain_{DMX}^{prev} + j g_{nrg})\, DMX_R(j) & \forall\, j < framesize/2 \\ gain_{DMX}^{smooth}\, DMX_R(j) & \forall\, j \geq framesize/2 \end{cases} \quad (5.6\text{-}47)$$

$DMX'_L(j)$ and $DMX'_R(j)$ are then the final transport channel signals to be coded and transmitted with j the sample index. Finally, $gain_{DMX}^{smooth}$ is stored to be used as $gain_{DMX}^{prev}$ in the next frame to be processed.

5.6.3.5 Encoded Data

In summary, the encoded audio signal of each input audio frame comprises two encoded transport channels that are fed into the core coder as two SCEs, and the parameter data, further consisting of:

- two object identifications for the relevant audio objects of each time-frequency parameter band in the form of object indices 0, 1, 2, or 3
- quantized and encoded direction data (azimuth and elevation) for each audio object in the time frame, with the direction data being constant for all time-frequency parameter bands of the time frame
- one quantized power ratio describing the ratio between the two relevant objects of each time-frequency parameter band
- one flag indicating whether the frame was classified as noisy speech

The bit expenditure of the parametric data in ParamISM mode is summarized in Table 5.6-2. The encoding of the metadata, including the direction data, is shared with the discrete ISM mode.

Table 5.6-2: Bit expenditure of parametric data per frame

Parametric data	Number of bits
Object identifications	$2 \cdot \#\, parameter\ bands \cdot 2$
Power ratios	$\#\, parameter\ bands \cdot 3$
Noisy speech flag	1
Direction data	See clause 5.6.4.2

5.6.4 ISM metadata coding

5.6.4.1 General

In the ISM format, several groups of metadata parameters are analysed, coded, quantized, and transmitted in the bitstream.

The first group of metadata parameters comprises an azimuth and an elevation. If available as the input metadata, they are coded at all bitrates and encoded and quantized as described in clause 5.6.4.2.

The second group of metadata parameters comprises the first group of metadata parameters and extended metadata parameters. The extended metadata parameters comprise a yaw, a pitch, and a radius. The coding of the extended metadata is supported for $brate_{IVAS} \geq 64$ kbps.

Finally, when a non-diegetic audio stream is encoded, a non-diegetic gain is coded and quantized while metadata parameters from first and second group are not coded and transmitted. The coding of non-diegetic gain is supported for $brate_{IVAS} \geq 64$ kbps.

The metadata are analyzed and processed in the metadata processing module from Figure 5.6-1 where the metadata and core-coder parameters of each of the N_{ISM} audio stream are analyzed in order to determine whether the current frame is

inactive ($VAD_{local} = 0$) or active ($VAD_{local} = 1$) with respect to this particular audio stream. In inactive frames, no metadata is coded relative of that ISM. In active frames, the metadata are quantized and coded for this audio stream using a variable bitrateusing the information about the audio streams from the processing module analysis. The metadata processing module further uses a coding logic that controls the use of absolute of differential coding of metadata parameters which limits (a) a range of fluctuation of a metadata coding bit-budget between frames and (b) when a frame is lost, a number of lost metadata parameters from the audio objects coded using relative coding. More details about metadata quantization and coding are provided in the following clauses.

Once the metadata of the N_{ISM} audio streams are analyzed, quantized and encoded, information from the metadata processing module about the bit-budget $bits_{meta}$ for the coding of the metadata per audio stream is supplied to a configuration and decision processor (and more specifically to the bit-budget allocator) described in clause 5.6.2.3. When the configuration and bitrate distribution between the audio streams is completed, the coding continues with further pre-processing. Finally, the N_{ISM} audio streams are encoded using an encoder comprising N_{ISM} fluctuating bitrate SCE core-encoders.

5.6.4.2 Direction metadata encoding and quantization

5.6.4.2.1 General

The metadata processing module of Figure 5.6-1 quantizes and encodes the metadata of the N_{ISM} audio streams sequentially in a loop while a certain dependency is employed between quantization of audio streams and the metadata parameters of these audio streams.

In case of the first group of metadata parameters, two metadata parameters, azimuth and elevation (as included in the N_{ISM} input metadata), are considered and further referred as a directional metadata. The metadata processing module comprises a quantizer of the metadata parameter indexes using the following resolution of metadata parameters in order to reduce the number of bits being used:

a) Azimuth parameter: An azimuth parameter index from the input metadata is quantized to B_{az}-bit index where $B_{az} = 7$. Giving the minimum and maximum azimuth limits (-180 and +180º), a quantization step for a 7-bit uniform scalar quantizer is 2.5º between -140º and 135º and 5º otherwise.

b) Elevation parameter: An elevation parameter index from the input metadata is quantized to B_{el}-bit index where $B_{el} = 6$. Giving the minimum and maximum elevation limits (-90º and +90º), a quantization step for 6-bit uniform scalar quantizer is 2.5º between -70º and 65º and 5º otherwise.

Both azimuth and elevation indexes, once quantized, are coded using either absolute or differential coding. In the case of absolute coding, the current signed value of a parameter is coded. In case of differential coding, a signed difference between a current value and a previous value of a parameter is coded. As the indexes of the azimuth and elevation parameters usually evolve smoothly (i.e. a change in azimuth or elevation position can be considered as continuous and smooth), the differential coding is used by default. However, absolute coding is used in the following instances:

a) There is too large a difference between current and previous values of the parameter index which would result in a higher or equal number of bits for using differential coding compared to using absolute coding (this may happen exceptionally);

b) No metadata were coded and sent in the previous frame;

c) There were too many consecutive frames with differential coding. In order to control decoding in a noisy channel when a frame is lost, the metadata encoder codes the metadata parameter indexes using absolute coding if a number of consecutive frames which are coded using differential coding is higher than $\beta = 10$ frames.

The metadata encoder produces a 1-bit absolute coding flag, $flag_{abs}$, to distinguish between absolute and differential coding.

In the case of absolute coding, the coding flag, $flag_{abs}$, is set to 1, and is followed by the B_{az}-bit (or B_{el}-bit) index coded using absolute coding, where B_{az} and B_{el} refer to the indexes of the azimuth and elevation parameters to be coded, respectively.

In the case of differential coding, the 1-bit coding flag, $flag_{abs}$, is set to 0 and is followed by a 1-bit zero coding flag, $flag_{zero}$, signaling a difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames equal to 0. If the difference Δ is not equal to 0, the metadata encoder continues coding by producing a 1-bit sign flag, $flag_{sign}$, followed by a difference index, of which the number of bits is adaptive, in a form of a unary code

indicative of the value of the difference Δ. Figure 5.6-3 is a diagram showing different scenarios of bitstream coding of one metadata parameter.

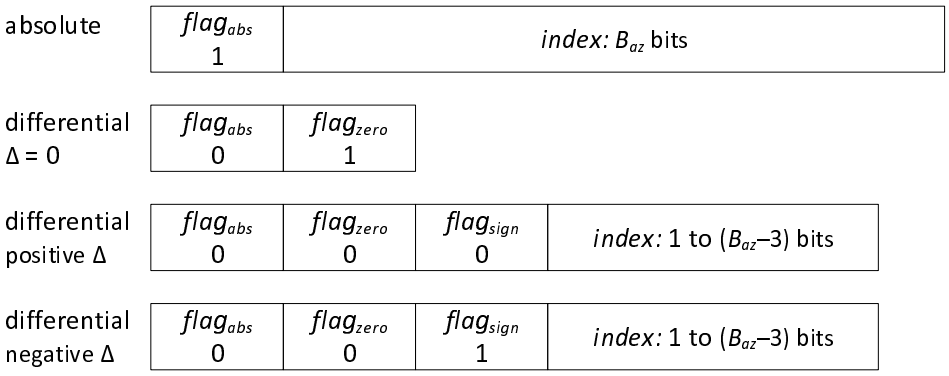


Figure 5.6-3: Different scenarios of bitstream coding of one metadata parameter in ISM format

There are thus in total four (4) different scenarios how to encode a metadata parameter:

Scenario 1: In the case of absolute coding (first line of Figure 5.6-3), the absolute coding flag, $flag_{abs}$, and the B_{az}-bit index (respectively the B_{el}-bit index) are transmitted.

Scenario 2: In the case of differential coding with the difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames equal to 0 (second line of Figure 5.6-3), the absolute coding flag $flag_{abs} = 0$, and the zero coding flag $flag_{zero} = 1$ are transmitted;

Scenario 3: in the case of differential coding with a positive difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames (third line of Figure 5.6-3), the absolute coding flag $flag_{abs} = 0$, the zero coding flag $flag_{zero} = 0$, the sign flag $flag_{sign} = 0$, and the difference index (1 to ($B_{az} - 3$)-bits index (respectively 1 to ($B_{el} - 3$)-bits index)) are transmitted; and

Scenario 4: in the case of differential coding with a negative difference Δ between the B_{az}-bit index (respectively the B_{el}-bit index) in the current and previous frames (last line of Figure 5.6-3), the absolute coding flag $flag_{abs} = 0$, the zero coding flag $flag_{zero} = 0$, the sign flag $flag_{sign} = 1$, and the difference index (1 to ($B_{az} - 3$)-bits index (respectively 1 to ($B_{el} - 3$)-bits index)) are transmitted.

5.6.4.2.2 Intra-object metadata coding logic

The logic used to set absolute or differential coding is further extended by an intra-object metadata coding logic. Specifically, in order to limit a range of metadata coding bit-budget fluctuation between frames and thus to avoid a too low bit-budget left for the core-encoders, the metadata encoder limits absolute coding in a given frame to one, or generally to a number as low as possible of, metadata parameters.

In the case of azimuth and elevation metadata parameter coding, the metadata encoder uses a logic that avoids absolute coding of the elevation index in a given frame if the azimuth index was already coded using absolute coding in the same frame. In other words, the azimuth and elevation parameters of one audio stream are never both coded using absolute coding in the same frame. As a consequence, the absolute coding flag, $flag_{abs,el}$, for the elevation parameter is not transmitted in the audio stream bitstream if the absolute coding flag, $flag_{abs,az}$, for the azimuth parameter is equal to 1, and vice versa.

5.6.4.2.3 Inter-object metadata coding logic

The metadata encoder applies a similar logic from clause 5.6.4.2.2 to metadata coding of different audio streams. The implemented inter-object metadata coding logic minimizes the number of metadata parameters of different audio streams coded using absolute coding in a current frame. This is achieved by the metadata encoder by controlling frame counters of metadata parameters coded using absolute coding chosen from robustness purposes and represented by the parameter $\beta = 10$ frames. In general, a scenario where the metadata parameters of the audio streams evolve slowly and smoothly is considered. In order to control decoding in a noisy channel where indexes are coded using absolute coding every β frames, the azimuth B_{az}-bit index of audio object #1 is coded using absolute coding in frame M, the elevation B_{el}-bit index of audio object #1 is coded using absolute coding in frame $M + 1$, the azimuth B_{az}-bit index of audio

object #2 is encoded using absolute coding in frame $M + 2$, the elevation B_{el}-bit index of audio object #2 is coded using absolute coding in frame $M + 3$, etc.

An advantage of the inter-object metadata coding logic and the intra-object metadata coding logic is to limit a range of fluctuation of the metadata coding bit-budget between frames. Moreover, they increase robustness of the codec in a noisy channel; when a frame is lost, then only a limited number of metadata parameters from the audio objects coded using absolute coding is lost. Consequently, any error propagated from a lost frame affects only a small number of metadata parameters across the audio objects and thus does not affect whole audio scene.

5.6.4.3 Extended metadata encoding and quantization

In addition to the direction metadata (azimuth and elevation) described in clause 5.6.4.2, the metadata is extended for higher bitrate operations ($brate_{IVAS} \geq 64$ kbps) of DiscISM mode to optionally include yaw φ, pitch ψ, and radius r. The default values and the range for these parameters are:

$$\varphi = 0; \quad \in [-180, 180] \tag{5.6-48}$$

$$\psi = 0; \quad \in [-90, 90] \tag{5.6-49}$$

$$r = 1; \quad \in [0, 15.75] \tag{5.6-50}$$

Yaw φ and pitch ψ are used to define the directivity of the object. Yaw represents the horizontal orientation of the source around the z-axis which is perpendicular to the ground plane. Meanwhile pitch represents the vertical orientation around the y-axis. The quantization of the extended metadata (yaw and pitch) follows the same procedure with the direction metadata (azimuth and elevation) which is explained in detail in clause 5.6.4.2.

The radius defines the distance from the origin (0,0) when the listener is also placed at the origin by default. The radius is set to $r = 1$ by default placing the source(s) at the one-unit distance on the unit circle around the listener. The quantization of the radius is uniform with a unit step value of 0.25.

The gain metadata coding operation is the same with the direction metadata coding where absolute or differential coding is used according to the principles described in clause 5.6.4.2.

5.6.4.4 Panning gain in non-diegetic rendering

For non-diegetic audio stream the rendering panning gain is encoded and quantized like an azimuth (see Clause 5.6.4.2), no elevation value is transported. The default values and the range for this parameter is:

$$azimuth = 0; \quad \in [-90, 90] \tag{5.6-51}$$

An azimuth of 90 degrees put the signal on the left ear only, -90 degrees put the signal on the right ear only, 0 degrees put the signal in the centre of the head. The left/right panning gains are computed as fellow:

$$pan_{left} = \left(\frac{azimuth}{90} + 1\right) * 0.5 \tag{5.6-52}$$

$$pan_{right} = 1 - pan_{left} \tag{5.6-53}$$

5.6.5 Bitstream structure

5.6.5.1 Overview

Figure 5.6-4 is a schematic diagram illustrating the structure of the bitstream produced by the bitstream multiplexer from Figure 5.6-1 at the encoder. Regardless whether metadata are present and transmitted or not, the structure of the bitstream is structured as illustrated in Figure 5.6-4.

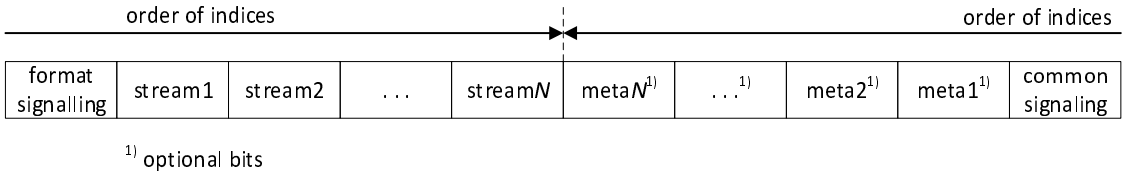


Figure 5.6-4: Structure of ISM format bitstream

5.6.5.2 ISM signalling

The bitstream multiplexer from Figure 5.6-1 first writes the ISM common signalling from the end of the bitstream. The ISM common signalling is produced by the configuration and decision processor from Figure from Figure 5.6-1 and comprises a variable number of bits representing:

a) a number of ISMs, N_{ISM}: the signalling of the number NISM of coded ISMs present in the bitstream is in the form of a unary code with a stop bit (e.g. for $N_{ISM} = 3$ ISMs, the first 3 bits of the ISM common signalling are "110").

b) extended metadata presence flag: written at $brate_{ISM} \geq 64$ kbps, one one-bit flag per frame;

c) non-diegetic object flag: written only when extended metadata presence flag is equal to 1, one one-bit flag per frame;

d) the ISM importance class: comprises two bits per audio stream to indicate the ISM importance class, $class_{ISM}$, (ISM_NO_META, ISM_LOW_IMP, ISM_MEDIUM_IMP, and ISM_HIGH_IMP), as defined in clause 5.6.2.3.2;

(e) a metadata presence flag, $flag_{meta}$: written if metadata are not present or the $class_{ISM}$= ISM_NO_META; the flag comprises one bit per ISM to indicate whether metadata for that particular ISM are present ($flag_{meta} = 1$) or not ($flag_{meta} = 0$) in the bitstream;

f) an ISM VAD flag, $flag_{VAD}$: the ISM VAD one-bit flag is transmitted per ISM when $flag_{meta} = 0$, respectively $class_{ISM}$ = ISM_NO_META, and distinguishes between the following two cases:

1) input metadata are not present so that the audio stream needs to be coded by an active coding mode ($flag_{VAD} = 1$); and

2) input metadata are present and transmitted so that the audio stream can be coded by an inactive coding mode ($flag_{VAD} = 0$).

5.6.5.3 Coded metadata payload

The bitstream multiplexer is next supplied with the coded metadata and writes the metadata payload sequentially from the end of the bitstream for the ISMs for which the metadata are coded in the current frame. The metadata bit-budget for each audio object is not constant but rather inter object and inter frame adaptive. Different metadata format scenarios are shown in Figure 5.6-3.

In the case that metadata are not present or are not transmitted for at least some of the ISMs, the metadata flag is set to 0 for these ISMs. Then, no metadata indices are sent in relation to those audio objects, i.e. $bits_{meta}[n] = 0$.

5.6.5.4 Audio streams payload

Finally, the multiplexer receives the N_{ISM} audio streams coded by the N_{ISM} core encoders through the N_{ISM} transport channels, and writes the audio streams payload sequentially for the N_{ISM} audio streams in chronological order from the beginning of the bitstream (See Figure 5.6-4). The respective bit-budgets of the N_{ISM} audio streams are fluctuating as a result of the bitrate adaptation algorithm described in clause 5.6.2.3.2.

5.6.6 DTX operation

5.6.6.1 Overview

This Clause describes a method for discontinuous transmission (DTX) of audio objects (ISMs) in the ISM format within IVAS. It consists in analysing the audio signals of the audio objects or – for ParamISM mode – the transport channels

for producing signal/voice activity information on the audio signals, detecting, in response to the activity information on the audio streams, a DTX signal segment of the audio objects, generating a SID frame bitstream within the DTX signal segment, wherein the segment and frame detection comprises (a) updating a global SID counter of inactive frames, and (b) signalling the SID frame within the DTX signal segment depending on a value of the global SID counter, and finally encoding the SID frame using SID frame coding of CNG parameters in case no voice/signal activity was detected in any of the object or transport channel sgnals. Each SID frame contains information on the background noise in the form of an FD-CNG mono SID frame, object direction metadata and a coherence value between each object and the dominant object that serves as a control parameter in CNG.

5.6.6.2 ISM DTX operation overview

Similar to the EVS, the ISM DTX coding employs a more efficient DTX coding ("regular") strategy at lower bitrates while a conservative coding strategy is used at higher bitrates. The conservative coding means that the DTX segments correspond to signals with a very low energy. The DTX strategy overview per number of ISMs and per IVAS total bitrate is summarized in Table 5.6-3.

Table 5.6-3: Overview of DTX strategies in ISM format

bitrate [kbps]	number of ISMs			
	1	2	3	4
13.2	regular	n/a	n/a	n/a
16.4	regular	regular	n/a	n/a
24.4	regular	regular	regular	regular
32	conservative	regular	regular	regular
48	conservative	regular	regular	regular
64	conservative	conservative	regular	regular
80	conservative	conservative	regular	regular
96	conservative	conservative	conservative	regular
128	conservative	conservative	conservative	conservative
160	n/a	conservative	conservative	conservative
192	n/a	conservative	conservative	conservative
256	n/a	conservative	conservative	conservative
384	n/a	n/a	conservative	conservative
512	n/a	n/a	n/a	conservative

In order to correctly set the number of transport channels and to enable smooth transitions between active and inactive segments, the CNG spatial parameters depend on the ISM mode (DiscISM, ParamISM). Otherwise, the DTX/SID classification and CNG core-coding are the same regardless the ISM mode.

In the DiscISM mode, the number of core-encoders (SCEs) is usually equal to the number N_{ISM} of streams. In case of ParamISM, the number of core-encoders (SCE) is two (2) which less than the number of audio streams (3 or 4). It is obvious that coding several ISMs in an SID frame would result in N_{ISM} times (2.4 kbps + metadata) bitrate which would further result in a too high IVAS SID bitrate. In order to keep it reasonably low, the IVAS ISM SID bitrate is set to 5.2 kbps similarly as in other IVAS formats and an efficient coding of SID frames is employed.

5.6.6.3 Classification of Inactive Frames in ISM Encoder

5.6.6.3.1 General

As described in clause 5.6.2.2, the audio stream analysis and front pre-processing block classifies the input audio streams and produces the voice/signal activity detection (VAD/SAD) flag f_{SAD}, one per audio stream, while the SAD flag with DTX hangover addition as described in clause 5.1.12.8 of [3] is employed. A change from $f_{SAD} = 1$ to $f_{SAD} = 0$ indicates a start of inactive signal segment for a particular audio signal. It is obvious that the start of an inactive signal segment usually happens at different time instances for different audio streams.

By default, a DTX segment is declared when an inactive signal segment is declared for all audio streams, i.e. when it was determined that none of the input/transport channel signals exhibits voice/signal activity. Then, additional classification stages are used to classify the inactive sound signal segments. Also, the values of metadata have an impact whether a frame is declared as an inactive frame. The classification logic of inactive (SID or NO_DATA) frames or active frames is shown in Figure 5.6-5.

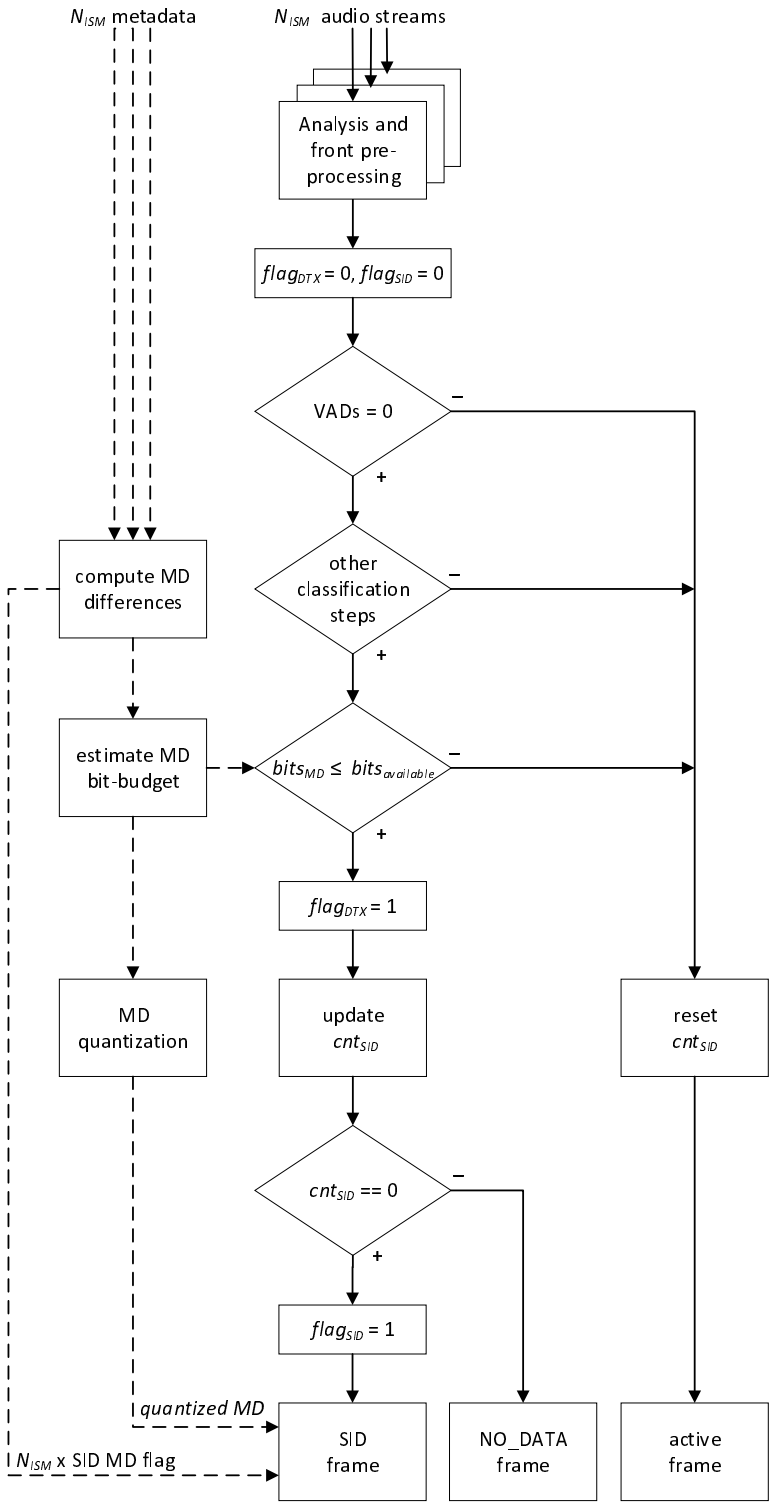


Figure 5.6-5: Flow chart of a SID/DTX controller logic in the DTX ISM format operation

5.6.6.3.2 Global SID Counter

Figure 5.6-5 is a flow chart illustrating a DTX controller used in the DTX operation implemented in the ISM encoder from Figure 5.6-1. Referring to Figure 5.6-5, in order to control the SID/NO_DATA frame decision and the start of a DTX segment, the DTX controller uses a global SID counter, cnt_{SID}. This global SID counter cnt_{SID} allows to control the SID update rate and synchronize the SID/NO_DATA across all audio streams. Consequently, the global SID counter cnt_{SID} enables efficient tuning of a hang-over (hysteresis) DTX logic, or permits other than a default SID update rate of 8 frames or SID adaptive update rate. The global SID counter cnt_{SID} is thus superior to per audio stream SID counters and it effectively ensures that the individual per audio stream SID counters are synchronized.

Referring to Figure 5.6-5, the DTX controller receives the information from the analysis and front pre-processing block, including the SAD information. The DTX controller then initializes to "0" the DTX flag, $flag_{DTX}$, and the SID flag $flag_{SID}$.

By default, the DTX controller detects a DTX signal segment (SID or NO_DATA frame) when the VAD flags, $flag_{VAD}[n]$, $n = 0, \dots, N_{ISM} - 1$, of all audio streams are equal to 0 and signals it by setting the DTX flag to 1, i.e. $flag_{DTX}$. This can be expressed using the following relation:

$$flag_{DTX} = \begin{cases} 1 & \text{if } flag_{VAD} = 0 \quad \text{for all streams} \\ 0 & \text{otherwise} \end{cases} \tag{5.6-54}$$

where $flag_{DTX}$ is the DTX flag. When the VAD flags, $flag_{VAD}[n]$, $n = 0, \dots, N_{ISM} - 1$, of all audio streams are not equal to 0, the DTX controller signals an active frame and selects active frame coding.

Further, the DTX controller signals SID frames within the DTX signal segment using a SID flag, $flag_{SID}$. The SID flag $flag_{SID}$ is set (a) to 1 when the global SID counter cnt_{SID} equals to 0 in which case the DTX controller signals a SID frame and selects SID frame coding and (b) to 0 when the global SID counter cnt_{SID} does not equal to 0 in which case the DTX controller signals a NO_DATA frame and selects NO_DATA frame coding, i.e. not transmitting a bitstream for this frame. This can be expressed using the following relation:

$$flag_{SID} = \begin{cases} 1 & \text{if } cnt_{SID} = 0 \\ 0 & \text{otherwise} \end{cases} . \tag{5.6-55}$$

Next, the DTX controller resets the SID counter cnt_{SID} to -1 in every active frame and it is incremented by 1 at every inactive frame up to the value corresponding to the SID update rate (by default 8 frames). If the value of counter cnt_{SID} reaches the SID update rate, it is set to 0 and the DTX controller signals a SID frame and selects SID frame coding.

The DTX controller can alter the DTX flag $flag_{DTX}$ by using other classification stages based on core-encoder preprocessing values for all audio objects. Specifically, the DTX controller comprises a logic that forces active frame coding ($flag_{DTX} = 0$) and selection of active frame coding in cases when 1) mean value of the LT (Long-Term) background noises over all audio streams, $noise_{mean}$, is higher than a first threshold β_1, or 2) mean value of LT background noises over all audio streams, $noise_{mean}$,, is higher than a second threshold β_2 and LT background noise variations over all audio streams, $noise\_var_{mean}$,, is higher than a third threshold β_3. This can be expressed using the relation:

$$flag_{DTX} = 0 \quad \text{if} \quad noise_{mean} > \beta_1 \text{ or } (noise_{mean} > \beta_2 \text{ and } noise\_var_{mean} > \beta_3) \tag{5.6-56}$$

where the thresholds are set to $\beta_1 = 50$, $\beta_2 = 10$, and $\beta_3 = 2$. The assessment of the LT background noise variations is introduced in order to check for background noise similarities among all streams. The parameter $noise_{mean}$, represents the mean value of the long-term background noise energy values of all audio streams (see Clause 5.1.11 in [3] for details about the background noise energy). The parameter $noise\_var_{mean}$, then represents the variation of the long-term background noise energy values of all audio streams.

5.6.6.4 SID Metadata Analysis, Quantization and Coding

The analysis, quantization and coding of metadata in SID frames follows several principles from active frame coding as described in Clause 5.6.4.2 though there are a few differences. First, in the SID frame, always only two metadata parameters, azimuth and elevation (the object direction information as included in the N_{ISM} input metadata), are quantized and coded. Then, both parameters are coded absolutely meaning that the differential coding is not used in order to avoid potential degradation in long segments of inactive frames due to a lost SID frame in case of a noisy channel if relative coding would be used.

The IVAS SID bitrate is 5.2 kbps from which roughly one half is reserved for a metadata (MD) payload. In order to transmit as much as possible MD values, some compromises are made.

First, one possibility to match the bit-budget constraints is to lower the resolution of MD values. Specifically, when the number of audio objects is low, the available bit-budget for coding the metadata (MD) is relatively generous and the MD resolution is thus kept relatively high. On the other hand, when the number of coded audio objects is high, the resolution of MD values is relatively low and less bits are needed to encode them. There are thus two scenarios used:

a) azimuth and elevation indexes are encoded by means of $B_{az} = 8$ bits and $B_{el} = 7$ bits when one or two audio streams are coded,

b) azimuth and elevation indexes are encoded by means of $B_{az} = 6$ bits and $B_{el} = 5$ bits when three or four audio streams are coded,

while the actual number of coded bits and thus the resolution is explicitly known from the ISM common signalling (see clause 5.6.6.6.2 below).

Second, in order to keep the SID bit-budget as low as possible, a saving in MD bit-budget is achieved by computing a flag, one flag per audio stream, indicating that the MD parameters have not changed (or has not changed significantly) since the last frame and consequently that the metadata (MD) parameters for a specific audio stream are not coded and transmitted. Similarly, this flag serves as an indication that the input MD parameters are not present for that specific audio object.

The DTX controller computes a MD on/off flag, $flag_{MD}[n], n = 0, \dots, N_{ISM} - 1$, for all MD parameter values. Specifically, a flag $flag_{MD,\theta}$, is calculated for the azimuth MD parameter using the following relation:

$$flag_{MD,\theta}[n] = \begin{cases} 1 & \text{if } |\theta[n] - \theta_{last}[n]| > \delta_\theta \\ 0 & \text{otherwise} \end{cases} \quad (5.6\text{-}57)$$

where $\theta[n]$ is the current frame azimuth for stream n, $\theta_{last}[n]$ is the last frame azimuth for stream n and δ_θ is azimuth maximum difference value set to $\delta_\theta = 10$. In the same manner (see Equation (5.6-57)), the DTX controller computes a flag $flag_{MD,\varphi}[n]$ for the elevation MD parameter of stream n while the elevation maximum difference value is set to $\delta_\varphi = 10$. The final MD on/off flag $flag_{MD}[n]$ for one audio stream n is obtained as a logical OR (symbol V in the Equation (5.6-58) below) operation between all particular MD flags:

$$flag_{MD}[n] = flag_{MD,\theta}[n] \ \vee \ flag_{MD,\varphi}[n]\,. \quad (5.6\text{-}58)$$

Then, the flag $flag_{MD}[n]$ represented as a 1-bit information per audio stream, is inserted into the bitstream in the SID frame right after the ISM signalization of the number N_{ISM} of coded audio streams (see clause 5.6.6.6.2 below).

Third, the DTX controller comprises a mechanism that estimates the bit-budget $bits_{MD}$ for metadata quantization. It should be noted that it is only the bit-budget for the quantization of metadata values which is estimated (computed in advance) at this stage while the quantization itself is possibly performed only later. When the estimated bit-budget $bits_{MD}$ is higher than a maximum available bit-budget for the MD coding, $bits_{available}$, the flag $flag_{DTX}$ is reset to 0 (see Figure 5.6-5) and active frame coding is performed. On the other hand, when the estimated MD bit-budget, $bits_{MD}$, is lower or equal to the maximum available bit-budget for the MD coding, $bits_{available}$, the flag $flag_{DTX}$ is not changed, i.e. $flag_{DTX} = 1$, and the DTX coding segment continues. While the maximum available bit-budget for the MD coding, $bits_{available}$, is computed as the difference between the codec SID bit-budget minus the bit-budget needed for other than MD coding (e.g. SID frame signalling, core-coder SID bit-budget, spatial information bit-budget, ISM signalling) in the SID frame.

5.6.6.5 CNG parameters encoding

Along with the quantized object direction metadata, both information on the background noise and control parameters for the CNG needs to be encoded in each SID frame. Due to the tight bitrate limit of 5.2 kbps, information on the background noise is only encoded for one mono signal. A mono signal generator determines which of the object audio signals or transport channels – in case of ParamISM mode – is used by choosing the signal which exhibits the higher long-term energy (the “dominant signal”). The core-coder bitrate of 2.4 kbps is assigned to the respective SCE core-coder and FD-CNG encoding is used to encode the information on the background signal of the chosen audio channel while processing is skipped for other core-coders. The resulting FD-CNG SID for the mono signal is encoded as part of the IVAS SID frame payload. As additional control parameters for the CNG, the coherence between the chosen mono signal and the other object audio signals or transport channels are calculated and encoded in the SID bitstream.

To determine the dominant signal, the long-term energy of all signals is computed as the sum of the energy of the core-coder signal *j* of the last 10 frames (including the current one). The energy of the current frame core-coder input signals is computed as

$$E_{j,0} = \sum_{n=0}^{L_{frame}-1} s_{inp}(n) \quad (5.6\text{-}59)$$

where *j* indicates the input object or transport channel index. $E_{j,0}$ is stored in a buffer which holds the last 10 frame’s energy values and is updated each frame by discarding the oldest value and adding the current frame’s energy value.

The buffer is initialized with zeros at encoder startup. The long-term energy for each signal is then calculated by summing all values in the buffer, according to

$$E_{j,long-term} = \sum_{i=0}^{9} E_{j,i}, \quad (5.6\text{-}60)$$

where $E_{j,1}$ denotes the long-term energy values stored in the buffer with $E_{j,1}$ denoting the energy computed in the last frame, $E_{j,2}$ denoting the energy computed in the second-to-last frame, etc. The index of the core-coder SCE corresponding to the dominant signal is then

$$SCE\_ID = argmax_j\left(E_{j,long-term}\right). \quad (5.6\text{-}61)$$

In case of an SID frame, i.e. when $flag_{SID}$ is 1, the corresponding SCE core-coder is assigned processed with 2.4 kbps and a mono SID information is encoded using FD-CNG as described in clause 5.6.3 of [3] with the modified 1st stage of the MSVQ described in 5.2.2.2.5.3 and added to the payload of the IVAS SID frame (see Figure 5.6-6).

To correctly synthesize a spatial comfort noise in the decoder, coherence values between the dominant signals and each non-dominant signal are estimated and encoded in the IVAS SID payload. They serve as control parameters for the CNG. In case of ParamISM mode, only coherence between the two transport channel signals is used. The coherence between the dominant object signal or transport channel signal and an object signal or transport channel signal with index j is calculated as

$$C_{SCE\_ID,j} = \begin{cases} 1, & if\ j = SCE\_ID \\ \frac{\left|\langle s_{inp,SCE\_ID}, s_{inp,j}\rangle\right|}{\sqrt{\sum_{i=0}^{L_{frame}-1}(s_{inp,SCE\_ID}(i))^2 \cdot \sum_{i=0}^{L_{frame}-1}(s_{inp,j}(i))^2 + \varepsilon}}, & otherwise \end{cases} \quad (5.6\text{-}62)$$

where $\varepsilon = 10^{-15}$. Each inter-object coherence value is limited to the range of [0,1]:

$$C_{SCE\_ID,j} := \min(\max(C_{SCE\_ID,j}, 0), 1). \quad (5.6\text{-}63)$$

Both the *SCE_ID* index and the coherence values are encoded in the IVAS SID frame. The *SCE_ID* index is directly written to the bitstream using 1 or two bits, depending on the number of objects and the ISM coding mode. The coherence values are written sequentially for all objects/transport channels but one starting with the object/transport channel at index 0 and going up to the number of objects/transport channels. No coherence value is written for the dominant object and no coherence value is written if $N_{ISM} = 1$. In case of ParamISM mode (where three or more objects are represented by two transport channel signals) always only one coherence value is written, being the coherence between the two transport channels. Each coherence value is uniformly quantized using four bits, resulting in the respective indices that are written to the bitstream according to

$$I_{C_{SCE\_ID,j}} = \left\lfloor C_{SCE\_ID,j} \cdot (2^4 - 1) + 0.5 \right\rfloor. \quad (5.6\text{-}64)$$

The coherence values, SCE_ID, and the ISM mode represent spatial information which is inserted into the IVAS SID bitstream as shown in Figure 5.6-6.

5.6.6.6 SID bitstream structure

5.6.6.6.1 Overview

Figure 5.6-6 is a schematic diagram illustrating, for a frame, the structure of the SID bitstream produced by the bitstream multiplexer from Figure 5.6-1 and transmitted from the ISM format encoder. Regardless whether metadata are present and transmitted or not, the structure of the SID bitstream is structured as illustrated in Figure 5.6-6.

First the bitstream multiplexer writes the indices of SID format signalling followed by the indices of one core-encoder SID from the beginning of the bitstream while the indices of ISM common signalling, spatial information, and metadata are written from the end of the bitstream.

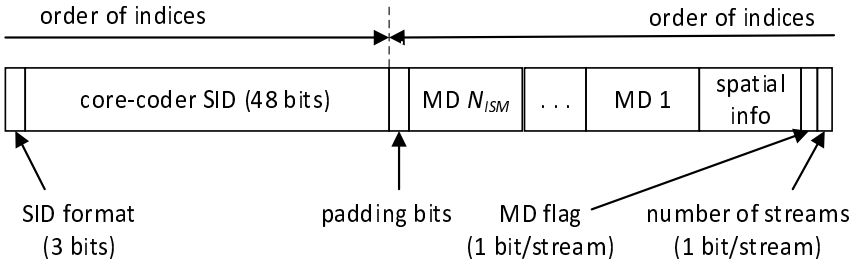


Figure 5.6-6: Structure of SID ISM format bitstream

Further, Table 5.6-4 summarizes the bit-budget of different indexes present in the SID frame depending on the number of transported ISMs.

Table 5.6-4: SID payload in ISM format

	DiscISM mode				ParamISM mode	
parameter	**1 ISM**	**2 ISMs**	**3 ISMs**	**4 ISMs**	**3 ISMs**	**4 ISMs**
SID format	3	3	3	3	3	3
core-coder SID	48	48	48	48	48	48
number of streams	1	2	3	4	3	4
SID metadata flag	1	2	3	4	3	4
ISM mode flag	n/a	n/a	1	1	1	1
noisy speech flag	n/a	n/a	n/a	n/a	1	1
SCE ID	n/a	1	2	2	1	1
coherence	n/a	4	8	12	4	4
azimuth [max]	8	16	18	24	18	24
elevation [max]	7	14	15	20	15	20
padding bits	36	10	3	-14[1)]	7	-6[1)]
total	104	104	104	104	104	104

Note 1): The negative values mean that an active frame coding can be used as a result of the MD quantization logic following the third principle from clause 5.6.6.4.

5.6.6.6.2 ISM Common Signalling in SID Frame

The bitstream multiplexer writes the ISM common signalling from the end of the bitstream. The ISM common signalling in SID frame is produced by the configuration and decision processor (bit-budget allocator) and comprises a variable number of bits representing:

(a) a number N_{ISM} of audio objects: the signaling for the number N_{ISM} of coded audio streams present in the bitstream is in the form of a unary code with a stop bit similarly as in the active frame (for example, for $N_{ISM} = 3$ audio objects, the first 3 bits of the ISM common signaling would be “110” written in a backward order).

(b) a metadata on/off flag $flag_{MD}[n]$, $n = 0, \ldots, N_{ISM} - 1$, one per audio stream, as defined in Equation (5.6-58).

5.6.6.6.3 SID Data Payload

In a SID frame, right after the ISM common signalling, in the backward order, there are written into the bitstream a) spatial information indices, and finally b) metadata values as quantized in clause 5.6.6.4 above.

5.6.6.6.4 SID Audio Streams Payload

In active frames, the multiplexer receives N_{ISM} audio streams coded by N_{ISM} core-coders through N_{ISM} transport channels, and writes them from the beginning of the bitstream right after the IVAS format bits as described in clause 5.6.5.4.

However, in SID frames, the bitstream multiplexer receives one audio stream coded by one of the core-coders through one transport channel, and writes it from the beginning of the bitstream (see Figure 5.6.5) right after the IVAS SID format bits (signalization of the SID mode related to the IVAS format).

5.6.7 ISM bitrate switching

Bitrate switching in ISM format mainly causes reconfiguration of the underlying SCE core-coders as the bitrate might influence selection of the core-coder tools to be used. If a bitrate switch causes the ISM mode to change from DiscISM to ParamISM, or the other way around, more reconfiguration is needed as the number of audio channels – and thus also the number of SCE core-coder instances – changes. Note that this can only happen if the number of objects is 3 or 4, namely when switching from a bitrate lower than 48 kbps to a bitrate of 48 kbps or higher (or the other way around). The performed reconfigurations are:

- Re-setting the core-coder bitrates according to clause 5.6.2.3 step 1 and 2. This results in an initial equal bitrate distribution between the SCEs. The remaining steps are skipped.
- Reconfiguration of the SCE core-coders. This can include deallocation of old SCE core-coders and/or setting up new ones in case the bitrate switch causes the ISM mode to change. For 3 or 4 objects, the number of transport channels can change when switching from a DiscISM mode bitrate to a ParamISM mode bitrate or vice versa.
- Reconfiguration and (de-)allocation of the MDFT structures if the ISM mode changes.

5.7 Multi-channel audio (MC) operation

5.7.1 MC format overview

Coding of multi-channel (MC) inputs is available for the channel layouts 5.1, 7.1, 5.1+2, 5.1+4, and 7.1+4. The coding technique is selected from a set of MC coding modes based on the available bitrate and specified channel layout. The general principle in technique selection is to aim for best possible quality given the allowed bitrate. The MC coding modes thus include multi-channel MASA (McMASA, clause 5.7.3) mode, Parametric multi-channel coding (ParamMC, clause 5.7.4) mode, Parametric multi-channel upmix (ParamUpmix, clause 5.7.5) mode, and discrete multi-channel coding (DiscMC, clause 5.7.6) mode. The use of individual modes is summarized in Table 5.7-1. For all techniques, LFE channel coding is also offered either separately or within the technique.

The multi-channel operation supports output to mono, stereo, multi-channel (at the same or any other layout with up to 16 speakers), Ambisonics (at up to order 3), and binaural. Full bitrate switching is supported. In MC format, DTX is not supported.

Table 5.7-1: Overview of bitrates and coding modes in MC format

Bitrate [kbps]	MC layout				
	5.1	7.1	5.1.2	5.1.4	7.1.4
13.2, 16.4, 24.4, 32	McMASA	McMASA	McMASA	McMASA	McMASA
48, 64, 80	ParamMC	ParamMC	ParamMC	McMASA	McMASA
96	DiscMC	ParamMC	ParamMC	ParamMC	McMASA
128	DiscMC	DiscMC	DiscMC	ParamMC	ParamMC
160	DiscMC	DiscMC	DiscMC	DiscMC	ParamUpmix
192, 256, 384, 512	DiscMC	DiscMC	DiscMC	DiscMC	DiscMC

5.7.2 LFE channel encoding

5.7.2.1 Low-pass filtering

The LFE signal $lfe_{inp}(n)$ of the input audio sampled at 16, 32 or 48 kHz, n being the sample index, is low-pass filtered to suppress undesired high frequency components for subwoofer. The LFE encoder low-pass filter LP_{lfe_enc} is a 4th order Butterworth filter with cut-off frequency of 130 Hz (–3 dB). The transfer function of LP_{lfe_enc} is given by

$$H_{130Hz}(z) = \left(\frac{b_{00}+b_{01}z^{-1}+b_{02}z^{-2}}{1+a_{01}z^{-1}+a_{02}z^{-2}}\right)\left(\frac{b_{10}+b_{11}z^{-1}+b_{12}z^{-2}}{1+a_{11}z^{-1}+a_{12}z^{-2}}\right). \quad (5.7\text{-}1)$$

The coefficients of the LP_{lfe_enc} filter for a given input sampling frequency are given in the table below.

Table 5.7-2: Coefficients of the 130 Hz LP_{lfe_enc} filter

	16 kHz	32 kHz	48 kHz
b_{00}	3.97464794223146e-07f	2.56674586654460e-08f	5.12617881476274e-09f
b_{01}	7.94929601777927e-07f	5.13349181918215e-08f	1.02523584294987e-08f
b_{02}	3.97464788468481e-07f	2.56674582938215e-08f	5.12617879059970e-09f
b_{10}	1.0f	1.0f	1.0f
b_{11}	1.99999996645833f	1.99999996645833f	1.99999984394358f
b_{12}	1.00000001447843f	1.00000001447843f	1.00000000471366f
a_{01}	-1.90746797905194f	-1.95329015717623f	-1.96875982668433f
a_{02}	0.909956295365785f	0.953926661219383f	0.969044914826862f
a_{11}	-1.95913666348268f	-1.98000953138860f	-1.98677297369091f
a_{12}	0.961692382252710f	0.980654742275836f	0.987060670205863f

The input signal, filtered by the $\boldsymbol{LP}_{lfe\_enc}$ filter, is denoted as $LFEin_{LP}(n)$.

5.7.2.2 MDCT based LFE encoding

5.7.2.2.1 Overview

In Parametric upmix encoding mode (see subclause 5.6.4) and MCT (clause 5.6.5), the LFE channel is coded with a MDCT based encoding approach.

5.7.2.2.2 Windowing and MDCT

MDCT based coding of the LFE channel is based on applying a symmetric Kaiser-Bessel derived (KBD) window of 20 ms length followed MCDT transform. This operation is applied in two subframes of the window size, i.e., twice per frame with a stride of 10 ms.

As illustrated in Figure 5.7-1, the left half of the window applied before MDCT comprises a 1 ms zero-pad portion, followed by a 8 ms cross-over portion and a 1 ms portion equalling 1. The right half of the window is obtained through mirroring at the 10 ms point. The size of the cross-over portion of 8 ms causes a corresponding lookahead delay.

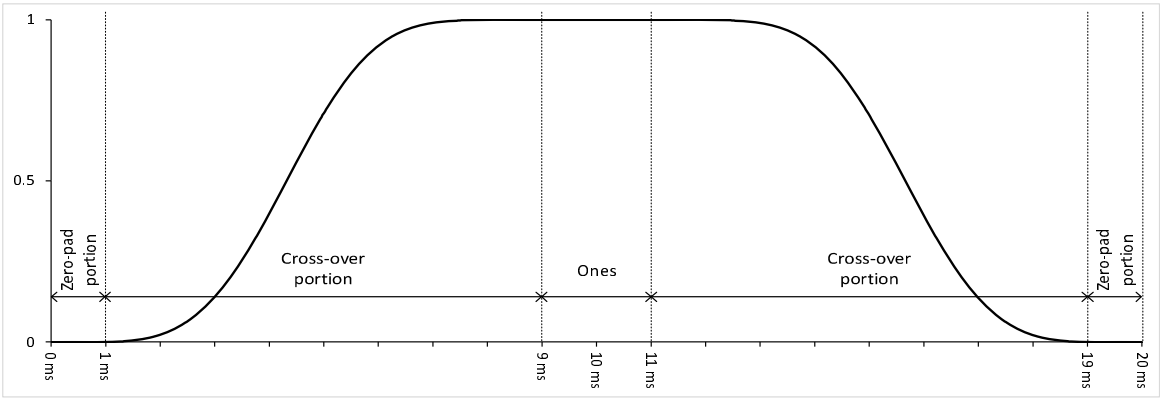


Figure 5.7-1: KBD window used in MDCT based LFE channel codec

In each run of the windowing and MDCT operation the windowed samples $x_w(n)$ are subsequently transformed using an *L*-point MDCT of the following form (except for an MDCT gain scaling factor that is treated separately):

$$X(k) = \sum_{n=0}^{L-1} x_w(n) \cos\left[\frac{2\pi}{L}\left(n + \frac{1}{2} + \frac{L}{4}\right)\left(k + \frac{1}{2}\right)\right], \; k = 0 \ldots \frac{L}{2} - 1 \;, \qquad (5.7\text{-}2)$$

where *L* equals 960, 64 or 320 for an input audio sampling frequency of, respectively, 48000 Hz, 32000 Hz or 16000 Hz.

The MDCT is efficiently computed by means of FFT. This requires the following steps.

Firstly, the windowed samples are folded by subdividing the buffer into 4 portions of length $L/4$ and then adding the sign-inverted and time reversed first portion to the second portion and adding the time-reversed forth portion to the third portion:

$$x_{wf}(n) = x_w\left(n + \frac{L}{4}\right) - x_w\left(-n + \frac{L}{4} - 1\right),\ n = 0 \ldots \frac{L}{4} - 1 \text{ , and} \tag{5.7-3}$$

$$x_{wf}(n) = x_w\left(n + \frac{L}{4}\right) + x_w(-n + L - 1),\ n = \frac{L}{4} \ldots \frac{L}{2} - 1 \text{ .} \tag{5.7-4}$$

Next, the folded samples are twiddled and combined to a sequence of $L/4$ complex samples, as follows:

$$\tilde{x}_{wf}(n) = -x_{wf}(2n)e^{j\frac{2\pi}{L}(n+1/8)} + jx_{wf}\left(\frac{L}{2} - 2n - 1\right)e^{j\frac{2\pi}{L}(n+1/8)} \text{ .} \tag{5.7-5}$$

After these preparations, the buffer $\tilde{x}_{wf}(n)$ of $L/4$ samples is transformed into frequency domain using a complex valued FFT, yielding $\tilde{X}_{wf}(k)$.

Finally, to generate the MDCT output samples, the buffer $\tilde{X}_{wf}(k)$ is firstly scaled with an MDCT gain scaling factor $g = g(L)$ that depends on the applied window length L. This removes gain dependencies on the input audio sampling frequency. After that, the following post-rotation operation is carried out with the same twiddling factors as used prior to the FFT:

$$\begin{aligned} X\left(\frac{L}{2} - 2k - 1\right) &= \Re\left(g\tilde{X}_{wf}(k)e^{j\frac{2\pi}{L}(k+1/8)}\right) \\ X(2k) &= \Im\left(g\tilde{X}_{wf}(k)e^{j\frac{2\pi}{L}(k+1/8)}\right) \end{aligned},\ k = 0 \ldots \frac{L}{4} - 1 \text{ .} \tag{5.7-6}$$

Specifically, the MDCT coefficients for the first and second subframes are respectively denoted as $X_1(k)$ and $X_2(k)$.

5.7.2.2.3 Quantization

The number of MDCT coefficients for each of the two subframes is 480, 320 or 160 for an input audio sampling frequency of, respectively, 48000 Hz, 32000 Hz or 16000 Hz. Each MDCT coefficient represents a frequency band of 50 Hz.

Given that subwoofers typically have a LP filter characteristic with cut-off frequency around 100-120 Hz and that the post-LP filter energy above 400 Hz is typically very low, only MDCT coefficients up to 400 Hz are relevant and need to be coded while the rest of the MDCT coefficients can be ignored and quantized to 0. The total number of MDCT coefficients to quantize and code is therefore equal to $2 \cdot \frac{400}{50} = 16$.

For quantization of these MDCT coefficients, two quantization strategies $\epsilon = \{0,1\}$ are available that may be executed in two successive runs. Initially, a fine quantization strategy ($\epsilon = 0$) is applied followed by a coarse strategy ($\epsilon = 1$) in case the fine strategy is unsuccessful.

To prepare quantization of the MDCT coefficients they are arranged in a buffer $X_i, i = 0 \ldots 15$, with 4 subband groups. Each subband group corresponds to a 100 Hz-wide band with the following bands: 0 – 100 Hz, 100 Hz – 200 Hz, 200 Hz – 300 Hz, 300 Hz – 400 Hz. Accordingly, each subband group comprises 4 MDCT coefficients:

$$X_{0\ldots3} = \text{subband group}_1 = \{X_1(0), X_1(1), X_2(0), X_2(1)\}, \tag{5.7-7}$$

$$X_{4\ldots7} = \text{subband group}_2 = \{X_1(2), X_1(3), X_2(2), X_2(3)\}, \tag{5.7-8}$$

$$X_{8\ldots11} = \text{subband group}_3 = \{X_1(4), X_1(5), X_2(4), X_2(5)\}, \tag{5.7-9}$$

$$X_{12\ldots15} = \text{subband group}_4 = \{X_1(6), X_1(7), X_2(6), X_2(7)\}. \tag{5.7-10}$$

In case of the coarse quantization strategy, the buffer X_i will only contain the 12 MDCT coefficients of subband groups 1-3, i.e., $i = 0 \ldots 11$.

The quantization of the MDCT coefficients relies on shifting them properly to fit them to the quantizer range and to make effective use of the bits available for coding. The shift value s thus represents a gain scaling factor that is encoded with 5 bits within the range of [4, 35] for $\epsilon = 0$ and [0, 31] for $\epsilon = 1$. Furthermore, the quantization of the MDCT coefficients is initially done using mere integer rounding. After this initial rounding stage, uneven allocation of quantization points to the different subband groups is applied. The allocation of quantization points follows the shape of

the LP filter frequency response curve, which retains more information in the lower frequencies than at the higher frequencies and no information outside the cut-off frequency. This has the effects that very low frequencies up to about 130 Hz are very precisely represented while also representing higher frequencies, which avoids or minimizes aliasing effects. The quantization scheme allows representing MDCT coefficients of up to 400 Hz or 300 Hz, depending on quantization strategy.

At the beginning of the quantization procedure, an initial shift value s_0 is determined based on the maximum quantization point $Q_{max,\epsilon}$ available given the quantization strategy ϵ and the sum of absolute values of the buffered MDCT coefficients X_i, according to the following formula:

$$s_0 = \text{floor}(\log_2 \frac{Q_{max,\epsilon}}{\sum |X_i|}) \quad \text{with} \begin{cases} Q_{max,0} = 63 \\ Q_{max,1} = 31 \end{cases} . \tag{5.7-11}$$

The initial shift value is further bounded to the 5-bit encoded shift value range and applied in the below-described quantization scheme unless $\sum |X_i|$ is below a threshold in which case it is forced to the maximum initial shift value of 35. Frames for which the initial shift value equals 35 are treated as silence frames. They are coded according to a specific silence frame coding scheme described below.

The MDCT coefficients of buffer X_i are then quantized according to the following recursion. Firstly, the quantized MDCT coefficients given the current shift value s are obtained by rounding to the nearest integer:

$$X_{q,i} = \text{round}\left(X_i \cdot 2^{s/4}\right) \tag{5.7-12}$$

If any of the absolute values of the quantized MDCT coefficients $X_{q,i}$ exceeds the quantizer overload point $Q_{max,\epsilon}$, the quantization is repeated with decremented s. Otherwise, the quantization recursion ends.

If, at the end of the quantization recursion and when applying the fine quantization strategy, the final shift value is less than the minimum of the 5-bit encoded range, i.e., $s < 4$, then the quantization procedure restarts with coarse quantization strategy (setting ϵ to 1).

Otherwise, the quantization procedure is finalized by the following post quantization operations:

Firstly, the quantized MDCT coefficients are represented by signs and magnitudes. The magnitudes of the negative MDCT coefficients are obtained by taking their absolute value and shifting them by 1 so that their range starts at 0 like for the positive MDCT coefficients.

Secondly, the magnitudes are limited to specific quantization ranges for each subband group, with limits defined according to Table 5.7-3. Note that the ranges start with 0. The table also displays the number of bits required in case of base-2 coding of the MDCT magnitudes.

Table 5.7-3: Maximum available magnitude quantization points and number of bits required for base-2 coding for different subband groups depending on quantization strategy

Quantization Strategy	Subband group$_1$	Subband group$_2$	Subband group$_3$	Subband group$_4$
Fine ($\epsilon = 0$)	63 (6 bits)	31 (5 bits)	7 (3 bits)	1 (1 bit)
Coarse ($\epsilon = 1$)	31 (5 bits)	15 (4 bits)	3 (2 bits)	0 (0 bit)

5.7.2.2.4 Coding

For coding of the magnitudes of the quantized MDCT coefficients two strategies are available, arithmetic coding and base-2 coding. The choice of strategy is based on which of them requires fewer bits.

Firstly, arithmetic coding is tried. To that end, the set of MDCT magnitudes obtained after the limiter operation is arithmetically encoded. Specific arithmetic codes for the different subband groups are used. Each of these arithmetic codes is based on specifically measured probability distributions of quantized MDCT coefficient magnitudes for each given subband group.

Secondly, the number of bits used for arithmetic coding is compared to the required amount for base-2 coding of the magnitudes of the quantized MDCT coefficients. These are 60 bits respectively 44 bits for fine and coarse quantization strategies. In case base-2 coding requires fewer bits, the bits obtained with arithmetic coding are discarded and instead, the magnitudes of the quantized MDCT coefficients are re-encoded using base-2 coding. The selected coding strategy is signaled with one bit. If the bit is set, base-2 coding is used, otherwise arithmetic coding.

Silence frames identified by an initial shift value equal to 35 are treated as a special case. They are coded using a single silence indicator bit set to 1. For any other frames, this bit is set to 0.

The coding of the LFE channel ultimately results in the bit fields illustrated in Table 5.7-4. Note that no other fields than the silence indicator are used if this indicator bit is set. The order of coded MDCT coefficient magnitudes is as in in the above defined buffer X_i.

Table 5.7-4: Bit fields used for coded LFE frames

Silence indicator	Quantization strategy ϵ	Shift value s	MDCT coefficient signs	Coding strategy	MDCT coefficient magnitudes
1 bit	1 bit	5 bits	16 bits (if $\epsilon = 0$) 12 bits (if $\epsilon = 1$)	1 bit	up to 60 bits (if $\epsilon = 0$) up to 44 bits (if $\epsilon = 1$)

5.7.3 Multi-channel MASA (McMASA) coding mode

5.7.3.1 McMASA coding mode overview

Multi-channel MASA (McMASA) encodes multi-channel playback audio signals configured to reproduce a sound scene using MASA spatial audio parameters, LFE parameters, and transport audio signals that are downmix audio signals with a fewer number of channels than in the playback audio signals. The number of transport audio signals in McMASA operation are summarized in table 5.7-5.

Table 5.7-5: Number of transport audio signals in McMASA coding mode per bitrate

Bitrate [kbps]	MC layout				
	5.1	7.1	5.1.2	5.1.4	7.1.4
13.2	1	1	1	1	1
16.4	1	1	1	1	1
24.4	1	1	1	1	1
32	1	1	1	1	1
48	n/a	n/a	n/a	2	2
64	n/a	n/a	n/a	2+1	2+1
80	n/a	n/a	n/a	2+1	2+1
96	n/a	n/a	n/a	n/a	2+1

Figure 5.7-2 presents the McMASA encoder block diagram. The MASA spatial audio parameters comprise directional (or spatial) parameters (azimuth, elevation, and direct-to-total energy ratio) and coherence (or relationship) parameters (spread coherence and surround coherence). The LFE parameters contain a lower frequency effect (LFE) information parameter indicating the proportion of the LFE energy in the multi-channel signals, called the LFE-to-total energy ratio. The LFE parameters are determined for one frequency band using a processed version (a time-frequency transformed, or low-pass filtered version) of the received multi-channel signals. The spatial, coherence and LFE parameters are then encoded along with the downmix signals to be conveyed to the decoder for spatial audio reproduction.

The MASA spatial audio parameters are used together with the transport audio signals for providing spatial audio reproduction in the decoder. The LFE parameters are used together with the transport audio signals for determining the LFE signal in the decoder.

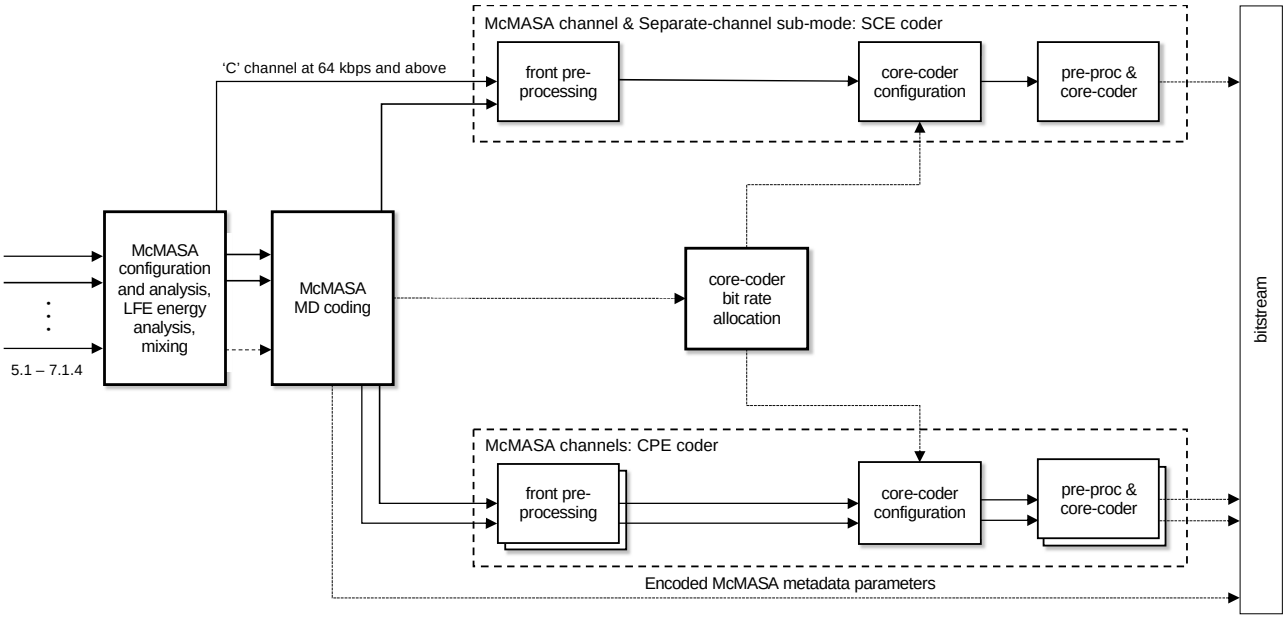


Figure 5.7-2: Block diagram of McMASA encoder

There are two operation sub-modes in McMASA, "normal" and "separate-channel" sub-mode. The sub-mode is selected based on the bitrate. The separate-channel sub-mode is used with bitrates equal to or larger than 64 kbps. The normal sub-mode is used with bitrates smaller than 64 kbps. In case the separate-channel sub-mode was selected, the separate channel index is determined, which equals to the centre channel index, i.e., $i_{sep} = i_c$.

The number of coding frequency bands B_{coding} is determined in the configuration step, see clause 5.5.3.2.4.

As an input to the processing, time-domain multi-channel audio signals $s(n, i)$ are received, (where n is the sample index and i the channel index). There are N input channels ($N = 6 \ldots 12$, depending on the input multi-channel layout: 5.1, 7.1, 5.1+2, 5.1+4, or 7.1+4, see clause 5.7.1). The LFE signal is LP-filtered, as described in clause 5.7.2.1, however all other LFE processing in McMASA operation is described in clause 5.7.3.

LFE energies are determined using the input multi-channel playback audio signals $s(n, i)$. This is explained in detail in clause 5.7.3.2. The outcome is LFE energy $E_{LFE}(m)$ for the first frequency band (where m is the subframe index). In addition, in the separate-channel sub-mode, the total low-frequency energy $E_{LF,tot}(m)$ is obtained (which depicts the energy of all channels of the multi-channel signals in the same frequency band where the LFE energy was determined).

Then, the LFE signal is combined with the centre channel by

$$s'(n, i_c) = s(n, i_c) + s(n, i_{LFE})$$

where i_c is the channel index of the centre channel and i_{LFE} of the LFE channel. Other channels of $s'(n, i)$ correspond to $s(n, i)$.

When operating in the separate-channel sub-mode, the audio signal to be separated from the multi-channel audio signals is first identified based on the i_{sep}. Then the multiple audio signals $s'(n, i)$ are separated, based on the identified audio signal, into a first sub-set of audio signals and a second sub-set of audio signals. The first sub-set contains the identified audio signal, i.e., the identified audio signal is separated to its own signal

$$s_{sep}(n) = s(n, i_{sep})$$

The second sub-set contains the remaining audio signals, i.e., the remaining channels of the multi-channel signals are separated to the multi-channel signal $s''(n, i)$ (i.e., $s''(n, i)$ otherwise corresponds to $s'(n, i)$, but the channel i_{sep} is not present). Thus, in the separate-channel mode, the transport audio signals and the spatial metadata are determined by analysing the remaining audio signals $s''(n, i)$, but not the separated channel audio signal $s_{sep}(n)$. In the normal sub-mode, $s''(n, i)$ corresponds to $s'(n, i)$.

Using the multi-channel playback audio signals $s''(n, i)$, MASA spatial audio parameters are determined by analysing the multi-channel playback audio signals $s''(n, i)$ (i.e., the remaining audio signals, in case of the separate-channel mode). This is explained in detail in clause 5.7.3.3. As the outcome, azimuth $\theta(b, m)$, elevation $\phi(b, m)$, direct-to-total

energy ratio $r_{dir}(b,m)$, spread coherence $\zeta(b,m)$, and surround coherence $\gamma(b,m)$ are obtained. They are determined for each frequency band b and subframe m of the multi-channel playback audio signals (there are 5 frequency bands and 4 subframes). In addition, in the normal sub-mode, the total low-frequency energy $E_{LF,tot}(m)$ is obtained for the first frequency band.

With lower sampling rates (lower than 48 kHz), zeros may be set for the parameter values of the (highest) frequency bands for which the metadata was not analysed.

The LFE-to-total energy ratio $\Xi(m)$ is determined using the LFE energy $E_{LFE}(m)$ and the total low-frequency energy $E_{LF,tot}(m)$

$$\Xi(m) = \frac{E_{LFE}(m)}{E_{LF,tot}(m)}$$

The LFE-to-total energy ratio $\Xi(m)$ is determined for the first frequency band (as were the LFE energy $E_{LFE}(m)$ and the total low-frequency energy $E_{LF,tot}(m)$). In the decoder, rendering based on the LFE-to-total energy ratio (and the transport audio signal(s)) enables the determination of the low frequency effect (LFE) channel.

McMASA transport audio signals, i.e., downmix audio signals, are determined based on the multi-channel playback audio signals $s''(n,i)$ (based on the analysis of the multi-channel playback audio signals $s''(n,i)$, i.e., the remaining audio signals, in case of the separate-channel mode). The number of transport audio signals is therefore smaller than the number of multi-channel playback audio signals. This is described in detail in clause 5.7.3.5. The outcome is the transport audio signal(s) $s_{trans}(n,i)$, having 1 or 2 channels, depending on the bitrate (see Table 5.7-5).

Using the determined the transport audio signals $s_{trans}(n,i)$ and MASA spatial audio parameters (azimuth $\theta(b,m)$, elevation $\phi(b,m)$, direct-to-total energy ratio $r_{dir}(b,m)$, spread coherence $\zeta(b,m)$, and surround coherence $\gamma(b,m)$), the sound scene captured (or in other words, represented) by the original multi-channel playback audio signals can then be reproduced in the decoder. Thus, the MASA spatial metadata is representing the spatial energy distribution in time-frequency tiles. The MASA metadata, the transport audio signals, and the identified separated audio signal (in case of the separate-channel mode) are encoded, as described in the following.

First, it is checked if the coherence parameters are significant enough so that they are to be encoded and transmitted. As a first step, all_coherence_zero parameter is set to one, meaning that no coherence value is significant enough to be transmitted. Then, all spread coherence values $\zeta(b,m)$ are compared against the MASA coherence threshold (having the value of 0.1). If the value for any time-frequency tile (b,m) is larger than the threshold, all_coherence_zero is set to zero. Then, if all_coherence_zero is still having the value of one, the significance of the surround coherence is inspected. This is done using the method described in clause 5.5.3.2.3.5. If surround coherence is determined to be significant, the value of all_coherence_zero is set to zero. Otherwise, it is kept as one.

The determined spatial audio metadata (azimuth $\theta(b,m)$, elevation $\phi(b,m)$, direct-to-total energy ratio $r_{dir}(b,m)$, spread coherence $\zeta(b,m)$, and surround coherence $\gamma(b,m)$) and LFE metadata (LFE-to-total energy ratio $\Xi(m)$) are provided for encoding. This is explained in detail in clause 5.7.3.6.

The determined transport audio signal(s) $s_{trans}(n,i)$ are provided for encoding. In case of separate-channel sub-mode, also the separated channel signal $s_{sep}(n)$ is provided for encoding. This is explained in detail in clause 5.7.3.7.

5.7.3.2 LFE energy computation

5.7.3.2.1 LFE energy computation in normal sub-mode

First, the LFE signal $s(n,i_{LFE})$ is converted to the time-frequency domain using the filter bank described in clauses 5.2.5 and 5.4.3.3. The outcome is the time-frequency domain LFE signal $S(k,m,i_{LFE})$ (where k is the frequency bin index, m the subframe index corresponding also to the slot index, and i_{LFE} is the channel index of the LFE signal). The LFE energy $E_{LFE}(m)$ is then computed for first frequency band (which corresponds to the four first frequency bins)

$$E_{LFE}(m) = \sum_{k=0}^{3} |S(k,m,i_{LFE})|^2$$

which corresponds to a frequency range from 0 to 400 Hz.

5.7.3.2.2 LFE energy computation in separate-channel sub-mode

First, the LFE signal $s(n, i_{LFE})$ and the separate-channel signal $s(n, i_{sep})$ are delayed by $L_{lfe,ana,del}$ samples to provide the correct alignment. $L_{lfe,ana,del}$ corresponds to 9.5 milliseconds, and the value depends on the input sampling rate (e.g., 456 samples at 48 kHz). The result is the delayed versions $s_{del}(n, i_{LFE})$ and $s_{del}(n, i_{sep})$, which are next low-pass filtered.

The low-pass filtering is performed by

$$s_{lp}(n, i) = s_{lp}(n-1, i) + c_{lp} s_{del}(n, i) - c_{lp} s_{del}(n - L_{lp}, i)$$

where L_{lp} corresponds to 5 milliseconds (e.g., 240 samples at 48 kHz), and $c_{lp} = 1/L_{lp}$. The cross-over frequency of this filter is at 120 Hz; thus this signal is associated with the frequency band from 0 to 120 Hz.

The LFE energy $E_{LFE}(m)$ is then computed by

$$E_{LFE}(m) = \sum_{n=n_1(m)}^{n_2(m)} s_{lp}(n, i_{LFE})^2$$

and the total low-frequency energy $E_{LF,tot}(m)$ is computed by

$$E_{LF,tot}(m) = E_{LFE}(m) + \sum_{n=n_1(m)}^{n_2(m)} s_{lp}(n, i_{sep})^2$$

5.7.3.3 McMASA spatial audio parameter estimation

First, the multi-channel playback audio signals $s''(n, i)$ are converted to the time-frequency domain using the filter bank described clauses 5.2.5 and 5.4.3.3. The outcome is the time-frequency domain multi-channel signals $S(k, m, i)$ (where k is the frequency bin index, m the subframe index corresponding also to the slot index, and i is the channel index). It can be presented in a vector form as $\mathbf{s}(k, m)$ (which is a column vector with channels as rows).

There are N_{ana} analysis channels. $N_{ana} = N - 1$ for the normal sub-mode, and $N_{ana} = N - 2$ for the separate-channel sub-mode.

Next, the covariance matrix (which contains inter-channel coherence information) is computed by

$$\mathbf{C_s}(b, m) = \sum_{k=k_1(b)}^{k_2(b)} \mathbf{s}(k, m)\, \mathbf{s}(k, m)^H$$

where $()^H$ denotes the conjugate transpose, and k_1 and k_2 are the first and the last bin of the frequency band b. The size of the covariance matrix is N_{ana} x N_{ana}, and individual entry is referred to by $c_s(b, m, i, j)$, where i and j refer to loudspeaker channels.

The energy of the multi-channel signals is determined by

$$E(b, m) = \sum_{i=0}^{N_{ana}-1} c_s(b, m, i, i)$$

The diagonal entries are real, so the resulting energy values are real as well.

In the normal sub-mode, the total low-frequency energy $E_{LF,tot}(m)$ is computed by

$$E_{LF,tot}(m) = \sum_{i=0}^{N_{ana}-1} \sum_{k=0}^{3} |S(k, m, i)|^2$$

which corresponds to the frequency band from 0 to 400 Hz.

The loudspeaker directions are described by azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles. The multi-channel signals are converted to FOA signals (where the channels 0, 1, 2, and 3 correspond to spherical harmonics W, Y, Z, and X) by

$$S_{FOA}(k,m,0) = \sum_{i=0}^{N_{ana}-1} S(k,m,i)$$

$$S_{FOA}(k,m,1) = \sum_{i=0}^{N_{ana}-1} \sin\theta_{LS}(i)\cos\phi_{LS}(i)\,S(k,m,i)$$

$$S_{FOA}(k,m,2) = \sum_{i=0}^{N_{ana}-1} \sin\phi_{LS}(i)\,S(k,m,i)$$

$$S_{FOA}(k,m,3) = \sum_{i=0}^{N_{ana}-1} \cos\theta_{LS}(i)\cos\phi_{LS}(i)\,S(k,m,i)$$

The loudspeaker setup is converted also to an even distribution version of the original setup, see clause 5.7.3.4 for details. The FOA signals are computed also based on the even setup (the even setup is handled as a horizontal setup)

$$S_{FOAeven}(k,m,0) = \sum_{i=0}^{N_{ana}-1} S(k,m,i)$$

$$S_{FOAeven}(k,m,1) = \sum_{i=0}^{N_{ana}-1} \sin\theta_{LSeven}(i)\,S(k,m,i)$$

$$S_{FOAeven}(k,m,2) = 0$$

$$S_{FOAeven}(k,m,3) = \sum_{i=0}^{N_{ana}-1} \cos\theta_{LSeven}(i)\,S(k,m,i)$$

The azimuth $\theta(b,m)$ and elevation $\phi(b,m)$ angles are determined using the FOA signals $S_{FOA}(k,m,\mathrm{i})$ using the methods described in clause 5.4.3.4. The diffuseness $r'_{diff}(b,m)$ is determined using the FOA signals $S_{FOAeven}(k,m,\mathrm{i})$ using the methods described in clause 5.4.3.4.

The resulting diffuseness values are adjusted in case of non-horizontal loudspeaker layout. First, the temporally averaged intensity is computed in the Z-direction using the FOA signals $S_{FOA}(k,m,\mathrm{i})$, as presented in clause 5.4.3.4 for the normal diffuseness computation. Then, intermediate vertical energy ratio $r'_{vert}(b,m)$ is computed using it and the corresponding energy, as presented in clause 5.4.3.4. These intermediate vertical energy ratios are adjusted by

$$r_{vert}(b,m) = \frac{r'_{vert}(b,m) - 0.15}{1 - 0.15}$$

Then, the corresponding vertical diffuseness parameter is determined by

$$r_{diff,vert}(b,m) = 1 - r_{vert}(b,m)$$

and the resulting values are limited to values between 0 and 1. Then, in case of non-horizontal loudspeaker layout, the diffuseness $r'_{diff}(b,m)$ value is adjusted by

$$r'_{diff}(b,m) \coloneqq \min(r'_{diff}(b,m), r_{diff,vert}(b,m))$$

Next, the coherence parameters are determined between the multi-channel playback audio signals. First, absolute values of the complex coherences in the covariance matrix are computed

$$\mathbf{C}_{\mathbf{s},\mathrm{abs}}(b,m) = |\mathbf{C}_{\mathbf{s}}(b,m)|$$

The energies of the loudspeaker channels can be obtained by

$$E(b,m,i) = c_{s,abs}(b,m,i,i)$$

Next, the channel with the largest energy is selected

$$i_{loudest}(b,m) = \underset{i}{\text{maxindex}}(E(b,m,i))$$

where $\underset{i}{\text{maxindex}}()$ returns the index of the entry that has the largest value (i.e., the index of the channel that has the largest energy).

Normalized coherences between the channels i and j are computed by

$$c_{s,\text{norm}}(b,m,i,j) = \frac{c_{s,\text{abs}}(b,m,i,j)}{\sqrt{E(b,m,i)E(b,m,j)}}$$

The initial surround coherence is computed by taking the minimum value of the normalized coherences between the loudest channel $i_{loudest}(b,m)$ and all other channels and squaring that value

$$\gamma'(b,m) = (\min_{i}(c_{s,\text{norm}}(b,m,i,i_{loudest}(b,m))))^2$$

where $\min_{i}()$ return the minimum value. The resulting surround coherences $\gamma'(b,m)$ are limited to values between 0 and 1 to account for numerical inaccuracies.

Spread coherence computation is performed differently depending on the elevation angle. If elevation $\phi(b,m) \geq 17.5$ degrees, the spread coherence is set to zero $\zeta(b,m) = 0$. Otherwise, i.e., when elevation $\phi(b,m) < 17.5$ the spread coherence is determined as follows.

First, the loudspeaker channel having the azimuth $\theta_{LS}(i)$ closest to the azimuth $\theta(b,m)$ is determined (denoted $i_{closest}(b,m)$).The determination is performed among the loudspeaker channels being on the horizontal plane, i.e., $\phi_{LS}(i) = 0$. In addition, the loudspeaker closest on the left side of $i_{closest}(b,m)$ is determined (denoted $i_{left}(b,m)$), and the loudspeaker closest on the right side of $i_{closest}(b,m)$ is determined (denoted $i_{right}(b,m)$). This determination is also performed among the loudspeaker channels being on the horizontal plane, i.e., $\phi_{LS}(i) = 0$.

Then, stereo coherence is determined by

$$c_{st}(b,m) = c_{s,\text{norm}}\left(b,m,i_{left}(b,m),i_{right}(b,m)\right)$$

and a stereo energy relation parameter $\xi_{lr/lrc}(b,m)$ is determined by

$$\xi_{lr/lrc}(b,m) = \frac{E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right)}{E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right) + E\left(b,m,i_{closest}(b,m)\right)}$$

Using them, a stereoness parameter is determined by

$$\mu(b,m) = c_{st}(b,m)\xi_{lr/lrc}(b,m)$$

Then, front coherence is determined by

$$c_{clr}(b,m) = \min(c_{s,\text{norm}}\left(b,m,i_{closest}(b,m),i_{left}(b,m)\right), c_{s,\text{norm}}\left(b,m,i_{closest}(b,m),i_{right}(b,m)\right))$$

and a front energy relation parameter $\xi_{clr}(b,m)$ is determined by

$$\xi_{clr}(b,m) = \min\left(\frac{E\left(b,m,i_{left}(b,m)\right)}{E\left(b,m,i_{closest}(b,m)\right)}, \frac{E\left(b,m,i_{closest}(b,m)\right)}{E\left(b,m,i_{left}(b,m)\right)}, \frac{E\left(b,m,i_{right}(b,m)\right)}{E\left(b,m,i_{closest}(b,m)\right)}, \frac{E\left(b,m,i_{closest}(b,m)\right)}{E\left(b,m,i_{right}(b,m)\right)}\right)$$

Using them, a coherent panning parameter is determined by

$$\kappa(b,m) = c_{clr}(b,m)\xi_{clr}(b,m)$$

Using these variables, the spread coherence is determined by

$$\zeta(b,m) = \begin{cases} \max(0.5, \mu(b,m) - \kappa(b,m) + 0.5), & if\ \max\left(\mu(b,m),\kappa(b,m)\right) > 0.5\ \&\ \kappa(b,m) > \mu(b,m) \\ \max\left(\mu(b,m),\kappa(b,m)\right), & else \end{cases}$$

The resulting spread coherences $\zeta(b,m)$ are limited to values between 0 and 1 to account for numerical inaccuracies.

Then, stereo energy ratio tuning parameter is determined by

$$r_{st}(b,m) = c_{st}(b,m)\frac{E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right)}{E(b,m)} - \gamma'(b,m)$$

and coherent panning ratio tuning parameter is determined by

$$r_{clr}(b,m) = c_{clr}(b,m)\frac{E\left(b,m,i_{closest}(b,m)\right) + E\left(b,m,i_{left}(b,m)\right) + E\left(b,m,i_{right}(b,m)\right)}{E(b,m)} - \gamma'(b,m)$$

Using them, an initial coherence-based energy ratio tuning parameter is determined by

$$r'_{coh}(b,m) = \max\left(r_{st}(b,m), r_{clr}(b,m)\right)$$

The resulting $r'_{coh}(b,m)$ values are limited to values between 0 and 1 to account for numerical inaccuracies.

The surround coherence values $\gamma(b,m)$ computed for different subframes are combined so that all the subframes within a frame have the same value. This is performed determining the value for the first subframe by

$$\gamma(b,0) = \frac{\sum_{m=0}^{3} E(b,m)\gamma'(b,m)}{\sum_{m=0}^{3} E(b,m)}$$

and setting the same value to all subframes m of the frame, yielding $\gamma(b,m)$.

The diffuseness values $r'_{diff}(b,m)$ computed for different subframes are combined so that all the subframes within a frame have the same value. This is performed determining the value for the first subframe by

$$r_{diff}(b,0) = \frac{\sum_{m=0}^{3} E(b,m)r'_{diff}(b,m)}{\sum_{m=0}^{3} E(b,m)}$$

and setting the same value to all subframes m of the frame, yielding $r_{diff}(b,m)$.

The coherence-based energy ratio tuning parameter values $r'_{coh}(b,m)$ computed for different subframes are combined so that all the subframes within a frame have the same value. This is performed determining the value for the first subframe by

$$r_{coh}(b,0) = \frac{\sum_{m=0}^{3} E(b,m)r'_{coh}(b,m)}{\sum_{m=0}^{3} E(b,m)}$$

and setting the same value to all subframes m of the frame, yielding $r_{coh}(b,m)$.

The direct-to-total energy ratios are determined by

$$r_{dir}(b,m) = \max\left(1 - r_{diff}(b,n), r_{coh}(b,m)\right)$$

5.7.3.4 Even loudspeaker setup determination

The even setup is determined separately for the loudspeaker channels on the horizontal plane, i.e., $\phi_{LS}(i) = 0$ and the loudspeakers not on the horizontal plane, i.e., $\phi_{LS}(i) \neq 0$. The same method is applied for both; hence it is explained below only once. The number of channels fed to the method below is N_{even}.

The spacing between the loudspeakers is first computed by

$$\alpha_{spacing} = \frac{360}{N_{even}}$$

Then, the direction of the first loudspeaker is selected by

$$\alpha_{start} = -\alpha_{spacing}\frac{N_{even}}{2}$$

when N_{even} is odd, and by

$$\alpha_{start} = -\alpha_{spacing}\left(\frac{N_{even}}{2} - 0.5\right)$$

when N_{even} is even.

The even directions as ordered from the smallest azimuth to largest are then determined by

$$\theta_{order}(i') = i'\alpha_{spacing} + \alpha_{start}$$

where i' is the loudspeaker channel index from 0 to $N_{even} - 1$. These angles are ordered to the same order as the input loudspeaker channels were, yielding the even loudspeaker angles $\theta_{LSeven}(i)$. As said, this procedure is repeated separately for the horizontal and the non-horizontal loudspeakers (in case there are non-horizontal loudspeakers).

5.7.3.5 McMASA transport audio signal generation

McMASA transport audio signals are created by downmixing the multi-channel signals $s''(n, i)$. There are N_{ana} input channels to be downmixed. $N_{ana} = N - 1$ for the normal sub-mode, and $N_{ana} = N - 2$ for the separate-channel sub-mode.

Multi-channel energy is computed by

$$E_{mc} = \sum_{i=0}^{N_{ana}-1} \sum_{n=0}^{L_{frame}-1} s''(n, i)^2$$

where L_{frame} is the length of the frame in samples.

Then, prototype downmix signals are created. In case of one transport audio signal ($N_{trans} = 1$), the input channels are summed

$$s_{proto}(n, i') = \sum_{i=0}^{N_{ana}-1} s''(n, i)$$

In case of two transport audio signals ($N_{trans} = 2$), the input channels on the left (i.e., $\theta_{LS}(i) > 0$) are summed to the first prototype downmix signal $s_{proto}(n, 0)$, and the input channels on the right (i.e., $\theta_{LS}(i) < 0$) are summed to the second prototype downmix signal $s_{proto}(n, 1)$. In the normal sub-mode, the centre channel (i.e., $\theta_{LS}(i) = 0$) is summed to both prototype downmix signals after being multiplied by $1/\sqrt{2}$.

The energy of the prototype downmix signals is computed by

$$E_{proto} = \sum_{i'=0}^{N_{trans}-1} \sum_{n=0}^{L_{frame}-1} s''(n, i')^2$$

The energies are smoothed over time by

$$E_{mc,sm} = 0.1E_{mc} + 0.9E_{mc,sm}^{[-1]}$$

$$E_{proto,sm} = 0.1E_{proto} + 0.9E_{proto,sm}^{[-1]}$$

where $E_{mc,sm}^{[-1]}$ and $E_{proto,sm}^{[-1]}$ are the smoothed values computed for the previous frame.

Equalization gain is next determined for keeping the time-smoothed overall energy the same for the transport audio signals as the input multi-channel audio signals. It is determined by

$$g_{EQ} = \frac{E_{mc,sm}}{E_{proto,sm}}$$

Then, an interpolator is determined

$$g_{interp}(n) = \frac{n}{L_{frame}}$$

Using these gains, the transport audio signals are created by equalizing the prototype downmix signals

$$s_{trans}(n,i) = (g_{interp}(n)g_{EQ} + (1 - (g_{interp}(n))g_{EQ}^{[-1]})\, s_{proto}(n,i)$$

where $g_{EQ}^{[-1]}$ is the equalization gain determined for the previous frame.

5.7.3.6 McMASA metadata encoding

5.7.3.6.1 McMASA metadata pre-encoding configuration and processing

As McMASA operation essentially contains MASA metadata, the pre-encoding configuration and processing of McMASA is shared with MASA format. Thus, the process detailed in clause 5.5.3.2 is followed with the following adjustments.

With McMASA, the common metadata config detailed in clause 5.5.3.2.4 uses McMASA specific bit table "mcmasa_bits" for setting the number of "max_metadata_bits". Additionally, coherence coding with flag "useCoherence" is set to start already at the codec bitrate 16.4 kbps. The metadata composition related states are also pre-known and do not need to be detected. That is, "joinedSubframes" is always false, there is always coherence present, and number of directions is 1. As these states are constant, they are not signalled to the decoder either.

For McMASA, the encoding configuration parameters "mc_ls_setup" (as described in clause 5.5.3.2.2) are set based on the channel layout of the input MC-format.

None of the metadata checks and reductions are done at this stage (in contrast to MASA) as the metadata is formed already in the format and TF-resolution intended for encoding at the given bitrate. However, the low-rate coding methods detailed in clause 5.5.3.2.8 and the adjustments in clause 5.5.3.2.11 are performed.

5.7.3.6.2 Spatial metadata encoding

The spatial metadata for McMASA, consisting of the spread and surround coherences, the energy ratios, and the directional information (azimuth and elevation), is encoded in a manner similar to the encoding used for MASA spatial metadata, with the addition of special processing for McMASA as presented in clauses 5.2.4.3.2.2 and 5.2.4.5.1.4. An outline of the overall encoding process for McMASA is presented below:

1. The LFE-to-total energy ratios are encoded and written to the bitstream as presented in clause 5.7.3.6.3
2. McMASA Metadata content is reduced if necessary, according to clause 5.5.3.2.8
3. One bit is used to signal if both surround and spread coherences are present for IVAS total bitrates starting with 16.4 kbps
4. The direct-to-total energy ratios are quantized and encoded according to clause 5.5.3.3.2
5. The surround coherence, if present, is encoded according to clause 5.5.3.3.5
6. The direction dependent metadata is encoded as follows:
    a. Encode spread coherence according to the procedure described in clause 5.5.3.3.4
    b. If multichannel input format is 5.1 or 7.1
        i. Quantize directional metadata (elevation and azimuth) in 2D using the azimuth quantization for McMASA described in clause 5.2.4.3.2.2
    c. Else
        i. Quantize directional metadata in 3D using the azimuth quantization for McMASA described in clause 5.2.4.3.2.2
    d. End

e. Signal using 1 bit if the directional data is planar or not (it could be planar also in 3D case)

f. Encode the azimuth and elevation values using the EC1 method for McMASA encoding as in clause 5.2.4.5.1.4 and count the number of bits that would be used for it, *bits_EC1*

g. If *bis_EC1* < allowed bits

   i. Reset the bitstream

   ii. Try EC2 encoding method with its corresponding quantization resolution and calculate the bits needed for encoding, *bits_EC2*

   iii. If *bits_EC2* < allowed bits

      1. Use EC2 and write its corresponding bits in the bitstream

   iv. Else

      1. Reduce the quantization resolution for spatial metadata

      2. Use EC3 with its corresponding azimuth encoding for the McMASA case

      3. Write EC3 bits in the bitstream

   v. End

h. Else

   i. Use EC1

i. End if

5.7.3.6.3 LFE-to-total energy ratio encoding

For the operating bitrates of 13.2 and 16.4 kbps, the LFE-to-total energy ratios $\Xi(m)$ for each subframe of the current frame are quantized as a single ratio for the whole frame. For higher bitrates, an additional ratio is quantized for each subframe using residual VQ when the LFE frames have a higher energy. Inactive LFE frames for all bitrates are indicated with one bit (0) in the bitstream, and no further LFE-to-total ratio energy coding is performed. The bit allocation, dependent on the McMASA bitrate, is provided in table 5.7-6.

Table 5.7-6: Bit allocation for LFE-to-total energy ratio

Bitrate (kbps)	Bits used (inactive frames)	Bit allocation (active frames)	Bits used (active frames)
13.2	1	1 (activity / energy ratio modulation)	1
16.4	1	1 (activity) + 3 (scalar quantization)	4
≥ 24.4	1	1 + 3 (scalar quant) + 0…4 (subframe VQ for high energy ratio frames)	4…8

The activity of LFE is detected from the subframe $\Xi(m)$ LFE-to-total energy ratios. LFE-to-total energy ratio $\Xi(m)$ is only sent when any of the subframes in the frame have a $\Xi(m)$ which is above a threshold of 0.005. Otherwise, if the maximum $\Xi(m)$ of all the subframes in the frame is less than the threshold, one bit is sent with a zero (0) index for all bitrates.

In the case when a $\Xi(m)$ is above the threshold for any of the subframes in the current frame, the encoding process comprises determining an averaged $\log_2$ LFE-to-total energy ratio for a frame by

$$\Xi_{log_2} = \sum_{m=0}^{3} 0.25 * \Xi_{log_2}(m)$$

where the $\log_2$ LFE-to-total energy ratio for each subframe $\Xi_{log_2}(m)$ is clamped between [-9,1] by

$$\Xi_{log_2}(m) = \begin{cases} -9, if\ \log_2(\Xi(m)) < -9 \\ \log_2(\Xi(m)) \\ 1, if\ \log_2(\Xi(m)) > 1 \end{cases}$$

At the lowest bitrate of 13.2 kbps only one bit is allocated for the LFE-to-total energy ratios for the frame. In this case the LFE-to-total energy ratio bit is set to (1) if Ξ_{log_2} is higher than both a threshold value (MCMASA_LFE_1BIT_THRES=0.03) and a value depending on the previous frame's quantized value $\hat{\Xi}_{prev}$. The comparisons are made in the linear domain by converting the averaged $\log_2$ LFE-to-total energy ratio according to $\Xi = 2^{\Xi_{log_2}}$. Therefore, the condition for setting the LFE-to-total energy ratio bit to 1, is expressed as when $\Xi > 0.03$ and $\Xi > [0.5 * (0.09 + \hat{\Xi}_{prev}) + 0.5 * (0.67 * \hat{\Xi}_{prev}]$ are true the LFE-to-total energy ratio bit is set to (1). If either of conditions are not met LFE-to-total energy ratio bit is set to (0). In practice on a frame-by-frame basis, when the LFE-to-total energy ratio bit is one this signals an increase in the LFE-to-total energy ratio $\Xi(m)$ by a predetermined value and when the LFE-to-total energy ratio bit is zero this signals the dampening of the LFE-to-total energy ratio $\Xi(m)$ by a factor.

At the second lowest bitrate of 16.4 kbps 1 (activity) + 3 = 4 bits are used to encode the LFE-to-total energy ratios for the frame when the frame is classified as an active frame, with the first bit being used to signal that the frame is an active frame. When the first bit is 1 (active) next three bits are used to scalar quantize $\hat{\Xi}_{log_2}$ for the frame using a linear scalar quantizer using a codebook of $[-6.5, \ldots, 0.5]$ with step of 1.0. This codebook is given in Table 5.7-7 as the second column.

Table 5.7-7: Scalar quantizer codebook used by LFE-to-total energy ratio for bitrates ≥ 16.4 kbps and bit allocation for residual subframe energy ratio VQ encoding for bitrates ≥ 24.4 kbps.

Scalar codebook index	Scalar $\hat{\Xi}_{log_2}$ quantized value	Bits used for residual subframe VQ for bitrates ≥ 24.4 kbps
0	-6.5	0
1	-5.5	0
2	-4.5	1
3	-3.5	2
4	-2.5	3
5	-1.5	4
6	-0.5	4
7	0.5	4

For bitrates 24.4 kbps and above, an additional subframe residual $\tilde{\Xi}_{log_2}(m)$ is calculated for each subframe m. This can be obtained by

$$\tilde{\Xi}_{log_2}(m) = \hat{\Xi}_{log_2} - \Xi_{log_2}(m)$$

The subframe residuals $\tilde{\Xi}_{log_2}(m)$ for the subframes of the frame are then encoded as a 4-dimensional vector using a vector quantizer with a 16-entry trained codebook. The entries of the trained codebook are given by the array `McMASA_LFEGain_vectors[4x16].`

The residual is adaptively encoded depending on the scalar codebook index used to quantize the $\hat{\Xi}_{log_2}$. For low energy ratio frames (i.e., when $\hat{\Xi}_{log_2} \leq -5.5$), which are associated with the scalar codebook indices 0 and 1, only the $\hat{\Xi}_{log_2}$ value is quantized with the scalar quantizer and transmitted. For the next higher energy ratios, corresponding to scalar codebook indices (2-4), between 1 to 3 bits are used to quantise and encode the residual subframes. Finally, for the highest energy ratios, corresponding to scalar codebook indices (5-7), the maximum of 4 bits are used to quantise and encode the subframe residuals, which corresponds to the case when $\hat{\Xi}_{log_2} \geq -1.5$. Table 5.7-7 (third column) provides LFE-to-total energy ratio subframe VQ bit allocation.

5.7.3.7 McMASA transport audio signal encoding

The McMASA encoder encodes transport audio signal(s) in addition to the metadata, as shown in figure 5.7-2. The McMASA transport audio signals $s_{trans}(n, i)$ are encoded differently depending on the bitrate-related configuration and generation of transport channels. For bitrates 13.2 kbps to 32 kbps, a single downmixed transport audio signal is

encoded using a single channel element (SCE) encoder. For the bitrate 48 kbps, two downmixed transport audio signals are encoded using a channel pair element (CPE) encoder.

For bitrates 64 kbps to 96 kbps, a separate-channel mode is used, and the centre channel is separated from MASA spatial audio representation to a separated channel signal $s_{sep}(n)$ and encoded separately. In this case, two downmixed transport channels are encoded using a CPE encoder and the separated channel is encoded using a SCE encoder. The bitrate for the SCE and CPE streams is divided such that the SCE stream receives 25% of the total bitrate on bitrates above 64 kbps and 30% of the total bitrate on the bitrate 64 kbps. The rest of the bitrate is given to the MASA metadata and CPE encoder.

5.7.3.8 Bitstream structure

The high-level bitstream structure for McMASA multi-channel mode is generally similar as bitstream structure used for MASA input operation (see clause 5.5.6). However, in case of McMASA, the number of transport audio signals need not be provided in the bitstream. Another difference is that McMASA has two distinct sub-modes: so-called normal sub-mode and separate-channel sub-mode. As seen in table 5.7-5, the separate-channel sub-mode encodes one additional channel.

McMASA bitstream structure in normal sub-mode is as follows:

- IVAS format information bits
- Audio transport channel bits
- McMASA metadata bits

Additionally, in separate-channel sub-mode:

- SCE bits (for separated channel)

5.7.4 Parametric MC coding mode

5.7.4.1 ParamMC coding mode overview

The Parametric Multi-channel coding mode encoder generates a transport audio signal with at least two channels from the original multi-channel input signal. It estimates channel level and inter channel correlation parameters from the input audio signal. The transport audio signal channels are coded using either MDCT Stereo (for 2 transport audio channels) or MCT (for 3 transport audio channels) and are written to the bitstream. The estimated parameters are encoded as side information. The channel levels are encoded as inter-channel level differences ICLD for all channels and as the inter-channel correlations ICCs as a selected subset of all possible combinations of the original channels.

The ICLDs and ICCs are estimated and coding for a number of parameter bands in the frequency domain. To lower the side information load the parameter bands are divided into two sets that are sent alternately for each frame resulting in one set containing the channel level and correlation information for two consecutive frames when the signal has non-transient behaviour. In case of a transient signal parameter bands are combined and the reduces set of parameters is sent for the full band width in a frame with a transient, in this case the parameters are containing the channel level and correlation information specific to this frame. Figure 5.7-3 shows the signal flow in the ParamMC encoder.

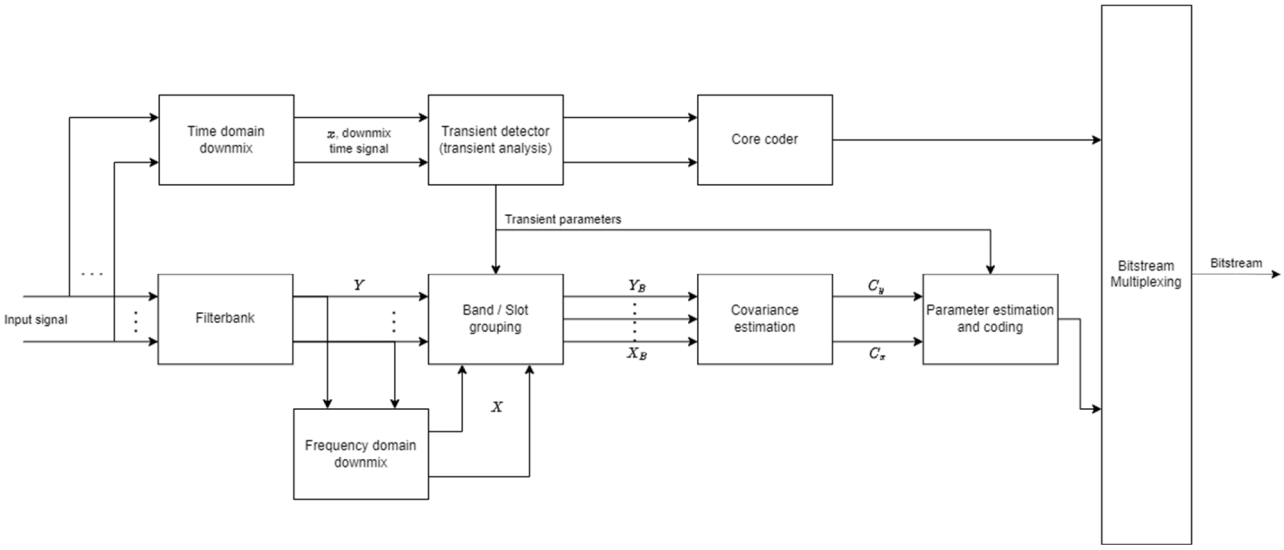


Figure 5.7-3: ParamMC Encoder

5.7.4.2 ParamMC parameter frame index initialization and update

The initial state of the parameter frame index i_F is set to zero $i_F = 0$. The parameter frame index can have the values $\{0,1\}$. Before processing the current frame, the parameter frame index is updated to the other value than currently stored.

5.7.4.3 ParamMC transport channel and parameter band configuration

Depending on the bit rate, and the input channel configuration the number of transport audio signals n_t is chosen from Table 5.7-8, and the parameter band grouping table from Table 5.7-16 with Tables 5.7-9, 5.7-11, and 5.7-13 containing the parameter band grouping for different parameter band numbers. The parameter band mapping i_f is chosen from one of the Tables 5.7-10, 5.7-12, and 5.7-14, depending on the chose parameter band grouping. The number of sent parameter bands n_B is chosen from Table 5.7-15 according to the chosen parameter band grouping and the band width of the input audio signal.

Table 5.7-8: Number of transport audio signals $n_t\ used$ in the ParamMC coding mode

Bitrate [kbps]	MC layout				
	5.1	7.1	5.1.2	5.1.4	7.1.4
48	2	2	2	n/a	n/a
64	2	2	2	n/a	n/a
80	2	2	2	n/a	n/a
96	n/a	3	3	3	n/a
128	n/a	n/a	n/a	3	3

Table 5.7-9: Band grouping table for 20 parameter bands $pmc_{b,20}$

	Parameter band b_p																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
b_{start}	0	1	2	3	4	5	6	7	8	9	10	12	14	17	20	23	27	33	40	52
b_{end}	0	1	2	3	4	5	6	7	8	9	11	13	16	19	22	26	32	39	51	59

Table 5.7-10: Parameter band mapping for 20 parameter bands

	Parameter band b_p																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
i_F	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Table 5.7-11: Band grouping table for 14 parameter bands $pmc_{b,14}$

	Parameter band b_p													
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
b_{start}	0	1	2	3	4	5	6	8	10	13	16	20	28	40
b_{end}	0	1	2	3	4	5	7	9	12	15	19	27	39	59

Table 5.7-12: Parameter band mapping for 14 parameter bands

	Parameter band b_p													
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
i_F	1	1	1	1	1	1	1	0	0	0	0	0	0	0

Table 5.7-13: Band grouping table for 10 parameter bands $pmc_{b,10}$

	Parameter band b_p									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
b_{start}	0	1	2	3	5	7	10	14	20	40
b_{end}	0	1	2	4	6	9	13	19	39	59

Table 5.7-14: Parameter band mapping for 10 parameter bands

	Parameter band b_p									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
i_F	1	1	1	1	1	0	0	0	0	0

Table 5.7-15: Table of sent parameter bands n_B depending on the input band width

Band grouping table	Band width			
	NB	**WB**	**SWB**	**FB**
$pmc_{b,20}$	10	14	18	20
$pmc_{b,14}$	8	11	13	14
$pmc_{b,10}$	6	8	9	10

Table 5.7-16: Overview the band grouping tables

Bitrate [kbps]	MC layout				
	5.1	**7.1**	**5.1.2**	**5.1.4**	**7.1.4**
48	$pmc_{b,10}$	$pmc_{b,10}$	$pmc_{b,10}$	n/a	n/a
64	$pmc_{b,14}$	$pmc_{b,14}$	$pmc_{b,14}$	n/a	n/a
80	$pmc_{b,14}$	$pmc_{b,14}$	$pmc_{b,14}$	n/a	n/a
96	n/a	$pmc_{b,20}$	$pmc_{b,20}$	$pmc_{b,14}$	n/a
128	n/a	n/a	n/a	$pmc_{b,20}$	$pmc_{b,14}$

5.7.4.4 ParamMC transport audio signal generation

Depending on the input channel configuration and the number of transport audio signals n_t the downmix matrix $\mathbf{M}_{DMX}$ is chosen from Table 5.7-17. The transport audio signal is generated by a time-domain sample-wise down mix of the input channels using the static downmix matrix:

$$s_{t,k}(n) = \sum_{l=0}^{n_i-1} s_{i,l}(n)\, m_{DMX\,lk} \qquad (5.7\text{-}13)$$

where $s_{t,k}$ is the kth transport audio signal channel, with $k = 0,1 \ldots n_t - 1$, $s_{i,l}$ is the lth input audio channel, with $l = 0,1, \ldots n_i - 1$, and n is the sample index with $n = 0,1, \ldots, framelength - 1$.

Table 5.7-17: Downmix matrix $\mathbf{M}_{DMX}$

MC layout	n_t	$\mathbf{M}_{DMX}$
5.1	2	$\begin{bmatrix} 1 & 0 & 1/\sqrt{2} & 1/\sqrt{2} & 1 & 0 \\ 0 & 1 & 1/\sqrt{2} & 1/\sqrt{2} & 0 & 1 \end{bmatrix}^{\mathrm{T}}$
7.1, 5.1.2	2	$\begin{bmatrix} 1 & 0 & 1/\sqrt{2} & 1/\sqrt{2} & 1 & 0 & 1 & 0 \\ 0 & 1 & 1/\sqrt{2} & 1/\sqrt{2} & 0 & 1 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$
	3	$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$
5.1.4	3	$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$
7.1.4	3	$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$

5.7.4.5 ParamMC time domain transient detection

On the generated time domain transport audio signals of the current frame, a transient analysis is done to determine the occurrence of a transient within the frame.

The Time-domain transient detection (clause 5.2.2.2.7) is run for each transport channel. First for the two last subblocks of the previous frame it is checked if the energy of the segment A transient is detected if the energy of a segment $ETD(i)$, i=-2..7 exceeds the accumulated energy by a constant factor of 8.5 and the transient position index is set to max(i,*0)* and the transient indicator for this channel is set to 1. The overall transient flag is set to one if at least one transport channel has the channel transient indicator set to 1. The overall transient position index is set to the smallest one of all channels having the transient indicator set to 1.

If no transient was detected, the transient flag is set to zero $f_T = 0$ and the overall transient position index is set to zero $i_T = 0$.

5.7.4.6 ParamMC analysis windowing and MDFT

Every frame of the multi-channel input signal $s_i(n)$ is divided into 8 time slots and each time slot is windowed with a sinusoidal window. Figure 5.7-4 shows the time slot windowing and MDFT. The input to MDFT is a 5 ms long time domain signal $s_{i,m}^{PMC_{ana},mdft}(n)$, where m is the subframe index with $0 \leq m < 8$ and n is the sample index with $0 \leq n < \frac{10\ sampling frequency}{2000}$, that is formed by windowing a time slot of s_i(n) and then applying the sin window function as shown in Equations (5.7-14), (5.7-15), and (5.7-16).

$$s_{i,m}^{PMC_{ana},mdft}(n) = s'^{PMC_{ana}}_{i,m}(n) Win^{PMC_{ana}}(n), \tag{5.7-14}$$

$$s'^{PMC_{ana}}_{i,m}(n) = \begin{cases} 0, & 1 \leq n \leq (L_{sf} - L_{fade}) \\ s_i(mL_{sf} - L_{prev} + n), & (L_{sf} - L_{fade}) \leq n < (2L_{sf}) \end{cases} \tag{5.7-15}$$

$$Win^{PMC_{ana}}(n) = \begin{cases} 0, & 0 \le n < (L_{sf} - L_{fade}) \\ \sin^2\left(\frac{n-(L_{sf}-L_{fade})}{L_{fade}} - \frac{1}{2L_{fade}}\right), & (L_{sf} - L_{fade}) \le n < (L_{sf}) \\ 1, & (L_{sf} - L_{prev} + L_{offset} + L_{fade}) < n \le (2L_{sf} - L_{fade}) \\ 1 - \sin^2\left(\frac{n-(2L_{sf}-L_{fade})}{L_{fade}} - \frac{1}{2L_{fade}}\right), & (2L_{sf} - L_{fade}) \le n < 2L_{sf} \end{cases} \quad (5.7\text{-}16)$$

where $L_{frame} = 20L_{ms}$, $L_{sf} = 2.5L_{ms}$, $L_{prev} = 8.5L_{ms}$, $L_{fade} = L_{ms}$, $L_{ms} = \frac{sampling frequency}{1000}$, m is the subframe index with $0 \le m < 8$ and $s_i(n)$ where $n < 0$ are samples from the previous frame.

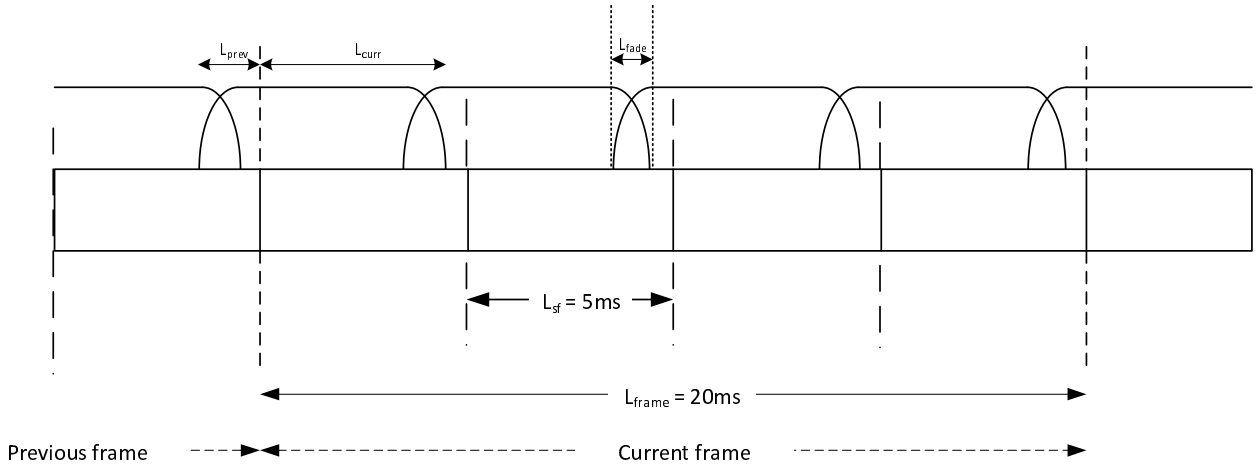


Figure 5.7-4: Subframe windowing and MDFT

For each m, $s_{i,m}^{PMC_{ana},mdft}(n)$ is converted to frequency domain signal $Y_{k,s}(b)$ by applying the MDFT (as described in subclause 5.2.5.2) to $s_{i,m}^{PMC_{ana},mdft}(n)$, where m is the subframe index and b is the frequency bin index.

5.7.4.7 ParamMC covariance estimation

A reference down mix representation is generated by applying the downmix matrix from 5.7.4.4 also on the input audio MDFT samples.

$$X_{l,s}(b) = \sum_{k=0}^{n_i-1} Y_{k,s}(b)\mathbf{M}_{DMX} \quad (5.7\text{-}17)$$

For both downmix and input MDFT representations covariance matrix estimates are calculated for each bin and timeslot in a frame

$$\mathbf{C}_{x,s,b} = Y_s(b)Y_s(b)^* \quad (5.7\text{-}18)$$

$$\mathbf{C}_{x,s,b} = X_s(b)X_s(b)^* \quad (5.7\text{-}19)$$

where $Y_s(b) = \left[Y_{0,s}(b) \dots Y_{n_i-1,s}(b)\right]^T$ is the column vector of the MDFT samples of all input channels of one band and one MDFT time slot and $X_s(b) = \left[X_{0,s}(b) \dots X_{n_t-1,s}(b)\right]^T$ is the column vector of the MDFT samples of all transport channels of one band and one MDFT time slot.

The covariance matrix estimates are accumulated per parameter band b_p for either all MDFT timeslots when no transient was detected or for the MDFT timeslots after the transient in case a transient was detected, and all MDFT bins belonging to a specific parameter band b_p to create the sum of the covariances as a basis for the transmitted channel level and correlation information.

$$\mathbf{C}_{y,b_p} = \sum_{b=2b_{start}}^{2b_{end}+1} \sum_{s=ts_{start}}^{8} \mathbf{C}_{y,s,b} \quad (5.7\text{-}20)$$

$$\mathbf{C}_{x,b_p} = \sum_{b=2b_{start}}^{2b_{end}+1} \sum_{s=ts_{start}}^{8} \mathbf{C}_{x,s,b} \quad (5.7\text{-}21)$$

where $ts_{start} = i_T$ is the first MDFT time slot to be processed.

5.7.4.8 ParamMC default settings for the LFE channel

The parameters for the LFE channel are only sent for the first parameter band. To ensure correct estimates in the following steps, the covariance estimate entries related to the LFE for all other bands are set to zero.

$$c_{y,b_{p_{LFE\_INDEX*}}} = \mathbf{0} \tag{5.7-22}$$

$$c_{y,b_{p_{*\mathrm{LFE\_INDEX}}}} = \mathbf{0} \tag{5.7-23}$$

where b_p is the parameter band, with b_p=1…$n_B - 1$.

5.7.4.9 ParamMC parameter based transient detection

To make sure that in frames where no time domain transient is detected the ICLDs in non-transmitted parameter bands change too much, the ICLDs for those bands are calculated according to clause 5.7.4.13.2, but with no limiting strategy applied and if the number of ICLDs where the difference to the ICLDs of the exceeding a parameter band specific threshold $ICLD_{thr,diff}$ from Table 5.7-18 is greater than 3 the transient flag is set to one. To determine which bands are not transmitted all those are chosen where the current i_F is not the same as the i_F in the configured parameter band mapping.

Table 5.7-18: parameter band specific threshold $ICLD_{thr,diff}$

	Parameter band b_p																			
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
$ICLD_{thr,diff}$	8	8	10	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20

5.7.4.10 ParamMC band combining in case of transients

In case of a set transient flag, the estimates co-variances are of two bands each are combined together to form a new set of estimates for a new set of parameter bands with a reduced number of parameter and with less frequency resolution and the step size s_p is set to 2.

$$\mathbf{C}_{y,b_p} = \begin{cases} \mathbf{C}_{y,b_p} + \mathbf{C}_{y,b_p+1} & if \quad b_p + 1 < n_B \\ \mathbf{C}_{y,b_p} & else \end{cases} \tag{5.7-24}$$

$$\mathbf{C}_{x,b_p} = \begin{cases} \mathbf{C}_{x,b_p} + \mathbf{C}_{x,b_p+1} & if \quad b_p + 1 < n_B \\ C_{x,b_p} & else \end{cases} \tag{5.7-25}$$

where b_p is the parameter band, with b_p=0,s_p, $2s_p$,…$n_B - 1$.

If the transient flag is 0, the step size s_p, is set to 1.

5.7.4.11 ParamMC complex valued to real valued covariance conversion

A parameter band index b_{p,max_abs_cov} is determined, where b_{p,max_abs_cov} is the highest parameter band where $b_{start}(b_p) < 20$. All entries in the (complex valued) covariance estimates for parameter bands below this index are converted to real values using the absolute value of the estimates, all entries for the parameter bands covariance values being equal or above this index are converted to real values using the real part of the complex estimates:

$$\mathbf{C}_{y,b_p} = \begin{cases} |\mathbf{C}_{y,b_p}| & if \quad b_p < b_{p,max\_abs\_cov} \\ \Re\mathrm{e}(\mathbf{C}_{y,b_p}) & else \end{cases} \tag{5.7-26}$$

$$\mathbf{C}_{x,b_p} = \begin{cases} \left|\mathbf{C}_{x,b_p}\right| & if b_p < b_{p,max\_abs\_cov} \\ \Re\mathrm{e}(\mathbf{C}_{x,b_p}) & else \end{cases} \tag{5.7-27}$$

where b_p is the parameter band, with b_p=0,s_p, $2s_p$,…$n_B - 1$.

5.7.4.12 ParamMC LFE activity detection

If the energy of the LFE channel, i.e. the entry in the first parameter band in the main diagonal element of C_y corresponding to the LFE channel is lower than the threshold $PARAM\_MC\_LFE\_ON\_THRESH = 8000$, then the LFE activity flag f_{LFE} is set to zero, otherwise to one.

5.7.4.13 ParamMC parameter quantization

5.7.4.13.1 Parameter quantizers

The parameter quantizers for ParamMC are realised with a non-linear mid-step quantization characteristic, i.e. the quantized value and its index are those of the quantizer level and its index closest to the unquantized value. Quantizers have a length l_q.

5.7.4.13.2 Interchannel Level differences (ICLDs)

The ICLD values are represented as normalized and logarithmic values per parameter band and input channel using the energy P_i of the original channel in this parameter band. The normalization is based on the energy $P_{dmx,i}$ of the reference (transport) channels belonging to the to be quantized, the reference energy being a linear combination of the values of the covariance information of the downmix signal. The ICLDs χ_i for a parameter band are rearranged using a mapping, the mapping parameters can be found in Table 5.7-19. When the parameter band to be processed is the band with LFE $b_p = 0$ and $f_{LFE} = 1$, the number of IDLs to estimate, quantize and code $n_{ILD} = n_{IFD,LFE}$, otherwise $n_{ILD} = n_{IFD,LFE}$.

$$\chi_i = 10\log_{10}\left(\frac{P_i}{P_{dmx,i}}\right) \quad (5.7\text{-}28)$$

$$P_i = c_{y,b_{p_{jj}}},\ j = m_{ILD}(i), \quad (5.7\text{-}29)$$

$$P_{dmx,i} = f_{ICLD}(i)\sum_{k=0}^{k<n_r(i)} c_{x_{ll}} \quad (5.7\text{-}30)$$

where $f_{ICLD}(i)$ is a factor dependent on the bit rate and the input multichannel format (Table 5.7-20).

Table 5.7-19: ICLD maps

MC layout	n_t	$n_{ILD,N}$	$n_{ILD,LFE}$	m_{ild}	n_r	$m_{dmx,ild}$
5.1	2	5	6	{0,1,2,4,5,3}	{1,1,2,1,1,2}	{{0},{1},{0,1},{0},{1},{0,1}}
7.1, 5.1.2	2	7	8	{0,1,2,4,5,6,7,3}	{1,1,2,1,1,1,1,2}	{{0},{1},{0,1},{0},{1},{0},{1},{0,1}}
	3	6	8	{0,1,4,5,6,7,2,3}	{1,1,1,1,1,1,1,1}	{{0},{1},{0},{1},{0},{1},{2},{2}}
5.1.4	3	8	10	{0,1,4,5,6,7,8,9,2,3}	{1,1,1,1,1,1,1,1,1,1}	{{0},{1},{0},{1},{0},{1},{0},{1},{2},{2}}
7.1.4	3	10	12	{0,1,4,5,6,7,8,9,10,11,2,3}	{1,1,1,1,1,1,1,1,1,1,1,1}	{{0},{1},{0},{1},{0},{1},{0},{1},{0},{1},{2},{2}}

Table 5.7-20: ICLD normalization factors f_{ICLD}

MC layout	n_t	f_{ICLD}
5.1	2	{0.391608498887651, 0.391608498887651, 0.208150823035836, 0.323713613305539, 0.323713613305539, 0.208150823035836}
7.1	2	{0.365151211522008, 0.365151211522008, 0.208008958987949, 0.209863469733018, 0.209863469733018, 0.171473949468979, 0.171473949468979, 0.208008958987949}
	3	{0.5, 0.5, 0.29, 0.29, 0.2, 0.2, 1.0, 0.25}
5.1.2	2	{0.36427054, 0.36427054, 0.18290930, 0.21193730, 0.21193730, 0.24564756, 0.24564756, 0.20800895}
	3	{0.49716263, 0.49716263, 0.25198298, 0.25198298, 0.29498283, 0.29498283, 1.0, 0.25}
5.1.4	3	{0.34, 0.34, 0.22, 0.22, 0.20, 0.20, 0.18, 0.18, 1.0, 0.25}
7.1.4	3	{0.3, 0.3, 0.17, 0.17, 0.12, 0.12, 0.19, 0.19, 0.19, 0.19, 1.0, 0.5}

Two limiting strategies are applied to the original channel energies to avoid artefacts. The first one checks if the sum of the energies of the downmix channels is smaller than the sum of the energies of the original channels.

$$E_{tot} = \sum_{i<n_i}^{i=0} C_{y,b_p,i,i}, \quad if \quad i \neq LFE\_INDEX \vee f_{LFE} = 1 \tag{5.7-31}$$

$$E_{dmx} = \sum_{i<n_{dmx}}^{i=0} C_{x,b_p,i,i} \tag{5.7-32}$$

$$fac_{ener} = 10log_{10}\left(\frac{E_{tot}+\epsilon}{E_{dmx}+\epsilon}\right) \tag{5.7-33}$$

If fac_{ener} is larger than the intraframe limiter threshold $PARAM\_MC\_ENER\_LIMIT\_INTRAFRAME = 1.5$, then the first limiter is applied:

$$P_i = 10^{\frac{0.3log(fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTRAFRAME+1)-(fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTRAFRAME)}{10}} P_i$$

The second limiting step checks that ILDs do not change too much

$$\Delta fac_{ener} = fac_{ener} - fac_{ener,prev} \tag{5.7-34}$$

If the frame is not marked as a transient frame and Δfac_{ener} is larger than $PARAM\_MC\_ENER\_LIMIT\_INTERFRAME = 2$ and Δfac_{ener} is smaller than $PARAM\_MC\_ENER\_LIMIT\_MAX\_DELTA\_FAC = 15$ the second limiter is applied and fac_{ener} is corrected:

$$fac_{limit} = 10^{\frac{0.3log(\Delta fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTERFRAME+1)-(\Delta fac_{ener}-PARAM\_MC\_ENER\_LIMIT\_INTERFRAME)}{10}}$$

$$P_i = fac_{limit} P_i \tag{5.7-35}$$

$$fac_{ener} = fac_{ener} + 10log_{10}(fac_{limit}) \tag{5.7-36}$$

fac_{ener} is saved as $fac_{ener,prev}$, if the current frame is a transient frame, it is also saved as the $fac_{ener,prev}$ for the parameter band $b_p + 1$ as long as this is smaller than n_B.

The unquantized ILDs χ_i are quantized using a quantizer as described in clause 5.7.4.13.1 with a length $l_{Q,ICLD} = 16$ (see Table 5.7-21), and the quantizer indices stored parameter band wise for encoding. The unquantized values are also stored per parameter band for the parameter based transient detection in the next frame.

Table 5.7-21: ICLD quantizer

	Quantizer index $i_{Q,ICLD}$															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
$\hat{\chi}_i$	-100	-20	-13	-10	-8	-5.5	-3.5	-1.5	0	1.5	3.5	5.5	8	10	13	20

5.7.4.13.3 Inter-channel coherences (ICCs)

The normalized inter channel coherences ICCs ξ_{ij} for a specific parameter band b_p are generated from the full covariance matrix of the input channels of the parameter band using the relation

$$\xi_{ij} = \frac{c_{y_{ij}}}{\sqrt{c_{y_{ii}} c_{y_{jj}}}} \quad (5.7\text{-}37)$$

Only for a sub-set of all possible channel combinations for the channels i, j the ICCs are estimated, the sub-set depending on the input format and the number of transport channels (Table 5.7-22). When the parameter band to be processed is the band with LFE $b_p = 0$ and $f_{LFE} = 1$, the number of ICCs to estimate, quantize and code $n_{ICC} = n_{ICC,LFE}$, otherwise $n_{ICC} = n_{ICC,LFE}$.

$$\xi_k = \xi_{ij}, i = m_{icc}(k, 0), j = m_{icc}(k, 1) \quad (5.7\text{-}38)$$

where $k = 0,1, \ldots, n_{ICC}$-1 is the index of the ICC within the parameter band b_p.

Table 5.7-22: ICC maps

MC layout	n_t	$n_{ICC,NOLFE}$	$n_{ICC,LFE}$	m_{icc}
5.1	2	4	5	{{0,4},{1,5},{0,2},{1,2},{2,3}}
7.1 5.1.2	2	6	7	{{0,4},{1,5},{0,2},{1,2},{2,6},{2,7},{2,3}}
	3	6	7	{{0,4},{1,5},{0,2},{1,2},{0,6},{1,7},{2,3}}
5.1.4	3	8	9	{{0,4},{1,5},{0,2},{1,2},{0,6},{1,7},{4,8},{5,9},{2,3}}
7.1.4	3	10	11	{{0,4},{1,5},{0,2},{1,2},{0,6},{1,7},{0,8},{1,9},{6,10},{7,11},{2,3}}

The unquantized ICCs ξ_k are quantized using a quantizer as described in clause 5.7.4.13.1 with a length $l_{Q,ICC} = 8$ and the and the quantization levels from Table 5.7-23.The quantization indices are stored in the order determined by the order of the subset in the chosen table for each parameter band.

Table 5.7-23: ICC quantizer

	Quantizer index $i_{Q,ICC}$							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
$\hat{\xi}_k$	-0.99	-0.589	0	0.36764	0.60092	0.84118	0.937	1

5.7.4.14 ParamMC parameter encoding

5.7.4.14.1 Common ParamMC parameter encoding

* At the begin of the metadata the common ParamMC parameters are written as follows:

1) f_{LFE} with 1 bit
2) the encoded band width with 2 bits
3) the parameter frame indicator i_F with 1 bit
4) the transient flag f_t with 1 bit, signalling the occurrence of a transient in the frame

a) if the transient flag is 1, the transient position i_T with 3 bits, signalling in which segment the transient has occurred.

5.7.4.14.2 ICC and ICLD Parameter quantization indices encoding

5.7.4.14.2.1 General parameter indices encoding

The quantized parameter indices are rearranged for all parameter bands belonging to the parameters bands to be encoded determined either by the coding band mapping and the current parameter frame indicator i_F or all bands if it is a transient frame, the number of bands n_{bands} sent being:

$$n_{bands} = \begin{cases} n_{b,I_F} & if \quad f_t = 0 \\ \frac{n_B}{s_p} + n_B \bmod s_p & else \end{cases} \tag{5.7-39}$$

where n_{b,I_F} is defined in Table 5.7-24.

The order is first all parameters not belonging to the coding of the LFE reordered in band direction and the parameters belonging to the coding of LFE at the very end of the sequence if $f_{LFE} = 1$ and either the transient flag is one or the parameter band with index 0 is part of the set of parameter bands to be coded. Both a sequence of absolute indices and delta indices are generated, resulting in two sequences with length l_{seq}. The reordering and creation of the absolute and delta sequences is done according to the following pseudo code:

```
idx_prev = l_Q / 2 + l_Q % 2 - 1;
idx_offset = l_Q - 1;
l_seq = n_NOLFE * n_bands

for ( j = 0; j < n_NOLFE; ++j )
{
    coding_band = 0;

    for ( i = 0; i < n_B; i += s_p )
    {
        if ( f_t == 1 || i_F == coding_band_mapping[i] )
        {
            idx = i_q[i][j];
            seq[coding_band + j * n_bands] = idx;
            seq_delta[coding_band + j * n_bands] = idx - idx_prev + idx_offset;
            idx_prev = idx;
            coding_band++;
        }
    }
}
if ( f_t == 1 )
{
    for ( i = 0; i < 1; i += band_step )
    {
        if ( f_t == 1 || i_F == coding_band_mapping[i] )
        {
             n_lfe_idx = n_LFE – n_NOLFE;
             for ( k = 0; k < n_lfe_idx; k++ )
             {
                 idx = quant_idx[( i + 1 ) * n_lfe - n_lfe_idx + k];
                 seq[sz_seq] = idx;
                 seq_delta[sz_seq] = idx - idx_prev + idx_offset;
                 idx_prev = idx;
                 sz_seq++;
             }
         }
     }
}
```

Table 5.7-24: Table of parameter bands n_{B,I_f} sent depending on the input band width and parameter frame index i_F

Band grouping table	i_F	Band width			
		NB	WB	SWB	FB
$pmc_{b,20}$	0	1	5	9	11
	1	9	9	9	9
$pmc_{b,14}$	0	1	4	6	7
	1	7	7	7	7
$pmc_{b,14}$	0	1	3	4	5
	1	5	5	5	5

Both sequences are encoded using the range coder functions from clause 5.2.2.3.3.3 with a codebook depending on the bit rate and input format with distinct code books for absolute and delta indices according to the following pseudo code:

```
ivas_param_mc_range_encoder( )
{
    rc_uni_enc_init()
    for ( i=0; i < sz_seq ; i++ )
    {
        rc_uni_encode_symbol_fast(seq[i],cum_freq,sym_freq,16);
    }
    rc_tot_bits = rc_uni_enc_finish();

    return total_bit_count;
}
```

The bit demand of the two encoding runs is compared and the minimum bit demand is chosen and a flag f_r is set to 0 if the range coding with absolute indices yields less bits and to 1 if range coding with delta indices yields less bits. To ensure a deterministic upper bound of the bit demand for coding parameters the bit demand per index $n_{uniform}$ determined by the quantizer size is multiplied by the number of indices to code in the sequence. If this uniform coding takes less bits than the better of range coding absolute and delta indices a zero bit is written to the bitstream and each absolute index in the rearranged sequence is written to the bitstream with the number of bits needed for uniform coding. Otherwise, a one is written to the bitstream, followed by writing the flag f_r with one bit followed by the encoded range coding values with the fewer number of bits.

5.7.4.14.2.2 ICLD parameter indices encoding

The quantization indices of the quantized ICLD values are encoded and written to the bitstream using the range coder code books specified in Tables 5.7-25 and 5.7-26, and a uniform bit demand $n_{uniform} = 4$, $l_Q = l_{Q,ICLD}$, $n_{NOLFE} = n_{ICLD,NOLFE}$, $n_{LFE} = n_{ICLD,LFE}$.

Table 5.7-25: Range coder ICC cumulative frequency tables $cum\_freq[]$ and symbol frequency tables $sym\_freq[]$ for delta indices

MC layout	$cum\_freq[]$	$sym\_freq[]$
5.1	{ 0, 4, 9, 124, 447, 1311, 4453, 18116, 48636, 60573, 63692, 64746, 65327, 65531, 65534, 65535}	{ 4, 5, 115, 323, 864, 3142, 13663, 30520, 11937, 3119, 1054, 581, 204, 3, 1}
7.1	{ 0, 2, 7, 53, 243, 979, 3994, 16732, 49642, 61343, 64331, 65158, 65438, 65532, 65534, 65535}	{ 2, 5, 46, 190, 736, 3015, 12738, 32910, 11701, 2988, 827, 280, 94, 2, 1}
5.1.2	{ 0, 3, 8, 36, 172, 763, 3436, 15845, 50168, 62005, 64676, 65298, 65481, 65533, 65534, 65535}	{ 3, 5, 28, 136, 591, 2673, 12409, 34323, 11837, 2671, 622, 183, 52, 1, 1}
5.1.4 7.1.4	{ 0, 3, 7, 74, 304, 1009, 3870, 16502, 49834, 61384, 64217, 65020, 65369, 65531, 65534, 65535}	{ 0, 1092, 5574, 8315, 10652, 13875, 19656, 27664, 36284, 47058, 56251, 62579, 65118, 65462, 65513, 65532, 65535}

Table 5.7-26: Range coder ICC cumulative frequency tables $cum\_freq[]$ and symbol frequency tables $sym\_freq[]$ for absolute indices

MC layout	$cum\_freq[]$	$sym\_freq[]$
5.1	{ 0, 24, 224, 20873, 42384, 51699, 57122, 60572, 65535 }	{ 24, 200, 20649, 21511, 9315, 5423, 3450, 4963}
7.1	{ 0, 30, 848, 26611, 47846, 57358, 61679, 63237, 65535}	{ 30, 818, 25763, 21235, 9512, 4321, 1558, 2298}
5.1.2	{ 0, 46, 826, 27798, 49552, 58447, 62046, 63284, 65535}	{ 46, 780, 26972, 21754, 8895, 3599, 1238, 2251}
5.1.4 7.1.4	{0, 34, 552, 24717, 45819, 54772, 59054, 61166, 65535}	{ 34, 518, 24165, 21102, 8953, 4282, 2112, 4369}

5.7.4.14.2.3 ICC parameter indices encoding

The quantization indices of the quantized ICC values are encoded and written to the bitstream using the range coder code books specified in Tables 5.7-27 and 5.7-28, and a uniform bit demand $n_{uniform} = 3$, $l_Q = l_{Q,ICC}$, $n_{NOLFE} = n_{ICC,NOLFE}$, $n_{LFE} = n_{ICC,LFE}$.

Table 5.7-27: Range coder ICLD cumulative frequency tables $cum\_freq[]$ and symbol frequency tables $sym\_freq[]$ for delta indices

MC layout	$cum\_freq[]$	$sym\_freq[]$
5.1	{ 0, 1, 2, 3, 5, 38, 146, 352, 638, 997, 1559, 2323, 3570, 5859, 10556, 21075, 44682, 55617, 60408, 62739, 63833, 64443, 64809, 65074, 65279, 65400, 65484, 65531, 65532, 65533, 65534, 65535}	{ 4, 5, 115, 323, 864, 3142, 13663, 30520, 11937, 3119, 1054, 581, 204, 3, 1}
7.1	{ 0, 1, 2, 3, 5, 18, 61, 149, 320, 592, 1083, 1793, 2974, 5257, 10133, 21274, 44342, 55891, 60895, 63174, 64244, 64793, 65100, 65287, 65406, 65477, 65517, 65531, 65532, 65533, 65534, 65535}	{ 1, 1, 1, 2, 13, 43, 88, 171, 272, 491, 710, 1181, 2283, 4876, 11141, 23068, 11549, 5004, 2279, 1070, 549, 307, 187, 119, 71, 40, 14, 1, 1, 1, 1}
5.1.2	{ 0, 1, 2, 3, 4, 12, 56, 153, 278, 475, 856, 1430, 2489, 4723, 9580, 20685, 45423, 56274, 60948, 63097, 64128, 64679, 65002, 65208, 65348, 65445, 65517, 65531, 65532, 65533, 65534, 65535}	{ 1, 1, 1, 1, 8, 44, 97, 125, 197, 381, 574, 1059, 2234, 4857, 11105, 24738, 10851, 4674, 2149, 1031, 551, 323, 206, 140, 97, 72, 14, 1, 1, 1, 1}
5.1.4 7.1.4	{ 0, 1, 2, 3, 5, 29, 98, 220, 414, 699, 1186, 1876, 3049, 5304, 10013, 20612, 45247, 56109, 60818, 63022, 64081, 64647, 64977, 65198, 65348, 65443, 65502, 65530, 65532, 65533, 65534, 65535}	{ 1, 1, 1, 2, 24, 69, 122, 194, 285, 487, 690, 1173, 2255, 4709, 10599, 24635, 10862, 4709, 2204, 1059, 566, 330, 221, 150, 95, 59, 28, 2, 1, 1, 1}

Table 5.7-28: Range coder ICLD cumulative frequency tables $cum\_freq[]$ and symbol frequency tables $sym\_freq[]$ for absolute indices

MC layout	$cum\_freq[]$	$sym\_freq[]$
5.1	{ 0, 1092, 5574, 8315, 10652, 13875, 19656, 27664, 36284, 47058, 56251, 62579, 65118, 65462, 65513, 65532, 65535}	{ 1092, 4482, 2741, 2337, 3223, 5781, 8008, 8620, 10774, 9193, 6328, 2539, 344, 51, 19, 3}
7.1	{ 0, 967, 6335, 9941, 12837, 16652, 22416, 29814, 38807, 48497, 57184, 62661, 64916, 65466, 65514, 65530, 65535}	{ 967, 5368, 3606, 2896, 3815, 5764, 7398, 8993, 9690, 8687, 5477, 2255, 550, 48, 16, 5 };}
5.1.2	{ 0, 229, 7068, 10910, 13856, 17467, 22629, 29174, 36906, 46558, 55579, 61802, 65222, 65505, 65527, 65534, 65535}	{ 229, 6839, 3842, 2946, 3611, 5162, 6545, 7732, 9652, 9021, 6223, 3420, 283, 22, 7, 1}
5.1.4 7.1.4	{ 0, 1453, 8326, 12221, 15164, 18764, 24177, 31297, 39520, 49154, 57135, 62460, 64821, 65468, 65514, 65530, 65535}	{ 1453, 6873, 3895, 2943, 3600, 5413, 7120, 8223, 9634, 7981, 5325, 2361, 647, 46, 16, 5}

5.7.4.15 ParamMC transport audio signal encoding

The time domain downmix channels are either encoded with MDCT stereo in case of bit rates using 2 transport channels or with MCT in case of bit rates with 3 transport channels. Note that since we ran the time domain transient detector already in the ParamMC processing it is in the case of ParamMC coding not necessary to run it again in the pre-processing of the core coding, but the results can be directly re-used.

5.7.4.16 ParamMC bit rate switching

If a bit rate switch happens and the last MC mode was also ParamMC, the transport channel configuration the number of transport audio signals n_t, the parameter band grouping, the downmix matrix $\mathbf{M}_{dmx}$ and the parameter band mapping used to determine which parameter bands are sent in a certain frame based on i_F are chosen according to the Tables 5.7-8, 5.7-9, 5.7-10, 5.7-11, 5.7-12, 5.7-13, 5.7-14, and 5.7-16 and the new bit rate.

5.7.5 Parametric upmix MC coding mode

5.7.5.1 General

For 7.1.4 operation at 160 kbps, a parametric upmix coding approach is used offering highly efficient coding or this configuration.

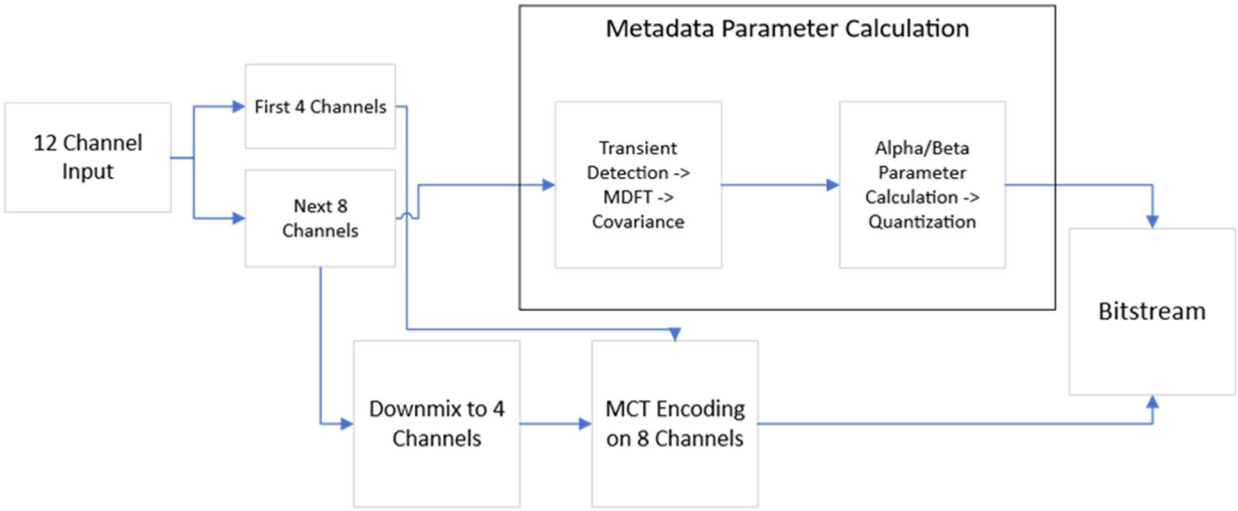


Figure 5.7-5: Schematic of parametric upmix MC encoder

5.7.5.2 Sectioning and downmixing

Input 12 channel 7.1.4 signal is subdivided into two sections P_1 and P_2: Left/Right/Center/LFE (P_1) and the other 8 channels (P_2). The P_2 section is processed by the Metadata Parameter Calculation and Quantization block to generate ParamUpmix parameters. At the same time P_2 is downmixed from 8 channels to 4 by combining the following channels using an index of 0 for the first channel of the input 12 channels: [4+6], [5+7], [8+10], [9+11].

Denoting the channels of the P_2 section as $P_{2,k}$, $k = 0 \ldots 7$, the downmix operation calculates the Mid transformation of the 8 channels as follows:

$$M_{2,k} = \frac{1}{2}(P_{2,k} + P_{2,k+2}) \quad k = 0, 1 \ , \tag{5.7-40}$$

$$M_{2,k} = \frac{1}{2}(P_{2,k+2} + P_{2,k+4}) \quad k = 2, 3 \ . \tag{5.7-41}$$

The 4 Mid channels $M_{2,k}$, $k = 0 \ldots 3$ are associated with the P_1 4 channels to make 8 channels which go on to be coded by the MCT () and respectively the MDCT based LFE codec (see 5.7.2.2).

5.7.5.3 Metadata parameter calculation

The Metadata Parameter Calculation block takes the P_2 channels above and transient detection is performed on each of them according to 5.4.3.2. The P_2 channels are also transformed to frequency using a windowed MDFT that is applied in the same way as for the parameter estimation of the Scene-based audio encoder (5.4.3.3).

The 8 MDFT channels are arranged into 4 pairs $q_k, k = 0 \ldots 3$, of complex data, analogous to the pairs in downmixing above.

$$\begin{aligned} q_k &= \{P_{2,k}, P_{2,k+2}\}, \ k = 0, 1 \\ q_k &= \{P_{2,k+2}, P_{2,k+4}\}, \ k = 2, 3 \end{aligned} \tag{5.7-42}$$

Transient data for the signal pair is created by combining the transient detector data of the first and the second channel with a logical or operation.

The sum of the first and second channel in each pair called the "Mid" signal replaces the second signal in the pair for the covariance calculation which follows.

These pairs of first signal and "Mid" signal are passed into the Covariance Calculator along with the transient detector data (TD) above. A covariance matrix is calculated as described in subsection 5.4.3.7.2 with the following differences:

Calculation of a covariance matrix for a channel pair results in a 2x2 matrix. Thus, with $S_1^{p\_upmix\_ana,mdft}(\mathrm{m},k)$ representing the MDFT transformed first signal and $S_2^{p\_upmix\_ana,mdft}(\mathrm{m},k)$ being the MDFT transformed "Mid" signal, equations (5.4-18) to (5.4-20) for banded covariance calculation turn into

$$V(m,k) = \begin{bmatrix} S_1^{p\_upmix\_ana,mdft}(\mathrm{m},k) \\ S_2^{p\_upmix\_ana,mdft}(\mathrm{m},k) \end{bmatrix} \tag{5.7-43}$$

$$C_{[2x2]}^{in,mdft}(m,k) = re(V(m,k)V^H(m,k)) \tag{5.7-44}$$

and

$$C_{[i,j]}^{in,banded}(b) = \sum_{m=1}^{m=2}\sum_{k=\boldsymbol{bin_{start}}(b)}^{\boldsymbol{k=bin_{end}}(b)}(C_{[\mathrm{i,j}]}^{in,mdft}(m,k)\,H_{abs,5ms}^{mdft}(b,k)) \tag{5.7-45}$$

Only transient detector output signal T_0 is used to reset covariance smoothing. The resetting mechanism by setting $\alpha(b)$ to a value of 1.0 from equation (5.4-23) is replaced by

$$\alpha(b) = \begin{cases} 1, & frameIdx_{prev} = -1\ OR\ T_0 = 1 \\ \alpha_{sm}(b), & otherwise \end{cases} \tag{5.7-46}$$

Covariance smoothing is based on different factors compared to the banded covariance smoothing for SBA operation. Computation of the forgetting factor $\alpha_{sm}(b)$ is achieved by modification of equations (5.4-24) and (5.4-25) to

$$N_b = \sum_{k=\boldsymbol{bin_{start}}(b)}^{\boldsymbol{bin_{end}}(b)} H_{abs}^{mdft}(b,k) \tag{5.7-47}$$

and

$$\alpha_{sm}(b) = \min\left(1.0, \frac{\mathrm{N}_b}{20}\right) \tag{5.7-48}$$

where $H_{abs}^{mdft}(b,k)$ is the magnitude response of the 20ms block size MDFT filterbank (described in clause 5.2.5) for band index b and frequency bin index k, $bin_{start}(b)$ is the bin index of first non-zero magnitude response bin of the 20ms block size MDFT filter bank and $bin_{end}(b)$ is the bin index of last non-zero magnitude response bin of the 20ms block size MDFT filter bank.

For each of the 12 parameter frequency bands the output will be a matrix of covariance coefficients, i.e., for each pair of channels k and for each frequency band b there will be a 2x2 matrix:

$$C_k(b) = \begin{Bmatrix} C_{k,1,1}(b) & C_{k,1,2}(b) \\ C_{k,2,1}(b) & C_{k,2,2}(b) \end{Bmatrix},\ k = 0 \ldots 3, b = 0 \ldots 11\ , \tag{5.7-49}$$

Subsequently, the covariance coefficients are used to calculate Alpha and Beta parameters. Using helper variable

$$C = \frac{C_{k,2,1}(b)}{(C_{k,2,2}(b)+\epsilon)}, \tag{5.7-50}$$

Alpha is obtained according to:

$$Alpha_k(b) = 2C - 1\text{ , and} \tag{5.7-51}$$

Beta is obtained according to:

$$Beta_k(b) = 2\sqrt{\frac{\max(0, C_{k,1,1}(b) - C^2 C_{k,2,2}(b))}{(C_{k,2,2}(b)+\epsilon)}}. \tag{5.7-52}$$

In the above ϵ is a very small positive number used to avoid divisions by zero and numerical stability.

5.7.5.4 Quantization

Alpha parameters are quantized to the nearest value given by the following table:

Table 5.7-29: Quantizer for Alpha parameter

Index	Value	Index	Value
0	-2.0	17	0.190625
1	-1.809375	18	0.3625
2	-1.6375	19	0.515625
3	-1.484375	20	0.65
4	-1.35	21	0.765625
5	-1.234375	22	0.8625
6	-1.1375	23	0.940625
7	-1.059375	24	1.0
8	-1.0	25	1.059375
9	-0.940625	26	1.1375
10	-0.8625	27	1.234375
11	-0.765625	28	1.35
12	-0.65	29	1.484375
13	-0.515625	30	1.6375
14	-0.3625	31	1.809375
15	-0.190625	32	2.0
16	0.0		

Beta parameters are quantized using a two-stage process where alpha parameters are first quantized using the above method, and these quantized values are used to index into a set of tables which beta parameters will be quantized by, to the nearest value in the table.

The below table gives the beta quantization table index to give the beta table to be used.

Table 5.7-30: Quantizer selector Beta parameter quantizer

Quantized Alpha Index	Beta Table Index	Quantized Alpha Index	Beta Table Index
0	0	17	1
1	1	18	2
2	2	19	3
3	3	20	4
4	4	21	5
5	5	22	6
6	6	23	7
7	7	24	8
8	8	25	7
9	7	26	6
10	6	27	5
11	5	28	4
12	4	29	3
13	3	30	2
14	2	31	1
15	1	32	0
16	0		

All possible quantization tables for the beta parameter are depicted in the following table.

Table 5.7-31: Quantizers for Beta parameter

Beta Table Index			Beta Table Index		
0	**Quantized Beta Index**	**Value**	**5**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.2375		1	0.101123
	2	0.55		2	0.2341797
	3	0.9375		3	0.3991699
	4	1.4		4	0.5960938
	5	1.9375		5	0.8249512
	6	2.55		6	1.085742
	7	3.2375		7	1.378467
	8	4.0		8	1.703125
1	**Quantized Beta Index**	**Value**	**6**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.2035449		1	0.08386719
	2	0.4713672		2	0.1942188
	3	0.8034668		3	0.3310547
	4	1.199844		4	0.494375
	5	1.660498		5	0.6841797
	6	2.18543		6	0.9004688
	7	2.774639		7	1.143242
	8	3.428125		8	1.4125
2	**Quantized Beta Index**	**Value**	**7**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.1729297		1	0.06995117
	2	0.4004688		2	0.1619922
	3	0.6826172		3	0.276123
	4	1.019375		4	0.4123438
	5	1.410742		5	0.5706543
	6	1.856719		6	0.7510547
	7	2.357305		7	0.9535449
	8	2.9125		8	1.178125
3	**Quantized Beta Index**	**Value**	**8**	**Quantized Beta Index**	**Value**
	0	0.0		0	0.0
	1	0.1456543		1	0.059375
	2	0.3373047		2	0.1375
	3	0.5749512		3	0.234375
	4	0.8585938		4	0.35
	5	1.188232		5	0.484375
	6	1.563867		6	0.6375
	7	1.985498		7	0.809375
	8	2.453125		8	1.0
4	**Quantized Beta Index**	**Value**			
	0	0.0			
	1	0.1217188			
	2	0.281875			
	3	0.4804688			
	4	0.7175			
	5	0.9929688			
	6	1.306875			
	7	1.659219			
	8	2.05			

5.7.5.5 Entropy Coding

Quantized alpha and beta indices are Huffman coded differentially in frequency. The quantized indices for the lowest frequency band are encoded using non-differential Huffman coding, then the indices for each subsequent higher

frequency band is subtracted from the indices of the previous band, and Huffman coded. The determined codes are then written to the bitstream.

5.7.6 Discrete MC coding mode

For the discrete MC coding mode the LFE channel is coded separately using an MDCT-based encoding mode as described in clause 5.7.2. The remaining channels are then coded using the Multichannel Coding Tool (MCT) algorithm described in 5.2.3.4.

5.7.7 MC bitrate switching

Switching to a different bitrate in multi-channel format might also require a change of the MC mode. A selection of the current MC mode is done based on table 5.7-1.

When switching to McMASA mode (see clause 5.7.3), the number of transport channels and the separate channel parameters are determined based on the new bitrate. Then the necessary initializations of McMASA buffers and variables are performed. If the previous MC mode was already McMASA, the McMASA coder is completely re-initialized with settings according to the new bitrate.

When switching to ParamMC mode (see clause 5.7.4) from a different MC mode, the correct number of transport channels is set and the necessary initializations of ParamMC buffers and variables are done. If the previous mode was also ParamMC the coder is reconfigured according to clause 5.7.4.3.

When switching to ParamUpmix MC mode (see clause 5.7.5) from a different MC mode, the correct number of transport channels is set and the necessary initializations of paramUpmix buffers and variables are done.

When switching to DiscMC mode (see clause 5.7.6), the number of transport channels is set to the number of channels in the input MC layout.

Finally, a core-coder reconfiguration is performed to reflect possible changes in number of transport channels and per-channel bitrate.

5.8 Combined Object-based audio and SBA (OSBA) operation

5.8.1 OSBA format overview

The encoder supports combined input with 1 – 4 ISMs and an SBA signal of order 1 – 3. Depending on the IVAS total bitrate, different OSBA coding modes summarized in Table 5.8-1 are employed to combine these input signals.

Table 5.8-1: Overview of coding modes in OSBA format

IVAS bitrate [kbps]	number of ISMs			
	1	2	3	4
13.2 – 80	Pre-rendering	Pre-rendering	Pre-rendering	Pre-rendering
96	Discrete	Pre-rendering	Pre-rendering	Pre-rendering
128 – 512	Discrete	Discrete	Discrete	Discrete

Input to IVAS, consisting of audio signals in SBA and ISM formats and the associated metadata, is processed through a simplification stage and an encoding stage. At the simplification stage, the SBA and ISM signals are converted into a mezzanine format, as described in clauses 5.8.2 and 5.8.3, where the mezzanine format depends on the IVAS bitrate and SBA coding as described in clause 5.4. At bitrates less than 256 kbps, the mezzanine format is the First order Ambisonics (FOA) format whereas at bitrates greater than or equal to 256 kbps it includes FOA channels, selected HOA channels and all discrete ISM objects. At the encoding stage, the simplified audio output of the simplification stage is encoded into IVAS bitstream which is then transmitted to the decoder.

5.8.2 Low-bitrate pre-rendering OSBA coding mode

In the low-bitrate OSBA mode, the encoder generates a description of a combined audio scene based on a first-order ambisonics signal.

The input interface receives via the IVAS encoder API calls, one signal describing an audio scene in ISM format (first format) and another signal describing an audio scene in ISM format (second format) at the same time.

The first scene description in the first format (ISM format) is then converted into first-order ambisonics as the common format. The SBA input signal (second format) is truncated to first order. The ISM input signal is converted to first-order ambisonics by panning the objects in the ambisonics domain to the positions described by the metadata:

$$\boldsymbol{x}_{FOA,O}(t) = \boldsymbol{Y}(\theta_O)x_O(t) = \left[x_O(t)Y_{00},\ x_O(t)Y_{1-1}, x_{O(t)}Y_{1,0}, x_O(t)Y_{11}\right]. \quad (5.8\text{-}20)$$

$\boldsymbol{x}_{FOA,O}(t)$ is the converted audio signal corresponding to the object with the index O. $\boldsymbol{Y}(\theta_O)$ is the vector of the real spherical harmonics evaluated at the DoA angle θ_O that represents the position of the object on the sphere. $x_O(t)$ is the object audio signal in time-domain.

From the two scenes described in the common format (first-order ambisonics) the description of the combined audio scene is acquired. Specifically, the two first-order ambisonics signals describing the two individual scenes are added up and multiplied by a factor of 0.5, thereby obtaining the description of the combined audio scene in the common format.

From this common format (first-order ambisonics), transport channels are generated in the SPAR downmixing (cf. clause 5.4.5). This downmix is performed in the on each time-frequency tile.

On the low frequency bands, the covariance is estimated from the first-order ambisonics channels of the combined scene in the common format. On the high-frequency bands, a DirAC parameter analysis is performed, generating direction-of-arrival and diffuseness values. These parameters represent the combined scene. They are, in turn, used in the estimation of the covariance for the calculation of the downmix matrix (cf. clauses 5.4.3.8.3, 5.4.3.8.5 and 5.4.4.1) and encoded by the DirAC metadata encoder according to clause 5.4.3.4).

The transport channels in the first set resulting from the downmix are then encoded using the MCT according to clause 6.1.4.3), which, in turn, is based on EVS as a core-coder. The bitstream containing the encoded metadata and audio channels is then written to the encoder output interface (IVAS encoder API).

In summary, the processing of the first-order ambisonics signal (common format and common scene) is equivalent as a regular ambisonics input at the respective bitrate. The only difference to the bitstream of a regular coded SBA signal is that the number of ISMs in the input is signaled to the decoder such that the correct number of channels can be provided for external output.

5.8.3 High-bitrate discrete OSBA coding mode

In the high-quality high-bitrate OSBA mode, the objects are coded with the regular ISM processing as in clause 5.6.2. The SBA scene is coded as in the regular SBA coding at the respective bitrate. In principle, the regular SBA and ISM encoders run alongside each other in an unmodified way. The configuration of the SBA encoder is the same as with regular SBA input at the same total IVAS bitrate.

The ISM metadata are coded in the bitstream together with the SBA metadata and the required bits are subtracted from the budget of the core-coder. The downmix audio channels of the SBA encoder and the object audio channels are encoded jointly using the MCT (see clause 5.2.3.4). Thereby a dynamic bit allocation between the audio downmix channels of the SBA encoder and the object audio channels is achieved. An object that is silent, therefore, uses only very few bits.

5.8.4 OSBA bitrate switching

When the bitrate is switched in OSBA format, both encoders (SBA and ISM) are re-configured. The configuration is the same as is there were running as separate instances of IVAS. One special case for OSBA is the switching between bitrates corresponding to different OSBA coding modes. Then the encoder switches between the pre-rendering and the discrete coding mode.

When the high-bitrate mode is switched on, additional re-configurations are required as compared to bitrate switching for SBA in clause 5.4.10. Specifically, the number of MCT channels is set according to the SBA configuration and the number of objects. The ISM mode flag is set to signal discrete object coding.

When the high-bitrate mode is switched off, the ISM mode flag is set to signal pre-rendering mode.

5.9 Combined Object-based audio and MASA (OMASA) operation

5.9.1 OMASA format overview

The IVAS codec supports combined input format of ISMs and MASA called OMASA (Objects with Metadata-Assisted Spatial Audio). The input consists of 1 – 4 independent objects with associated metadata and 1 or 2 channels of MASA and the associated metadata. While the coding of MASA part is based on stereo-MASA coding, the coding of objects depends on a coding mode where an inter-format dependency between objects and MASA channels allows to improve the performance and lower the complexity and memory resources when compared to the coding of input audio using MASA format and ISM format individually. In OMASA format, DTX is not supported and only the first group of ISM metadata (i.e. azimuth and elevation) is considered.

5.9.2 OMASA format configurations

The OMASA format comprises four (4) coding modes while a selection of the actual coding mode is based on the number of objects and IVAS total bitrate as summarized in table 5.9-1. These coding modes are low-bitrate pre-rendering (Rend) coding mode, one object with MASA representation (One) coding mode, parametric one object (Param) coding mode, and discrete (Disc) coding mode. Their detailed description is provided in following clauses.

Table 5.9-1: Overview of bitrates and coding modes in OMASA format

IVAS total bitrate [kbps]	coding mode and ISM nominal bitrate [kbps]							
	1 ISM		2 ISMs		3 ISMs		4 ISMs	
	mode	ISM bitrate [kbps]	mode	ISM bitrate [kbps]	mode	ISM bitrate [kbps]	mode	ISM bitrate [kbps]
13.2	Rend	0	Rend	0	Rend	0	Rend	0
16.4	Rend	0	Rend	0	Rend	0	Rend	0
24.4	Disc	9.6	Rend	0	Rend	0	Rend	0
32	Disc	13.2	Param	13.2	One	13.2	One	13.2
48	Disc	16	Disc	11	One	16	One	16
64	Disc	16	Disc	11.7	Param	16	Param	16
80	Disc	20	Disc	16	Param	20	Param	20
96	Disc	32	Disc	20	Disc	20	Param	32
128	Disc	32	Disc	24.4	Disc	24	Disc	24
160	Disc	48	Disc	32	Disc	24.4	Disc	24
192	Disc	64	Disc	48	Disc	32	Disc	24.4
256	Disc	96	Disc	64	Disc	48	Disc	32
384	Disc	128	Disc	80	Disc	64	Disc	48
512	Disc	128	Disc	96	Disc	80	Disc	64

Table 5.9-1 lists also values of nominal bitrate used for coding one object in ISM coder. E.g. at 13.2 kbps with 1 ISM there is used Rend mode and no bitrate is allocated for ISM coding (there is no ISM coding at all), or at 64 kbps and 3 ISMs there is used Param mode and the nominal bitrate for coding the one separated object is 16 kbps, or at 512 kbps with 4 ISMs there is used Disc mode and the nominal bitrate for coding four ISMs is 4 x 64 kbps. It is noted that the difference between the IVAS total bitrate and nominal ISM bitrate forms the MASA nominal bitrate which is related to the coding of the MASA part of the OMASA input audio. The MASA nominal bitrate is used to determine the MASA configurations according to clause 5.5.3.2. In addition, if the MASA nominal bitrate is less or equal to 32 kbps, the OMASA low rate mode is activated, and it influences the encoding of the MASA-to-total energy ratio parameters described in clause 5.9.6.3.2.

Figure 5.9-1 then shows a block diagram of the OMASA encoder. It can be seen from figure 5.9-1 that the OMASA encoder receives $N_{ISM} + 1$ or 2 channels where N_{ISM} is the number of input ISMs and "1 or 2" corresponds to the number of MASA input channels. First, the OMASA mode is set in the configuration module (see table 5.9-1). Then the input channels are subject of OMASA analysis, pre-coding and mixing (clause 5.9.3) forming M_{ISM} ISM transport channels and 2 MASA transport channels. The two MASA transport channels are coded using the CPE coding tool (clause 5.2.3.3) while M_{ISM} ISM transport channels are coded using the ISM coder (clause 5.6). The number of ISM coded channels is equal to 0, 1, or N_{ISM} depending on the OMASA mode. Finally, a combined format bitrate adaptation logic (clause 5.9.8) is used to distribute the available IVAS total bit-budget between MASA and ISM coding parts.

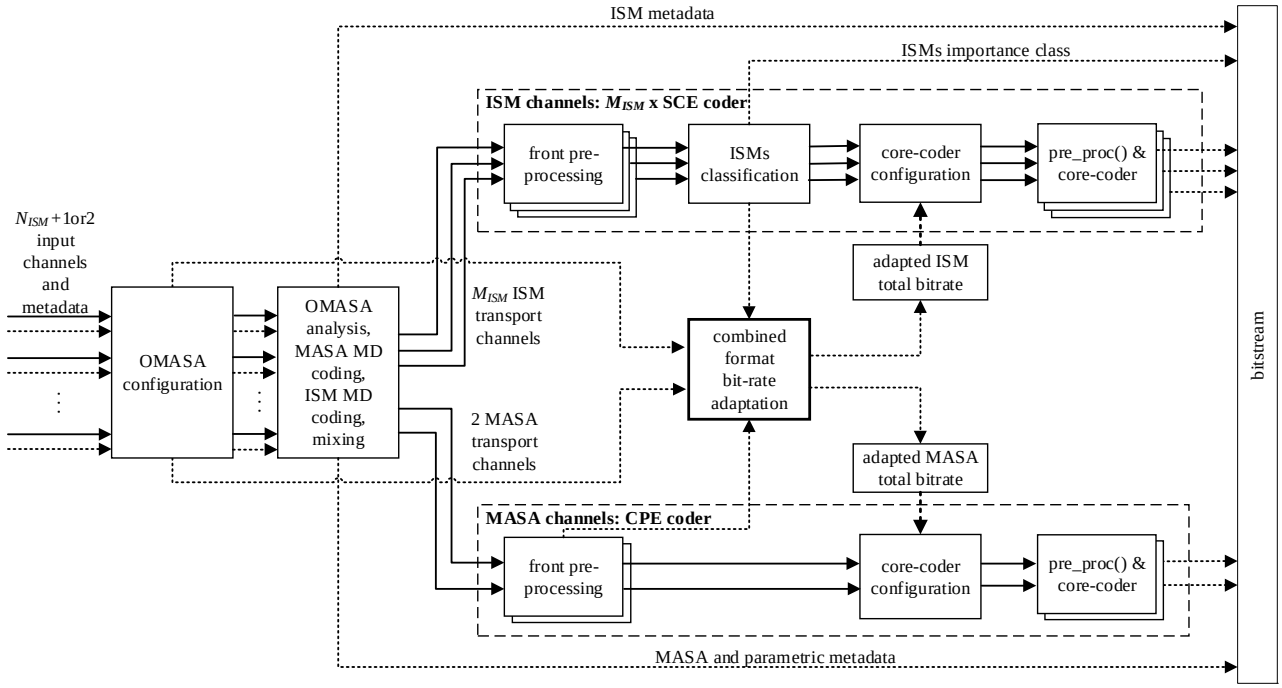


Figure 5.9-1: Block diagram of OMASA encoder

5.9.3 OMASA pre-coding processing tools

5.9.3.1 OMASA spatial audio parameter analysis

First, the object audio signals $s(n,i)$ are converted to the time-frequency domain using the filter bank described in clauses 5.2.5 and 5.4.3.3. The outcome is the time-frequency domain object signals $S(k,m,i)$ (where k is the frequency bin index, m the subframe index corresponding also to the slot index, and i is the object index).

Then, the time-frequency domain object signals are converted to FOA signals (where the channels 0, 1, 2, and 3 correspond to spherical harmonics W, Y, Z, and X) by

$$S_{FOA}(k,m,0) = \sum_{i=0}^{N_{ISM}-1} S(k,m,i)$$

$$S_{FOA}(k,m,1) = \sum_{i=0}^{N_{ISM}-1} \sin\theta_{obj}(i)\cos\phi_{obj}(i)\,S(k,m,i)$$

$$S_{FOA}(k,m,2) = \sum_{i=0}^{N_{ISM}-1} \sin\phi_{obj}(i)\,S(k,m,i)$$

$$S_{FOA}(k,m,3) = \sum_{i=0}^{N_{ISM}-1} \cos\theta_{obj}(i)\cos\phi_{obj}(i)\,S(k,m,i)$$

where N_{ISM} is the number of input objects, $\theta_{obj}(i)$ the azimuth angle of the object i for this frame, and $\phi_{obj}(i)$ the elevation angle.

The energy of the object signals is determined by

$$E(b,m) = \sum_{k=k_1(b)}^{k_2(b)} \sum_{i=0}^{N_{ISM}-1} |S(k,m,i)|^2$$

and k_1 and k_2 are the first and the last bin of the frequency band b. The frequency bands follow the MASA frequency bands (see Table 5.5-1).

The azimuth $\theta(b,m)$ and elevation $\phi(b,m)$ angles are determined using the FOA signals $S_{FOA}(k,m,\mathrm{i})$ using the methods described in clause 5.4.3.4. The diffuseness $r'_{diff}(b,m)$ is determined using the FOA signals $S_{FOA}(k,m,\mathrm{i})$ using the methods described in clause 5.4.3.4.

The diffuseness values $r'_{diff}(b,m)$ computed for different subframes are combined so that all the subframes within a frame have the same value. This is performed determining the value for the first subframe by

$$r_{diff}(b,0) = \frac{\sum_{m=0}^{3} E(b,m) r'_{diff}(b,m)}{\sum_{m=0}^{3} E(b,m)}$$

and setting the same value to all subframes m of the frame, yielding $r_{diff}(b,m)$.

The direct-to-total energy ratio is determined from the diffuse-to-total energy ratio by

$$r_{dir}(b,m) = 1 - r_{diff}(b,n)$$

The spread and surround coherences are set to zero

$$\zeta(b,m) = 0$$

$$\gamma(b,m) = 0$$

5.9.3.2 OMASA MASA metadata combining

This clause describes the merging of the MASA spatial audio parameters from the audio objects stream (the ISM objects in OMASA) in the OMASA format with the spatial audio parameters from the MASA audio stream. For the purpose of this clause, the MASA audio stream consists of the MASA spatial audio parameters and transport audio signals (the MASA part in OMASA), and the audio objects stream consists of MASA format spatial audio parameters and transport audio signals (the ISM part in OMASA), as determined in clauses 5.9.3.1 and 5.9.3.3. The merging process is done for each parameter coding band b and parameter subframe m independently, in other words, for each TF-tile separately.

The merging is performed based on the following criteria.

For the MASA audio stream, the signal energy parameter E_{MASA} describing the energy of the MASA transport audio signals, and the direct-to-total energy parameter $r_{dir;MASA}$ are obtained. The signal energy parameter E_{MASA} can either be calculated directly from the MASA transport audio signals or received as an input to the metadata merging process. Based on the signal energy parameter E_{MASA} and the direct-to-total energy parameter $r_{dir;MASA}$ associated with the original audio MASA stream, a weighting value w_{MASA} for the MASA audio stream is generated with

$$w_{MASA}(b,m) = E_{MASA}(b,m) r_{dir;MASA}(b,m)$$

Similarly for the audio objects stream, the signal energy parameter E_{ISM} of the transport audio signals and the direct-to-total energy ratio parameter $r_{dir;ISM}$ from the spatial audio parameters associated with the audio objects stream are obtained. As mentioned above, the transport audio signals and spatial audio parameters of the audio objects stream are in the MASA spatial metadata and MASA transport audio signal format generated from the original audio objects (ISM objects) according to clauses 5.9.3.1 and 5.9.3.3.

A diffuse-energy compensated direct-to-total energy ratio parameter $r_{dir;comp}$ is then generated based on the MASA audio stream diffuse-to-total energy ratio $r_{diff;MASA}$, signal energy parameter E_{MASA}, and the energy parameter E_{ISM} of the audio objects stream. The diffuse-energy compensated direct-to-total energy ratio parameter $r_{dir;comp}$ is determined as

$$r_{dir;comp}(b,m) = 1 - \frac{r_{diff;MASA}(b,m) E_{MASA}(b,m)}{E_{MASA}(b,m) + E_{ISM}(b,m)}$$

A weighting value w_{ISM} for the audio objects stream is then generated based on the diffuse-energy compensated direct-to-total energy ratio parameter, the direct-to-total energy ratio parameter $r_{dir;ISM}$, and the signal energy parameter E_{ISM} of the audio objects stream as

$$w_{ISM}(b,m) = E_{ISM}(b,m)\frac{r_{dir;comp}(b,m) + r_{dir;ISM}(b,m)}{2}$$

The merging process is then performed for each TF-tile by selecting either the MASA spatial audio stream parameters or the audio objects stream (in MASA format) spatial audio parameters as the merged MASA spatial audio parameters. The merging process is based on the comparison of the weighting value w_{MASA} with the weighting value w_{ISM} and is expressed according to the following:

When $w_{ISM}(b,m) > w_{MASA}(b,m)$, the MASA format direction parameters from the audio objects stream are taken as the merged direction parameters $\theta'(b,m)$ and $\phi'(b,m)$, and either the diffuse-energy compensated direct-to-total energy ratio parameter $r_{dir;comp}(b,m)$ or the diffuse-to-total energy ratio parameter $r_{dir;ISM}(b,m)$ from the audio objects stream is taken as the direct-to-total energy ratio parameter $r'_{dir}(b,m)$ for the merged parameters.

$$\theta'(b,m) = \theta_{ISM}(b,m)$$

$$\phi'(b,m) = \phi_{ISM}(b,m)$$

$$r'_{dir}(b,m) = min\left(r_{dir;ISM}(b,m), r_{dir;comp}(b,m)\right)$$

$$\zeta'(b,m) = \zeta_{ISM}(b,m)$$

$$r'_{diff}(b,m) = 1 - r'_{dir}(b,m)$$

Otherwise, the spatial parameters of the original MASA stream, including the original direct-to-total energy ratio parameter, are used as for the merged direction parameters and merged direct-to-total energy ratio.

$$\theta'(b,m) = \theta_{MASA}(b,m)$$

$$\phi'(b,m) = \phi_{MASA}(b,m)$$

$$r'_{dir}(b,m) = r_{dir;MASA}(b,m)$$

$$\zeta'(b,m) = \zeta_{MASA}(b,m)$$

$$r'_{diff}(b,m) = 1 - r'_{dir}(b,m)$$

The value of the energy parameter of the TF-tile of the transport audio signals associated with the merged MASA spatial audio parameters is updated by

$$E'(b,m) = E_{MASA}(b,m) + E_{ISM}(b,m)$$

5.9.3.3 OMASA audio signals downmix

First, the stereo amplitude panning gains $g(i,j)$ are determined for each object i and downmix channel j, based on the object directions (azimuth $\theta_{obj}(i)$ and elevation $\phi_{obj}(i)$), as described in clause 7.2.2.2.6 (for the "stereo" mode operation).

Then, an interpolator is determined

$$g_{interp}(n) = \frac{n}{L_{frame}}$$

Using these gains, the downmix signals are created by amplitude panning the object signals to the downmix stereo signals

$$s_{obj,dm}(n,j) = \sum_{i=0}^{N_{ISM}-1} \left(g_{interp}(n)g(i,j) + \left(1 - \left(g_{interp}(n)\right)g^{[-1]}(i,j)\right)s(n,i)\right)$$

where $g^{[-1]}(i,j)$ is the amplitude panning gains determined for the previous frame.

5.9.3.4 Determination of an object to be separated

The process of determining which audio object is to be separated from an audio object stream is performed on a per audio frame basis and requires as an input both the audio objects audio signals and the MASA audio signals. $s(n,i)$ is used to refer to both MASA and audio objects audio signals, $s_{obj}(n,i)$ is used to refer to the audio objects audio signals, and $s_{MASA}(n,i)$ is used to refer to the MASA audio signals. There are $N = N_{obj} + N_{MASA}$ input channels, where N_{obj} is the number of audio object channels and N_{MASA} is the number of MASA channels. The output from this process for a frame is the channel index of the separated audio object, the separated audio object channel index i_{sep}. The process is as follows.

First, the energy of each input channel is determined by

$$E(i) = \sum_{n=0}^{L_{frame}-1} s(n,i)^2$$

where L_{frame} is the length of the input audio frame in samples. These calculated energies are then smoothed over time using

$$E_{sm}(i) = 0.2E(i) + 0.8E_{sm}^{[-1]}(i)$$

where $E_{sm}^{[-1]}(i)$ is the smoothed energy value for channel i from the previous audio frame.

Then, the loudest object $i_{loudest}$ is determined by selecting the channel which has the largest value of $E_{sm}(i)$ (this selection is performed among the audio objects channels).

The determination of the separated audio object depends on the "change object mode" η of the previous frame. If the "change object mode" was enabled in the previous frame, then the loudest object is set as the separated object, i.e.,

$$i_{sep} = i_{loudest}, \quad if\ \eta^{[-1]} = 1$$

If the "change object mode" was not enabled in the previous frame (i.e., $\eta^{[-1]} = 0$), the following is performed.

First, it is checked if the loudest object for this frame is the same as the separated object of the previous frame. If it is, the loudest object is then separated, i.e.,

$$i_{sep} = i_{loudest}, \quad if\ i_{sep}^{[-1]} = i_{loudest}$$

If this is not the case (i.e., $i_{sep}^{[-1]} \neq i_{loudest}$), then the following is performed.

First, the selected channels energy is determined by

$$E_{sel} = E(i_{loudest}) + E\left(i_{sep}^{[-1]}\right) + E^{[-1]}(i_{loudest}) + E^{[-1]}\left(i_{sep}^{[-1]}\right)$$

which is a sum of the energies of the current and the previous frames for the loudest object of the current frame and the separated audio object of the previous frame.

Then, the sum of the energies for all the input channels of the current and the previous frames is determined

$$E_{tot} = \sum_{i=0}^{N-1} \left(E(i) + E^{[-1]}(i)\right)$$

Then, by using E_{tot} and E_{sel}, the selected channel energy ratio, which is a measure of the proportion of the energy of the selected channel to the total energy of all the input channels, is computed by

$$r_{sel} = \frac{E_{sel}}{E_{tot}}$$

Then, an adaptive threshold (in decibels) is determined by

$$\tau_{sel} = 9r_{sel} + 1$$

Then, the ratio between the loudest audio object and the previous separated audio object is computed in decibels by

$$r_{objects} = 10\log_{10}\left(\frac{E_{sm}(i_{loudest})}{E_{sm}\left(i_{sep}^{[-1]}\right)}\right)$$

The ratio between the audio objects is compared to the adaptive threshold to determine if the separated audio object is to be changed. If the ratio is not larger than the threshold, the separated audio object is not changed, i.e.,

$$i_{sep} = i_{sep}^{[-1]}, \quad if\ r_{objects} \leq \tau_{sel}$$

If the ratio is larger than the threshold (i.e., $r_{objects} > \tau_{sel}$), then the following is performed. First, it is checked if the selected channels ratio r_{sel} is smaller than the threshold of 0.25 (i.e., $r_{sel} < 0.25$). In that case, the levels of $i_{sep}^{[-1]}$ and $i_{loudest}$ are relatively low in comparison to the total energy (e.g., if the MASA part is louder), and the separate audio object is changed with a "hard switch" as the other channels are masking possible artefacts from the abrupt change in the signal waveform, i.e.,

$$i_{sep} = i_{loudest}, \quad if\ r_{sel} < 0.25$$

If the selected channels ratio is not smaller than the threshold of 0.25 (i.e., $r_{sel} \geq 0.25$), the separate audio object channel is not changed abruptly. Instead, fade out fade in procedure is performed (explained in detail in clause 5.9.3.5).

For the current frame, the separated channel of the previous frame is used, i.e.,

$$i_{sep} = i_{sep}^{[-1]}, \quad if\ r_{sel} \geq 0.25$$

Moreover, the "change object mode" η is enabled for this frame (i.e., $\eta = 1$). As a result, a new separate audio object channel will be determined in the next frame. In all other cases, the "change object mode" η is disabled for this frame (i.e., $\eta = 0$).

5.9.3.5 Separation of an object from other objects

The separation of the object from the other objects depends on the value of the "change object mode" η determined in clause 5.9.3.4. The audio signal $s(n, i)$ refers to the audio objects channels in this clause. The separation can be performed via fade out fade in when the separated audio object changes, or the separated audio object can change instantly. The fade out gains are determined by

$$g_{fadeout}(n) = \frac{1 + \cos\left(\pi \frac{n}{L_{frame}}\right)}{2}$$

The fade in gains are determined by

$$g_{fadein}(n) = 1 - g_{fadeout}(n)$$

If the "change object mode" is enabled for this frame (i.e., $\eta = 1$), the separate audio object signal is determined by

$$s_{sep}(n) = s\left(n, i_{sep}\right) g_{fadeout}(n)$$

The remaining audio objects signal is determined by

$$\begin{cases} s_{rem}(n, i) = s(n, i) g_{fadein}(n), & if\ i = i_{sep} \\ s_{rem}(n, i) = s(n, i), & if\ i \neq i_{sep} \end{cases}$$

If the "change object mode" is not enabled for this frame, but it was enabled for the previous frame (i.e., $\eta^{[-1]} = 1$), the separate audio object signal is determined by

$$s_{sep}(n) = s\left(n, i_{sep}\right) g_{fadein}(n)$$

The remaining audio objects signal is determined by

$$\begin{cases} s_{rem}(n, i) = s(n, i) g_{fadeout}(n), & if\ i = i_{sep} \\ s_{rem}(n, i) = s(n, i), & if\ i \neq i_{sep} \end{cases}$$

If the "change object mode" is not enabled for this frame nor the previous frame (i.e., $\eta = 0$ and $\eta^{[-1]} = 0$), the separate audio object signal is determined by

$$s_{sep}(n) = s(n, i_{sep})$$

The remaining audio objects signal is determined by

$$\begin{cases} s_{rem}(n,i) = 0, & if\ i = i_{sep} \\ s_{rem}(n,i) = s(n,i), & if\ i \neq i_{sep} \end{cases}.$$

5.9.4 Low-bitrate pre-rendering (Rend OMASA) coding mode

5.9.4.1 Overview

In this mode, the objects are converted to a MASA stream, and this stream is combined with the original MASA stream (i.e., the one that was inputted to the encoder), and the resulting MASA stream is encoded as a normal MASA stream.

First, in case the MASA stream has two-direction metadata, it is converted to one-direction metadata using the method described in clause 5.5.3.2.7 (all bands are one-direction bands in this case).

Then, the MASA spatial audio parameters are determined as presented in clause 5.9.3.1. The determined MASA spatial audio parameters are merged with the original MASA spatial audio parameters (using the converted one-direction parameters in case the input was two-direction parameters) as detailed in clause 5.9.3.2.

Then, the object audio signals are downmixed to a stereo downmix as detailed in clause 5.9.3.3. The created downmix signals are merged with the MASA transport audio signals of the original MASA stream by

$$s_{merged}(n,i) = s_{orig}(n,i) + s_{obj,dm}(n,i)$$

where $s_{orig}(n,i)$ is the original MASA transport audio signal, $s_{obj,dm}(n,i)$ the stereo downmix signal created from the objects, n is the time domain sample index, and i is the channel index.

5.9.4.2 Low-bitrate pre-rendering coding method

Once the MASA and audio object content have been merged to the MASA format, consisting of the 2 transport audio signal channels and MASA metadata, encoding of the aforementioned "merged" MASA format is performed on the basis that the "merged" MASA format is treated as a stereo-MASA format content at the corresponding overall bitrate. The coding format is signalled as a MASA format at the beginning of the bitstream. In addition, the number of input audio objects is encoded into the bitstream using two bits reserved from the encoding of the MASA metadata, which is as follows:

- '01' if there are 4 objects,
- '10' if there are 3 objects,
- '11' if there are 1 or 2 objects.

In the instance that the number of input audio objects is 1 or 2, i.e., encoded as '11' according to the list above, the bit used to signal the number of transport channels for the MASA format (the MASA number of transport channel signal bit) is repurposed for use in distinguishing between 1 and 2 audio objects. Thus, when the MASA number of transport channel signal bit is '0' this indicates the case of 1 input audio object, and when the MASA number of transport channel signal bit is '1' this indicates the case of 2 input audio objects. In the instance that the number of objects is 3 or 4, the MASA number of transport channel signal bit is used for the encoding of the combined MASA format audio signal (formed by combining the input audio objects converted into the MASA format with the input MASA audio signal comprising the transport audio signals and MASA metadata).

5.9.5 One object with MASA representation (One MASA) coding mode

5.9.5.1 Overview

In this mode, an audio object is adaptively determined for each frame to be separated from the other objects. Then, this object is separated from the other objects and encoded separately with an audio object encoder. The remaining audio objects are encoded together with the MASA format input stream. The combined encoding is performed by converting

the remaining audio objects to a MASA stream, combining it with the original MASA stream (i.e., the one that was inputted to the encoder), and encoding the resulting combined MASA stream.

First, the object to be separated from the input objects for this frame is determined as presented in clause 5.9.3.4. Then, this object is separated from the other objects as presented in clause 5.9.3.5, yielding $s_{sep}(n)$ (i.e., the separated object signal) and $s_{rem}(n, i)$ (i.e., the remaining objects signals).

Then, the remaining audio objects $s_{rem}(n, i)$ are converted to a MASA stream and combined with the input MASA stream. First, in case the MASA stream has two-direction metadata, it is converted to one-direction metadata using the method described in clause 5.5.3.2.7 (all bands are one-direction bands in this case).

Then, the MASA spatial audio parameters are determined using the remaining audio objects $s_{rem}(n)$ as presented in clause 5.9.3.1. The determined MASA spatial audio parameters are merged with the original MASA spatial audio parameters (using the converted one-direction parameters in case the input was two-direction parameters) as detailed in clause 5.9.3.2.

Then, the remaining object audio signals $s_{rem}(n, i)$ are downmixed to a stereo downmix as detailed in clause 5.9.3.3. The created downmix signals are merged with the MASA transport audio signals of the original MASA stream by

$$s_{merged}(n, i) = s_{orig}(n, i) + s_{obj,dm}(n, i)$$

where $s_{orig}(n, i)$ is the original MASA transport audio signal and $s_{obj,dm}(n, i)$ the stereo downmix signal created from the remaining objects, and n is the time domain sample index and i is the channel index.

5.9.5.2 One object with MASA representation coding method

The resulting two components that are encoded in this mode are one of the input audio objects with its metadata and the stereo MASA format data. The bitrate allocation between the two components is determined at each frame according to the procedure described in clause 5.9.8. The separated object, which might be a different object at each frame, is encoded with its corresponding directional metadata according to the procedures presented in ISM format coding from clause 5.6. The encoding of the separated object directional metadata follows the ISM metadata encoding from clause 5.6.4.2. However, the encoding is additionally conditional on the comparison between the current frame selected object and to the previous frame selected object. If as a result of the comparison a change to the selected separated object has been detected the encoding of the associated directional metadata is forced to function in an absolute coding mode rather than a differential coding mode for at least 5 consecutive frames to prevent potential sudden changes of spatial direction in noisy channel conditions. The stereo MASA format part is encoded following the procedures of the MASA format encoding from clauses 5.5.2 and 5.5.3.

5.9.6 Parametric one object (Param OMASA) coding mode

5.9.6.1 Overview

In this mode, an audio object is adaptively determined for each frame to be separated from the other objects. Then, this object is separated from the other objects and encoded separately with an audio object encoder. The remaining audio objects are encoded together with the MASA format input stream. The combined encoding is performed by first determining parametric information related to the relations between the objects and between the objects and the MASA stream. This parametric information and object metadata are then encoded, and the MASA metadata is encoded. Then, the audio signals of the remaining objects and the MASA stream are combined and encoded together.

First, the object to be separated for this frame is determined as presented in clause 5.9.3.4. Then, this object is separated from the other objects as presented in clause 5.9.3.5, yielding $s_{sep}(n)$ (i.e., the separated object signal) and $s_{rem}(n, i)$ (i.e., the remaining objects signals).

Then, parametric information related to the relations between the objects and between the objects and the MASA stream are determined as presented in clause 5.9.6.2.

Then, the remaining object audio signals $s_{rem}(n, i)$ are downmixed to a stereo downmix as detailed in clause 5.9.3.3. The created downmix signals are merged with the MASA transport audio signals of the MASA stream by

$$s_{merged}(n, i) = s_{orig}(n, i) + s_{obj,dm}(n, i)$$

where $s_{orig}(n, i)$ is the MASA transport audio signal and $s_{obj,dm}(n, i)$ the stereo downmix signal created from the remaining objects, and n is the time domain sample index and i is the channel index.

5.9.6.2 Determination of OMASA parametric information

In this operating mode, the MASA transport audio signal(s) and associated metadata parameters, and the remaining object audio signal(s) and their associated parameters are processed into combined audio signals and associated combined metadata parameters, such that the combined parameters include all direction parameter elements of both the MASA stream and remaining audio object stream inputs. The generated combined audio signals are a mix of these inputs in order to produce fewer number of audio channels for encoding. Additional combined format ratio parameters are determined that enable the amplitude-based separation of these mixed signals at the decoder.

These additional ratio parameters are determined according to:

First, the object audio signals $s(n, i)$ are converted to the time-frequency domain using the filter bank described in clauses 5.2.5 and 5.4.3.3. The result is the time-frequency domain audio object signals $S(k, m, i)$, where the indices k, m denote the frequency and time index respectively of a time-frequency tile and i denotes the index of the object audio signal.

The energy for each object signal is determined by

$$E_{obj}(b, m, i) = \sum_{k=k_1(b)}^{k_2(b)} |S(k, m, i)|^2$$

and k_1 and k_2 are the first and the last bin of the frequency band b.

The ISM energy ratios are then determined by

$$r_{ISM}(b, m, i) = \frac{E_{obj}(b, m, i)}{\sum_{i=0}^{N_{ISM}-1} E_{obj}(b, m, i)}$$

In the case when there is no energy in any of the audio objects signals for the time-frequency tile (b, m), the ISM ratios are set to $r_{ISM}(b, m, i) = 1/N_{ISM}$ for time-frequency tiles (b, m) across all audio objects N_{ISM}, so that they sum to 1.

Then, the MASA-to-total energy ratio can be determined by

$$r_{MASA2tot}(b, m) = \frac{E_{MASA}(b, m)}{E_{MASA}(b, m) + \sum_{i=0}^{N_{ISM}-1} E_{obj}(b, m, i)}$$

where $E_{MASA}(b, m)$ is the energy of the MASA stream, determined in clause 5.5.2.3. The MASA-to-total energy ratio is an audio scene separation metric between the MASA stream and the remaining audio object stream which is used for separating the MASA signal parts and the remaining audio object signal parts.

5.9.6.3 One object with parametric representation coding method

5.9.6.3.1 Coding method overview

The elements to be encoded in the one audio object with parametric representation are: the separated object audio content, the directional metadata of all audio objects, the MASA transport channels, the MASA metadata, and the combined format ratios as determined in clause 5.9.6.2 consisting of the ISM energy ratios and the MASA-to-total energy ratios. The allocation of bits between the separated audio object and the MASA format is performed according to the method described in clause 5.9.8, however part of the bitrate allocated for the encoding of the MASA format is also used for the encoding of the combined format ratios. The process of encoding the one audio object with parametric representation on a TF tile basis is as follows:

1. The total number of audio objects in the audio object stream is encoded into the bitstream by using two bits to encode the number of objects minus 1.

2. If there is more than 1 object in the audio object stream, then the index of the separated object is encoded into the bitstream using 2 bits.

3. The importance flag of the separated object is written into the bitstream using 2 bits.
4. Include the reserved 2 bits for the MASA stream into the bitstream.
5. The number of sound source directions of the TF tile for the MASA format is signalled using 1 bit to indicate the number of directions minus 1.
6. The subframe mode is signalled for the TF tile using one bit where a “1” indicates the one subframe mode for the frame and a “0” indicates the 4 subframe mode for the frame.
7. The low bitrate is signalled on 1 bit only if the maximum number of bits allowed for metadata is less than 100 and if the number of subframes is 4. If the low bitrate signalling conditions are met, the low bitrate mode is activated if the codec total bitrate is less or equal to 32 kbps.
8. The combined format ratios are quantized and encoded as presented in clauses 5.9.6.3.2 and 5.9.6.3.3.
9. The MASA metadata is quantized and encoded as laid out in clause 5.5.3.3.
10. The quantized MASA-to-total energy ratio for the TF tile from point 8 above is then used to update the quantized values of the total-to-direct energy ratio for the tile by multiplying the total-to-direct energy ratio with the MASA-to-total energy ratio for the TF tile.

5.9.6.3.2 Encoding of MASA-to-total energy ratios

The MASA-to-total energy ratios of all time frequency tiles of a frame are encoded at once. They are first arranged as a (*nblocks*×*nbands*) matrix where *nblocks* is the number of subframes and *nbands* is the number of sub bands. The matrix is 2D-DCT transformed. The maximum number of bits that can be used for the encoding of the MASA-to-total energy ratios is 50 if the encoding is in low bitrate mode or 1000 (equivalent to no limit) if there is no low bitrate restriction. Depending on the number sub bands and sub frames the total number of parameters can be 4, 5, 8, 12, 20 or 32. The obtained DCT coefficients are scalar quantized with a quantization step of 0.1. The obtained integer indexes are reordered using *ReorderGeneric*() function transforming them into positive integer indexes. All but the case when the number of coefficients is 32 are encoded as a single streak. The 32 coefficients case is encoded in 4 streaks of 8 coefficients. Encoding as one streak means that the encoding parameters are optimized on the streak data and there are common encoding parameter values per streak.

The first quantized coefficient index of the first streak is encoded on 7 bits using the first bit for its sign, and the following 6 bits for encoding in clear the value that is less or equal to 63. The first quantized coefficient index of the following streaks (when present) is quantized in clear on 6 bits. The remaining quantized coefficients indexes are encoded with Golomb Rice as follows. If the length of the streak is larger than or equal to 8, two optimal Golomb Rice parameters are obtained, one for the first part of the streak and the second one for the second part of the streak. The Golomb Rice optimization is done amongst the following order combinations: (2, 0), (2, 1), (1, 0). If the streak length is lower than 8 then a single Golomb Rice parameter is used for all quantized coefficients indexes of the streak. If the obtained number of bits is larger than the maximum bit allowed, the values of the quantized indexes are gradually decreased by 2 units (for indexes larger than 1) or by 1 unit (for indexes smaller than 2). After decreasing one index value, the optimal Golomb Rice parameters are calculated, and the resulting number of bits estimated. The gradual decrease continues as long as the obtained number of bits is larger than the allowed number of bits. When the maximum number of allowed bits is reached the following data is written in the bitstream for each streak: the number of indexes that are Golomb rice encoded with the first order (if needed) written on 4 bits, the first Golomb Rice order on 1 bit, the second Golomb Rice order on 1 bit (only if there were two orders and if the first one was equal to 2), the Golomb Rice encoded indexes. After the DCT coefficients are quantized and encoded, the quantized MASA-to-total energy ratios are reconstructed.

5.9.6.3.3 Encoding of ISM energy ratios

As presented above, the ISM energy ratios represent the content of a specific audio object within the audio object part of the total MASA and audio objects audio environment. The ISM energy ratios are defined for each audio object at the time frequency (TF) tile level. As a result, the sum of the ISM energy ratios over all the objects for one TF tile is one. This constraint together with other considerations results in a selection of the ISM energy ratios that is required to be transmitted whilst still conveying the necessary information. The encoding process is described in the following.

In essence the encoding process for the ISM energy ratios of the objects of a TF tile can be summarised as follows. The ISM energy ratios are quantized and encoded only when there is more than one object present in the input. All ISM energy ratios of the audio objects are considered for quantization, however only a selection of ISM energy ratios is

quantized and encoded. Encoding of the ISM energy ratios is performed on a per sub band basis comprising the following steps; firstly, a prior analysis is performed, which is followed by a reordering stage and then a selection of the ISM energy ratio quantization indexes is performed. The encoding step uses a combination of absolute encoding by jointly indexing the quantization indexes for the first subframe of a sub band and a differential coding approach for the subsequent subframes of the sub band. In addition, differential coding might also be considered for the first sub frames, but only for sub bands starting with the second one.

The following sections detail the encoding process for the ISM energy ratios of the audio objects of a TF tile.

For each TF tile, an initial quantization of the ISM energy ratios associated with all the audio objects in the TF tile is performed by considering the ISM ratios of the audio objects of the selection simultaneously, in the form of a vector. The ISM ratios are each quantized using *nb* = 3 bits, with a uniform K level quantizer having a step size of $\sigma = \frac{1}{K}, K = 7$.

If the ISM ratio of the separated audio object is smaller than σ, its quantized value is set to zero and it is not considered in the main quantization procedure, presented below. The quantization of the remaining ISM ratios, which will either be N_{obj} or $N_{obj} - 1$ (in the case of the separated audio object not being considered) is performed as follows, where O is used to signify the number of audio objects for which the ISM energy ratios are quantized in the main quantization:

1. If the MASA to total energy ratio for the current TF tile is larger than a threshold (0.98)
    a. The quantized ISM energy ratio indexes are equally distributed among all audio objects while ensuring that they sum up to *K*. For example, when *K* = 7, and there are 3 audio objects present in the TF tile the available quantized indices *idx* can be distributed amongst the three audio objects as (3, 2, 2).
2. Else
    a. For o = 1:O
        i. Quantize to the lowest nearest neighbor the ISM energy ratios $\widehat{rISM}(o)$ and obtain the index *idx(o)* (i.e., from the two adjacent possible quantized values, select the one that has the lower value). In other words, for each $rISM(o)$, this step involves finding the two adjacent quantized ISM energy ratio values $r_i \leq rISM(o) \leq r_{i+1}, i = 0{:}\, K$ and quantizing the ISM energy ratio with the lowest value ($\widehat{rISM}(o) = r_i$ and give the index *idx(o)*=*i* for the ISM energy ratio quantization index).
    b. End For
    c. Calculate the reconstructed quantized values of the ISM ratios (o = 1: O) using the formula $idx(o) \cdot \sigma$, where $\sigma < 1$ is the quantization step corresponding to the number of bits *nb*.
    d. Calculate the Euclidean distortion between the reconstructed ISM energy ratios arranged as a vector and their corresponding unquantized ISM energy ratios ($rISM(o)$) (o = 1: O) also arranged as a vector
    e. Calculate the sum of quantized indexes using $SI = \sum_{o=1}^{O} idx(o) \cdot \sigma$
    f. While $SI < \frac{1}{\sigma}$
        i. Check one by one which quantized index $idx(o)$ can be increased by 1 unit for a decrease of the resulting Euclidean distortion in the ISM energy ratio domain
        ii. Select the quantized index component ($idx(o)$) that results in the largest decrease in the Euclidean distortion or, if there cannot be any decrease, the one that increases it by the least amount.
        iii. Update the selected index component, by adding 1 unit
        iv. Update the sum of quantized indexes (*SI* = *SI*+1)
    g. End While
3. End If

4. Correct the quantized indexes
5. Encode the quantized indexes

Step 4 is applied for the instances when the following three conditions are met: the separated object ISM ratio is considered in the main quantization procedure, the corresponding MASA to total energy ratio is less than a threshold (0.98) and the resulting quantized index for the separated object ISM energy ratio is larger than zero. If this is the case, the value of the separated object quantized ISM energy ratio index is evenly distributed to the other ISM energy ratio indexes and the index of the quantized ISM energy ratio of the separated object is set to zero. This way it is ensured that the sum of the ISM energy ratio indexes for all the objects is *K*. In step 5, the encoding of the quantized indexes *idx* (of the quantised ISM energy ratios), is performed for each sub frame and each sub band in turn. Before encoding the vector of integer quantized indexes for a sub frame and sub band, where the components of the vector are the indexes *idx* of the quantized ISM energy ratios, an additional reordering of the group of indexes *idx* is performed. This is then followed by a selection of a number of the *idx* components in order to form a vector of selected quantised ISM energy ratio indexes to be sent for encoding. All audio objects are considered in this step. If the last audio object is the separated audio object, the order of the quantised ISM energy ratio indexes is changed such that the component index corresponding to last audio object is switched with the component index corresponding to the first audio object. After this, only the first $N_{obj} - 1$ component indexes are selected to be encoded. Furthermore, if the separated object quantized ISM energy ratio indexes corresponding to the first subframe, of the current sub band are all zero, then only the first N_{obj} - 2 quantized ISM ratio indexes are selected to be encoded.

The encoding procedure for the resulting selection of $T = N_{obj}$ - 1, or $T = N_{obj}$ - 2 indexes is performed as follows. For the first sub frame of each sub band the vector of selected quantized ISM ratio indexes is indexed either using an enumeration algorithm with the resulting integer index being transmitted to the decoder or using a differential coding scheme with respect to the ISM energy ratio index value from the corresponding subframe from the previous sub band. Determining which of the two different schemes to use is based on the estimation of the bit consumption. The differential coding scheme uses a Golomb Rice encoder with parameter 0.

Upon completion of the above quantization procedure, each component of the vector of selected quantized ISM energy ratio indexes has effectively been quantized based on *nb* bits and the sum of the resulting vector components has been constrained to *K*, where $K = 2^{nb} - 1$. The enumeration of integer vectors obeying this restriction is performed by listing the *T*-digit integers and considering only those whose decomposition in base 10 corresponds to valid vectors obeying the above restriction. As an example, for 3 objects, when $T = 2$ and $K = 7$, the following table 5.9-2 exemplifies how the enumeration index is formed:

Table 5.9-2: Enumeration index of ISM ratio quantization indexes

Number	Vector	Valid/Not valid	Index
0	0 0 (7)	Valid	0
1	0 1 (6)	Valid	1
2	0 2 (5)	Valid	2
3	0 3 (4)	Valid	3
4	0 4 (3)	Valid	4
5	0 5 (2)	Valid	5
6	0 6 (1)	Valid	6
7	0 7 (0)	Valid	7
8	**0 8 (-1)**	**Not valid**	
9	**0 9 (-2)**	**Not valid**	
10	1 0 (7)	Valid	8
11	1 1 (5)	Valid	9
12	1 2 (4)	Valid	10
13	1 3 (3)	Valid	11
14	1 4 (2)	Valid	12
…			
70	7 0 (0)	Valid	35

The column “Vector” corresponds to the vector of indexes obtained from the decomposition in base 10 of the integer number of the first column, the value in brackets is the component of the quantised ISM energy ratio vector which is not part of the selection, i.e. the component which is not sent. The last column “Index” indicates the corresponding enumeration index, as an increasing enumeration index for the valid vectors. The enumeration algorithm generates within a loop an increasing index up to a maximum value, it verifies if the corresponding vector is valid or not, and it increases the value of the resulting index by one unit when the vector is valid. The corresponding pseudo-code for

producing the enumeration index for the encoding of a vector of *n* positive integer values y_i , whose sum equals *K* is presented below. The output is the enumeration index *index.*

1. Take first *T* vector components of the vector of selected ISM energy ratio indexes, with component indexes *idx*
2. Form the number corresponding to the maximum possible index x for the input: $x = \sum_{i=0}^{T-2} idx_i \cdot 10^{T-2-i}$
3. Let $i = 0$, *index* = 0
4. While $i \leq x$
    a. If the vector of indexes corresponding to *i* is valid
        i. *index* = *index* +1;
    b. End if
    c. $i = i+1$
5. End while
6. Return *index*

The above enumeration is used for indexing of the ISM energy ratio quantization indexes vector for the first subframe (of the frame) in the encoding algorithm of the vector of ISM ratio quantization indexes:

1. For the first subframe, the selected quantized ISM energy ratio indexes are encoded with the above enumeration index encoding or with the differential encoding with respect to previous sub band.

2. For each sub frame from 2 to *nblocks,* differential encoding is used to encode the selected quantized ISM energy ratio indexes for these remaining subframes.

    a. For each sub band from 1 to *nbands*

        i. Calculate for each audio object 1: N_{obj} the difference index between the quantized ISM energy ratio index for a current subframe with respect to the corresponding quantized ISM energy ratio index from the previous sub frame

        ii. Transform the difference index for each audio object to a positive index

        iii. The positive indexes of all audio objects are encoded with Golomb Rice (GR) code with parameter 0 and the corresponding number of bits are estimated

        iv. The positive indexes of all the audio objects are encoded with GR code with parameter 1 and corresponding number of bits are estimated

        v. Calculate for each object the difference index between the quantized ISM energy ratio index for a current sub band with respect to the corresponding quantized ISM energy ratio index from the previous sub band (if there is no previous sub band, use data from previous subframe)

        vi. Transform the difference index for each audio object to positive index

        vii. The positive indexes of all audio objects are encoded with GR code with parameter 0 and corresponding number of bits are estimated

        viii. The positive indexes of all audio objects are encoded with GR code with parameter 1 and corresponding number of bits are estimated

    b. End For

    c. For each differential coding mode (with respect to previous sub band /with respect to previous sub frame)

        i. Select the optimal GR parameter for using the mode over all sub band data in current sub frame

    d. End for

    e. Select the differential coding mode (difference to previous subframe or previous sub band) for the current sub frame as the one giving the shortest codelength for the sub frame

3. End for

5.9.6.3.4 Encoding of ISM metadata

In the one separated audio object with parametric representation coding mode, the directional metadata associated with the N_{ISM} audio objects are quantized and encoded. As mentioned in clause 5.9.6.3.1 when the IVAS codec is operating in the one separated audio object mode the audio object stream is configured into a stream consisting of the separated audio object and a stream consisting of the remaining audio objects and the MASA format audio.

The MASA-to-total energy ratio represents the distribution of the MASA format contribution to the total audio environment for a TF tile, while the ISM energy ratio for each remaining audio object represents the distribution of the remaining audio objects within the audio object contribution to the total audio environment. As a result, after the quantization and encoding of the MASA-to-total energy ratio and of the ISM energy ratios, the distribution of each object within the total audio content for each time frequency tile is known. Based on this information, an object priority order is determined that is further used to determine the bit allocation for the directional metadata associated with each audio object.

The priority order of each audio object is obtained by:

$$p(i) = \max_{\substack{b=1:\text{nbands},\\ k=1:\text{nblocks}}} \left(\left(1 - r_{MASA2tot}(b,k)\right) r_{ISM}(i,b,k) \right), i = 1{:}\, N_{ISM} \quad (5.9\text{-}1)$$

where $r_{ISM}(i,b,k)$ is the ISM ratio for the audio object i for the TF tile b,k, $r_{MASA2tot}(b,k)$ is the MASA-to-total energy ratio for the TF tile b,k. This is performed for all audio objects $1{:}\, N_{ISM}$ which encompasses all remaining audio objects and the separated audio object.

In addition, the priority value for the separated audio object with index i_{sep}, (where i_{sep} is an index from the set $1{:}\, N_{ISM}$) is determined separately based on the above determined priority values of all the audio objects:

$$p(i_{sep}) = \begin{cases} 1, & \text{if } class_{ISM}(i_{sep}) == \text{ISM\_HIGH\_IMP} \\ \frac{1 + \max_{i=1:N_{ISM}} p(i)}{2}, & \text{if } class_{ISM}(i_{sep}) == \text{ISM\_MEDIUM\_IMP} \\ \max_{i=1:N_{ISM}} p(i), & other \end{cases} \quad (5.9\text{-}2)$$

where $class_{ISM}(i_{sep})$ is the ISM importance class for the separated audio object as assigned in clause 5.6.2.3.2 based on the audio signal type given by the *coder_type*.

The number of bits allocated for the quantization of the directional metadata for each audio object, (for all audio objects including the separated audio object), is given by:

$$bits_{ISM,MD}(i) = 11 - \left[\left(1 - p(i)\right) * 7 \right], i = 1{:}\, N_{ISM} \quad (5.9\text{-}3)$$

where the index $i = 1{:}\, N_{ISM}$ incorporates the index of the separated audio object i_{sep} and [ ] stands for the integer part operation.

For each audio object, quantization of the audio directional parameter for the current frame, where the directional parameter consists of an azimuth value and an elevation value, is performed as follows:

If the bit allocation is less than 8 bits, the directional parameter of the audio object is compared to the directional parameter of the same audio object from the previous frame. If they are the same, that is within a threshold of 0.01 degrees then one bit is sent to signal that they are the same. If at least one of elevation value or azimuth value of the directional parameter is different from the corresponding elevation value or azimuth value of the previous frame's directional parameter , then one bit is sent to signal they are different, and the remaining bits of the bit allocation are used to quantize the current frame's audio directional parameter using a spherical grid quantizer based on the remaining number of bits as described in clause 5.2.4.3.2.

If the bit allocation for the audio directional parameter of the audio object is higher or equal to 8, then the audio directional parameter is quantized and encoded using a spherical grid quantizer based the allocated number of bits as described in clause 5.2.4.3.2.

5.9.7 Discrete (Disc OMASA) coding mode

In the Disc OMASA coding mode, all input ISMs are coded separately incl. their metadata, thus $M_{ISM} = N_{ISM}$ in Figure 5.9-1. Consequently, the OMASA analysis and OMASA downmix modules are omitted in the Disc OMASA mode. Then the ISM channels are coded using the ISM format encoder (clause 5.6) and the two MASA transport channels are coded using the MASA encoder (clause 5.5) while an inter-format bitrate adaptation (further described in clause 5.9.8) is used to distribute the IVAS total bit-budget between the ISM and MASA parts of the OMASA format.

The ISM format encoder as part of the OMASA encoder includes M_{ISM} SCE tools (clause 5.2.3.2) where all the M_{ISM} ISM channels are analyzed and processed in parallel in a front pre-processor of the ISM format encoder (clause 5.6). The front pre-processing produces ISM pre-processing parameters including classification information. The classification information, in particular the VAD or the coder type, is passed to an ISM classifier performing a classification based on an ISM importance (clause 5.6.2.3.2). The ISM classifier includes also a metadata processor where the ISM metadata of all the M_{ISM} ISM channels are analyzed, quantized, coded and finally supplied to the bitstream. The ISM metadata coding and quantization is described in clause 5.6.4.

The ISM classification information further serves as a basis for the bitrate adaptation algorithm (clause 5.9.8) that distributes the available bit-budget among all the M_{ISM} ISM channels from the ISM classifier using the core-coder configurator of the ISM format encoder from Figure 5.9-1. The available bit-budget for the ISM encoder is then the bit-budget corresponding to the ISM total bitrate minus ISM metadata and ISM signaling bits while the initial (nominal) ISM total bitrate is set in the OMASA configuration module from Figure 5.9-1. The core-coder configuration module further sets high-level parameters of the core-coder, for example the internal sampling rate or the coded audio bandwidth, based on the nominal ISM total bitrate.

When the core-coder configuration and bitrate distribution between the ISM channels is done, the ISM format coding continues with a sequential further pre-processing (further classification, core selection, other resampling, etc.) and core-coding using the adapted ISM total bitrate. The core-coder comprises M_{ISM} fluctuating bitrate SCE coder tools which sequentially encode all the M_{ISM} ISM channels and the core-coder indices are finally written to the bitstream.

The MASA format encoder as part of the OMASA encoder then includes a MASA front pre-processor, a MASA core-coder configuration, and a pre-processor and core-coder performing a further pre-processing. The MASA audio channels – regardless of mono- or stereo-MASA input – are always coded using the CPE coding tool (clause 5.2.3.3).

The MASA format encoder starts with MASA metadata coding and then continues, similarly to the ISM format encoder, with the front pre-processing. Next, the core-coder configurator receives the information about the initial (nominal) MASA total bitrate from the OMASA configuration module and sets the high-level core-coder parameters. Finally, the further pre-processing and core-coding is performed on the two MASA audio channels using the adapted MASA total bitrate and the core-coder indices are written to the bitstream. The MASA metadata coding and quantization are described in clause 5.5.3.3.

5.9.8 Inter-format bitrate adaptation

The inter-format bitrate adaptation logic adaptively distributes the IVAS total bit-budget between coding of the MASA part audio channels and the ISM related audio channels in the combined OMASA format. The distribution logic is based on ISM and MASA pre-processing parameter and it is related to the following operations (also shown in Figure 5.9-1) front pre-processing of the ISMs which produces ISM pre-processing parameters, 2) front pre-processing of the MASA which produces MASA pre-processing parameters, 3) adaptation of ISM and MASA nominal (initial) bitrates to adapted ISM and MASA bitrates, 4) core-coding of ISM channels using an adapted ISM total bitrate, 5) core-coding of MASA channels using an adapted MASA total bitrate.

In OMASA, the constant IVAS total codec bitrate, $brate_{IVAS}$, represents a sum of the MASA format variable bitrate $brate_{MASA}$ (i. e. the bitrate to encode the MASA audio channels with MASA metadata) and the variable ISM total bitrate $brate_{ISM}$ (i.e. the sum of bitrates to encode all ISMs with their metadata):

$$brate_{IVAS} = brate_{MASA} + brate_{ISM}. \quad (5.9\text{-}4)$$

The bitrates allocated to individually coded ISM audio channels $m = 1, \dots, M_{ISM}$ are further denoted as $brate_{ism}(m)$ and they hold

$$\sum_{m=1}^{M_{ISM}} brate_{ism}(m) = brate_{ISM} \quad (5.9\text{-}5)$$

where M_{ISM} is the number of separately coded audio objects which holds $M_{ISM} \leq N_{ISM}$ and N_{ISM} is the number of input ISMs. The number of coded ISMs depends on the OMASA coding mode and it is $M_{ISM} = 1$ in One and Param OMASA

coding modes from clauses 5.9.5 and 5.9.6, respectively, or $M_{ISM} = N_{ISM}$ in the Disc OMASA coding mode from clause 5.9.7. The inter-format bitrate adaptation logic is thus used in the three of four OMASA coding modes.

The flexible inter-format bitrate adaptation module as shown in Figure 5.9-1 first receives the information about classification of the M_{ISM} ISMs from the ISMs classification module from Figure 5.9-1 while the ISM classification is based on the ISM front pre-processing parameters. It also receives an information from MASA pre-processing module. Further, the bitrates of audio object channels are adapted based on the classification information from ISM and MASA pre-processing modules and results in the adapted variable ISM total bitrate $brate_{ISM}$. Consequently, the adapted MASA total bitrate, $brate_{MASA}$, is variable as well.

The ISMs classification module used in OMASA uses the parameters and logic from clause 5.6.2.3.2 and defines four ISM classes: inactive class ISM_INACTIVE, low importance class ISM_LOW_IMP, medium importance class ISM_MEDIUM_IMP, and high importance class ISM_HIGH_IMP. In addition to the classification logic from clause 5.6.2.3.2, the classification is altered depending on the IVAS total bitrate. Specifically, the ISM_INACTIVE class is not used at highest bitrates where the nominal (initial) ISM bitrate for current ISM $brate_{ism}(m) > 48000$ bps or the low-pass filtered (LP) long-term (LT) noise energy value for current ISM is $lp_{noise,ism}(m) \geq 15$.

An output from the ISM classifier is thus an ISM importance parameter (one per ISM channel) which further serves as a driving parameter for setting the adapted bitrates for all ISMs, $brate'_{ism}(m)$, and adapted MASA total bitrate, $brate'_{MASA}$, in the combined format bitrate adaptation logic. It is noted that the OMASA inter-format bitrate adaptation is different from bitrates distribution between ISMs from clause 5.6.2.3 such that the adapted ISM total bitrate in OMASA, $brate'_{ISM}$, usually varies from frame to frame while it is constant in the ISM format coding.

The ISM importance class, $class_{ISM}$, is transmitted in the bitstream to the decoder where it serves as a driving parameter for setting the adapted bitrates for all ISMs, $brate'_{ism}(m)$, $m = 1, \ldots, M_{ISM}$, and adapted MASA total bitrate, $brate'_{MASA}$, at the decoder side. Thus, the same inter-format bitrate adaptation algorithm is used both at the encoder and the decoder.

In addition to the ISM pre-processing parameters (the importance classes $class_{ISM}(m)$), the inter-format bitrate adaptation logic depends also on the MASA pre-processing parameter. Specifically, low-pass filtered (LP) long-term (LT) noise energy value, $lp_{noise,MASA}$, from the MASA front pre-processing is employed and a difference between the $lp_{noise,MASA}$ of the scene ambience coded by MASA and $lp_{noise,ism}(m)$ of the ISM m is computed and compared to a threshold. Consequently, an ISM channel will be coded using the low bitrate core-coder coding mode if its background noise is masked by the MASA scene ambience audio. The inactive class ISM_INACTIVE is thus set under the condition of the following relation:

$$if(\ lp^{[-1]}_{noise,MASA} - lp_{noise,ism}(n) \geq \delta\ ) \quad \text{then} \quad class_{ISM} = \text{ISM\_INACTIVE} \tag{5.9-6}$$

while it is applied in a loop for all M_{ISM} ISMs and where the threshold $\delta = 30$. It is noted that the processing in the ISM format encoder and the MASA format encoder is performed sequentially and that the parameters of the current frame for the MASA part are not available when performing the inter-format bitrate adaptation. In order to get around of this limitation, a parameter from a previous frame are used instead which is indicated in equation (5.9-6) by the superscript $^{[-1]}$.

The bitrate adaptation logic then uses the following steps to assign a higher bitrate to an ISM with a higher importance and a lower bitrate to an ISM with a lower importance:

Step 1: Set nominal $brate_{ism}(m)$ bitrates for all M_{ISM} coded ISMs as the nominal $brate_{ISM}$ bitrate divided by the number of ISMs, i.e.

$$brate_{ism}(m) = brate_{ISM}/M_{ISM} \quad \text{for} \quad m = 1, \ldots, M_{ISM} \tag{5.9-7}$$

while the nominal ISM total bitrate, $brate_{ISM}$, is set in the OMASA configuration module from Figure , it is constant at one IVAS total bitrate and it represents a nominal bitrate around which the adapted ISM total bitrate, $brate'_{ISM}$, fluctuates.

Step 2: In frames classified as $class_{ISM}(m) = $ ISM_INACTIVE, a constant low bitrate $B_{VAD0} = 2450$ bps is assigned and a low-rate core-coder mode is selected for coding this frame.

Step 3: In frames classified as $class_{ISM}(m) = $ ISM_LOW_IMP, the nominal value of $brate_{ism}(m)$ is adapted using the following relation:

$$brate'_{ism}(m) = \gamma_{low} \cdot brate_{ism}(m) \tag{5.9-8}$$

where the weighting constant γ_{low} is set to a value lower than 1.0, see Table 5.9-3.

Step 4: In frames classified as $class_{ISM}(m)$ = ISM_MEDIUM_IMP, the nominal value of $brate_{ism}(m)$ is adapted using the following relation:

$$brate'_{ism}(m) = \gamma_{med} \cdot brate_{ism}(m) \quad (5.9\text{-}9)$$

where the weighting constant γ_{med} is set to a value higher than γ_{low}, see Table 5.9-3.

Step 5: In frames classified as $class_{ISM}(m)$ = ISM_HIGH_IMP, the nominal value of $brate_{ism}(m)$ is adapted using the following relation:

$$brate'_{ism}(m) = \gamma_{high} \cdot brate_{ism}(m) \quad (5.9\text{-}10)$$

where the weighting constant γ_{high} is usually set to a value higher than 1.0 and higher than γ_{med}, see Table 5.9-3.

Step 6: The adapted bitrate $brate'_{ism}(m)$ is checked against the minimum and the maximum threshold supported by the codec for a particular configuration (it is dependent for example on the core-coder internal sampling rate, coded audio band-width, etc.).

Step 7: Repeat the steps 2 to 6) for every ISM channel $m = 1, \ldots, M_{ISM}$.

Once the adapted bitrates $brate'_{ism}(m)$ are computed for all M_{ISM} ISM channels, the inter-format bitrate adaptation module computes the adapted ISM total bitrate, $brate'_{ISM}$, using the following relation:

$$brate'_{ISM} = \sum_{m=1}^{M_{ISM}} brate'_{ism}(m). \quad (5.9\text{-}11)$$

Next, the core-coder configuration module sets high-level parameters of the core-coders, for example the internal sampling rate or coded audio band-width based on the nominal ISM total bitrate, $brate_{ism}(m)$, while the adapted ISM bitrate, $brate'_{ism}(m)$, further specifies the available bit-budget for individual SCE core-coder parts (see clause 5.2.2.3.2.1).

Finally, the adapted MASA total bitrate $brate'_{MASA}$ is computed using the following relation:

$$brate'_{MASA} = brate_{IVAS} - brate'_{ISM}. \quad (5.9\text{-}12)$$

The following Table 5.9-3 then lists values of weighting constants $\gamma_{low}, \gamma_{med}, \gamma_{high}$. It can be seen that their values depend on the nominal (initial) ISM total bitrate (which depends on the IVAS total bitrate) and the number of separately coded objects.

Table 5.9-3: Weighting constant values used in OMASA bitrate adaptation

	γ_{low}	γ_{med}	γ_{high}
Set 1	0.8	1.2	1.4
Set 2	0.8	1.2	1.3
Set 3	0.85	1.15	1.3
Set 4	0.85	1.15	1.2
Set 5	0.8	1.0	1.2

In Table 5.9-3, five sets of weighting constants are defined while individual sets are applied in the following configurations:

Set 1: Default set.

Set 2: One OMASA coding mode at $brate_{IVAS} = 48000$ and $M_{ISM} = 3$, or $brate_{IVAS} = 48000$ and $M_{ISM} = 4$.

Set 3: Param OMASA coding mode except at $brate_{IVAS} = 32000$ and $M_{ISM} = 2$; Disc OMASA coding mode at $brate_{IVAS} = 24400$ and $M_{ISM} = 1$.

Set 4: Param OMASA coding mode at $brate_{IVAS} = 32000$ and $M_{ISM} = 2$.

Set 5: Disc OMASA coding mode at $brate_{IVAS} = 128000$ and $M_{ISM} = 4$, $brate_{IVAS} = 160000$ and $M_{ISM} = 4$, $brate_{IVAS} = 128000$ and $M_{ISM} = 3$, or $brate_{IVAS} = 48000$ and $M_{ISM} = 2$.

5.9.9 OMASA bitrate switching

OMASA format supports bitrate switching between any two IVAS bitrates from the range 13.2 kbps to 512 kbps. At the beginning of encoding of every frame, it is verified whether the current IVAS total bitrate R_{IVAS} equals to that of the previous frame $R_{IVAS}^{[-1]}$. If this is not the case, the OMASA mode selection is triggered according to Table 5.9-1 and the following steps are performed:

- Setting of the current OMASA coding mode and number of the transport channels.
- Resetting of the ISM total bitrate and MASA total bitrate to their initial (nominal) values according to the values from Table 5.9-1.
- Reconfiguration of MASA encoder.
- Reconfiguration of ISM encoder.
- Reconfiguration of SCE core-coders related to ISM coder part. If the number of ISM SCE core-coders in the current frame is higher than the number of ISM SCE core-coders in the previous frame, new memory handle instances are allocated and initialized. On the other hand, if the number of ISM SCE core-coders in the current frame is lower than the number of ISM SCE core-coders in the previous frame, obsolete memory handle instances are deallocated.
- Deallocation of the OMASA memory handle when there is a switching from Rend/One/Param OMASA mode to the Disc OMASA mode. Allocation and initialization of the OMASA memory handle when there is a switching from the Disc OMASA mode to Rend/One/Param OMASA mode. Reconfiguration of the OMASA memory handle in other OMASA mode switching scenarios.

5.9.10 OMASA bitstream structure

5.9.10 OMASA bitstream structure

Four bitstream structures are defined for the OMASA case, corresponding to each of the four coding modes. For the *Rend OMASA* coding mode the bitstream has the MASA bitstream structure presented in clause 5.5.6 with the modification indicated in clause 5.9.4.2 on the signalling of the number of input objects. For the *One MASA* coding mode the bitstream has the following order of components: IVAS format bits, the separated audio object data and spatial metadata, MASA transport channels, MASA metadata in which it has been inserted the number of objects and their importance flags. The *Param OMASA* coding mode has the following order of components in the bitstream: IVAS format bits, the separated object audio content, the MASA transport channels, MASA metadata including the ISM energy ratios and MASA-to-total energy ratios, the importance of audio objects, the index of separated audio object, and number of input audio objects. For the *Disc OMASA* coding mode, the bitstream is formed by: IVAS format bits, ISM bitstream, MASA transport channels, MASA metadata, number of input audio objects. In all coding modes except *Rend OMASA* the number of input audio objects is at the end of the bitstream.

5.10 EVS-compatible mono audio operation

The IVAS codec supports mono operation with EVS compatibility by incorporating EVS functionality in a bit-exact manner. The EVS-compatible mono audio encoder operation is described in clause 5 of [3].

In addition, stereo signal downmix for EVS mono coding is supported by the IVAS codec as described in detail in clause 5.11.

5.11 Stereo downmix operation for EVS mono coding

5.11.1 Overview

This downmix process aims to produce a mono signal for keeping the interoperability with EVS mono coding without additional delay. Depending on the characteristics of the input signal, one operation mode out of two types of downmix tools is selected for every frame. One mode is a simple weighted sum of two channels based on phase-only correlation

(clause 5.11.2), and the other is a sum of two channels modified by phase compensation (clause 5.11.3). The mode selection rule and the handling of mode switching are described in clauses 5.11.4 and 5.11.5, respectively.

5.11.2 Phase-only correlation (POC) mode

5.11.2.1 Overview

This tool generates a downmixed monaural signal by an adaptive weighted sum of two channels of input signal. This is achieved by detecting the preceding channel based on an efficient calculation of ITD (inter-channel time difference). According to the detected time difference between the two channels, a mixed monaural signal is produced with a larger weight for the preceding channel signal than that for another channel signal.

5.11.2.2 ITD analysis

5.11.2.2.1 Cross talk adding procedure in the frequency domain

The polarity of the ITD, which can be efficiently derived through a phase-only spectrum in the frequency domain shows the preceding channel. Channel 1 and 2 signals in the time domain are denoted as $s_1(n)$ and $s_2(n)$ at n in a frame. And frequency domain spectrums corresponding to these signals are represented by $S_1^{FFT}(k)$, $S_2^{FFT}(k)$, at frequency bin of k with time domain sine window $w(n)$ and frame length L. Note that frame length is corresponding to samples for 20 ms and 960 when 48 kHz.

$$S_i^{FFT}(k) = \frac{1}{\sqrt{L}}\sum_{n=0}^{L-1} w(n+1)s_i(n+1)e^{-j\frac{2\pi kn}{L}} \quad (5.11\text{-}1)$$

In order to obtain robust results for various input signals, an intentional cross talk process is applied. This is achieved by adding each channel signal and another channel signal with one-sample delayed and multiplied by a small value constant ε(=2π/L) in the time-domain. The modified (with intentional cross talk) complex spectrums $S_1^{\#}(k)$, $S_2^{\#}(k)$ are equivalently generated by adding another channel spectrum multiplied by ε and phase rotation of $e^{-j\frac{2\pi}{L}k}$ as

$$S_1^{\#}(k) = S_1^{FFT}(k) + \varepsilon S_2^{FFT}(k)\cdot e^{-j\frac{2\pi}{L}k} \quad (5.11\text{-}2)$$

$$S_2^{\#}(k) = S_2^{FFT}(k) + \varepsilon S_1^{FFT}(k)\cdot e^{-j\frac{2\pi}{L}k} \quad (5.11\text{-}3)$$

Using the modified complex spectrum $S_1^{\#}(k)$ and $S_2^{\#}(k)$, inter-channel phase difference complex spectrum $\phi(k)$ is defined as,

$$\phi(k) = \frac{S_1^{\#}(\mathrm{k})/|S_1^{\#}(\mathrm{k})|}{S_2^{\#}(\mathrm{k})/|\,S_2^{\#}(\mathrm{k})|} \quad (5.11\text{-}4)$$

5.11.2.2.2 Approximated phase spectrum

Calculation of $\phi(k)$ needs operations of square root and division for each spectral bin. Instead, $\phi(k)$ can be approximated by simple quantization of the angle of $\phi^*(k)$ by making use of the property that $\phi(k)$ has complex values on a unit circle. The angle of $\phi^*(k)$ is same as that of inter-channel difference spectrum $Y(k)$ which is derived by multiplication of $S_1^{\#}(k)$ and conjugate of $S_2^{\#}(\mathrm{k})$, $\overline{S_2^{\#}(\mathrm{k})}$ as

$$Y(k) = S_1^{\#}(k)\cdot \overline{S_2^{\#}(k)} \quad (5.11\text{-}5)$$

$\phi(k)$ can be approximated by the polarity of $u(k)$, $sign(u(k))$ and the polarity of $v(k)$ ($u(k) = real(Y(k))$ and $v(k) = imag(Y(k))$) and quantized values of $\phi(k)$, $|u_Q(k)|$ and $|v_Q(k)|$ shown in Table 5.**11-1**, where *real*(x) and $imag(x)$ mean real and imaginary part of complex value of x. The weighted and quantized inter-channel phase difference complex spectrum of $\phi(k)$, is produced as $\phi^*(k)$, where $w(k) = sin(k\pi/L)\,(1 - 0.75k/L)$.

$$\phi^*(k) = w(k)\phi_Q(k) = w(k)(sign(u(k))\cdot|u_Q(k)|,\ sign(v(k))\cdot|v_Q(k)|) \quad (5.11\text{-}6)$$

The lower resolution is applied for lower frequency bins due to the small weight values, which allow larger approximation errors due to quantization. For the lowest frequency bins up to the 10th, a smaller value is used for the quantized phase angle, while for other frequency regions, quantized angle values have equal spacing.

When Search intensity *R* equals 0 (0 bit in case of transmission of code), $(|u(k)|, |v(k)|)$ can be looked up from Table 5.**11-1** for the same quadrant of $Y(k)$. In case *R* is positive, the nearest angle region of $(|u_Q(k)|, |v_Q(k)|)$ with that of $(|u(k)|, |v(k)|)$ is detected among 2^R regions by *R* times binary search with thresholds, *Th*, such as $|u(k)| > |v(k)|$ or $|u(k)| > |v(k)| \cdot Th$.

Table 5.11-1: Quantized values $|u_Q(k)|$, $|v_Q(k)|$ on a unit circle. (in case only the real and imaginary parts are positive)

Frequency bin index *k*	Search intensity *R*	Quantized phase spectrum $(|u_Q(k)|, |v_Q(k)|)$	Quantized phase In angle value [rad]
$k < 10$ or w(k) = 0	0	(0.866, 0.5)	0.5236
$10 \le k < L/16$	0	(0.7071, 0.7071)	$\pi/4$
$L/16 \le k < L/8$	1	(0.923880, 0.382683)	$\pi/8$
		(0.382683, 0.923880)	$3\pi/8$
others	2	(0.980785, 0.195090)	$\pi/16$
		(0.831470, 0.555570)	$3\pi/16$
		(0.555570, 0.831470)	$5\pi/16$
		(0.195090, 0.980785)	$7\pi/16$

5.11.2.2.3 Calculation of phase only correlation

Time domain inter-channel cross correlation between channel 1 and 2 at a time difference of τ is defined as $\psi(\tau)$ and derived by inverse Fourier transformation of the weighted phase spectrum $\phi^*(k)$. Squared inter-channel correlation $C(\tau)$ and filtered inter-channel correlation $P(\tau)$ is derived from $C(\tau)$ as follows,

$$\mathrm{C}(\tau) = \psi^2(\tau) = \left(\frac{1}{\sqrt{L}}\sum_{k=0}^{L-1} \phi^*(k) e^{j\frac{2\pi k\tau}{L}}\right)^2 \tag{5.11-7}$$

$$P(\tau) = 1.25 \cdot C(\tau) - 0.125 \cdot (C(\tau+1) + C(\tau-1)) \tag{5.11-8}$$

$$\begin{cases} P(L/2 + ITD_{1,prev}) = 0.79 \cdot P_{prev}(L/2 + ITD_{1,prev}) + 0.21 \cdot P(L/2 + ITD_{1,prev}) \\ P(L/2 - ITD_{2,prev}) = 0.79 \cdot P_{prev}(L/2 - ITD_{2,prev}) + 0.21 \cdot P(L/2 - ITD_{2,prev}) \\ P(\tau) = 0.78 \cdot P_{prev}(\tau) + 0.22 \cdot P(\tau) \ when\ \tau \neq L/2 + ITD_{1,prev}\ and\ \tau \neq L/2 - ITD_{2,prev} \end{cases} \tag{5.11-9}$$

Here, $P_{prev}(\tau)$ is the filtered inter-channel correlation of the previous frame. $ITD_{1,prev}$ (resp. $ITD_{2,prev}$) is the previous frame's candidate of *ITD*, assuming the channel 1 (resp. 2) is preceding.

5.11.2.2.4 Calculation of ITD and weighting coefficients

To find the most preferable *ITD*, two candidates of ITD_1^{CAND} (negative, assuming the channel 1 is preceding) and ITD_2^{CAND} (positive, the channel 2 is preceding) are searched, which give the largest value of inter-channel correlation $P(L/2 - ITD_1^{CAND})$ and $P(L/2 + ITD_2^{CAND})$ within the rage of $0 \le |ITD_1^{CAND}|, |ITD_2^{CAND}| \le limit$. The *limit* is the maximum time difference in samples corresponding to the time difference of 5.16 ms; namely *limit* is 247 when the sampling rate is 48 kHz. Then, the updating process decides whether to update the probable *ITD*s, ITD_1 and ITD_2 by ITD_1^{CAND} and ITD_2^{CAND}. *ITD* is finally selected from ITD_1 and ITD_2 based on the quality factors Q_1 , Q_2 of the peaks of associated inter-channel correlation values, depending on the activities of the previous frame, such as ON_1 and OFF_1 . For the channel 1, $Q_1$and ITD_1 are calculated as in the following process. For the channel 2, the same process is used to obtain Q_2 and ITD_2

Start of processing block for calculation of $Q_1$and ITD_1.

Q_2 and ITD_2, can be calculated by replacing all suffixes of "1" with "2" in this processing block.

Peak range PR_1 is defined as $PR_1 = |ITD_1^{CAND}| + limit/32$. Peak width PW_1, which is the accumulated number of samples where the Boolean conditions (P is larger than $0.38 \cdot$ peak) are true, and cumulative peak value Q_1^{CUMU} are defined as

$$\begin{aligned} PW_1 = &\sum_{i=1}^{PR_1} \left(P(L/2 - ITD_1^{CAND} + i) > 0.38 \cdot P(L/2 - ITD_1^{CAND}) \right) \\ &+ \sum_{i=1}^{PR_1} \left(P(L/2 - ITD_1^{CAND} - i) > 0.38 \cdot P(L/2 - ITD_1^{CAND}) \right) \end{aligned} \tag{5.11-10}$$

and

$$Q_1^{CUMU} = P(L/2 - ITD_1^{CAND}) + \sum_{i=1}^{PR_1} (P(L/2 - ITD_1^{CAND} + i) + P(L/2 - ITD_1^{CAND} - i))$$ (5.11-11)

Using the previous frame's value of PW_1, $PW_{1,prev}$, PW_1 is updated as

$$PW_1 = 0.875 \cdot PW_{1,prev} + 0.125 \cdot PW_1$$

and then $\delta = 0.25 \cdot PW_1/L$.

Q_1 is derived as a relative difference between the correlation peak and the average correlation around it when the peak is large enough compared to the peak width.

$$Q_1 = (1 - (Q_1^{CUMU}/(2 \cdot PR_1 + 1) + (\delta \cdot PW_1))/(P(L/2 - ITD_1^{CAND}) + (\delta \cdot PW_1))) \quad (5.11\text{-}12)$$

The following process decides whether to update ITD_1 by ITD_1^{CAND}. If ON_1 $equals$ 1, namely the channel 1 was active (preceding) in the previous frame, ITD_1 and $peakQ_1$ is of the current frame is updated as follows;

`if(` $Q_1 > (0.3 - 0.2 \cdot |ITD_1^{CAND}|)/limit \cdot peakQ_1$ `)` $\{ITD_1 = 0,\ ON_1 = 0, Q_1 = 0, peakQ_1 = 0\}$

`else if (` $Q_1 > 1.25 \cdot peakQ_1$ `)` $\{ITD_1 = ITD_1^{CAND}$ and $peakQ_1 = \max(peakQ_1,\ Q_1)\}$

`else` {keep ITD_1 used in the previous frame}

If ON_1 $equals$ 0, namely the channel 1 was not active (not preceding) in the previous frame, ITD_1 and Q_1 is updated as follows.

`if(` $Q_1 < (0.75 - 0.2 \cdot |ITD_1^{CAND}|)/limit$ `)` $\{ITD_1 = 0$ and $Q_1 = 0\}$

`else` $\{ITD_1 = ITD_1^{CAND}$ and $ON_1 = 1\}$

End of processing for calculation of $Q_1$and ITD_1.

Depending on Q_1 , Q_2 and previous ITD_{prev}, *ITD* for this frame is set according to the rule as in Table 5.11-2. Note that $OFF_{1,prev}$ and $OFF_{2,prev}$ mean that the channel 1 or channel 2 were not active in the frame before the previous.

Table 5.11-2: Determination of ITD

Primary condition	Secondary condition	decision
If(ON_1 & $OFF_{1,prev}$ & ON_2 & $OFF_{2,prev}$)	If ($Q_1 > Q_2$)	$ITD = ITD_1$
	else	$ITD = ITD_2$
else if (ON_1 & $OFF_{1,prev}$ & ($Q_1 > (Q_2 - 0.1)$))		$ITD = ITD_1$
else if (ON_2 & $OFF_{2,prev}$ & ($Q_2 > (Q_1 - 0.1)$))		$ITD = ITD_2$
else if ($Q_1 > (Q_2 + 0.25)$)		$ITD = ITD_1$
else if ($Q_2 > (Q_1 + 0.25)$)		$ITD = ITD_2$
else if (ITD_{prev} ==0)		$ITD = 0$
else	If ($ITD_{prev} > 0$)	$ITD = ITD_1$
	else	$ITD = ITD_2$

Downmix weight *h* is determined by a parameter defined as *conf* and *ITD*, depending on the preceding channel and *conf* of the previous frame, $conf_{prev}$.

$$conf = \sqrt{(|Q_1 - Q_2|)}$$ (5.11-13)

$$conf = 0.78 \cdot conf_{prev} + 0.22 \cdot conf$$ (5.11-14)

$$h = \begin{cases} 0.5 \cdot (1 - conf) & when\ ITD > 0 \\ 0.5 & when\ ITD = 0 \\ 0.5 \cdot (1 + conf) & when\ ITD < 0 \end{cases}$$ (5.11-15)

We can see that if Q_1 is larger than Q_2, *ITD* is negative, and the channel 1 signal is likely preceding to the channel 2. As a result, the weighting to the channel 1, h is larger than 0.5, and a downmixed signal is generated with a larger weight for the preceding channel signal than that for another channel signal. Moreover, in case of *ITD* is negative, the larger value of inter-channel correlation, the larger value of Q_1 and the larger value of *conf* and h are derived. In case of *ITD* is positive, the larger value of inter-channel correlation, the larger value of Q_2, the larger value of *conf* and the smaller value of h are derived. For both cases, the larger the inter-channel correlation, the larger the mixing weight is assigned for the preceding channel signals.

5.11.2.3 Downmix process

The first mixing process is executed as follows with the window $w(n) = sin^2(\pi n/L)$ and $w(L/2) = 1$ and previous frame's value of h , h_{prev} .

$$d(n) = \begin{cases} (h_{prev} + (h - h_{prev})w(n))s_1(n) + (1 - (h_{prev} + (h - h_{prev})w(n)))\, s_2(n) & 0 \le n < L/2 \\ hs_1(n) + (1 - h)s_2(n) & L/2 \le n < L \end{cases} \quad (5.11\text{-}16)$$

$d(n)$ is modified by the weighting coefficients g_1 and g_2 derived from the energy E_1 for $s_1(n)$, E_2 for $s_2(n)$ and E_D for $d(n)$. The following trapezoidal windowed frame energy E_y for signal $y(n)$, namely, either $s_1(n), s_2(n),$ *or* $d(n)$ is used.

$$E_y = \sum_{n=0}^{L/16-1} w^2(n)y^2(n) + \sum_{i=L/16}^{L-L/16-1} y^2(n) + \sum_{i=L-L/16}^{L-1} w^2(n)y^2(n) \quad (5.11\text{-}17)$$

$$E_y = 0.25 \cdot E_{y.prev} + 0.75 \cdot (E_y/L)\,, \quad (5.11\text{-}18)$$

where $E_{y.prev}$ is the previous frame's value of E_y and

$$w(i) = \begin{cases} 16 \cdot (0.5 + i)/L & 0 \le i < L/16 \\ 16 \cdot (L - 0.5 - i)/L) & L - L/16 \le i < L \end{cases} \quad (5.11\text{-}19)$$

For enhancing the weight for the preceding channel, weighting coefficients g_1 for $s_1(n)$ and g_2 for $s_2(n)$ are derived from the frame energy as in Table 5.11-3. Note that $g_{1,prev}$, and $g_{2,prev}$ are previous frame's values of g_1 and g_2

Table 5.11-3: Determination of g_1 and g_2 with $\varepsilon = 1024$

State	Energy	g_1	g_2
$g_{2,prev} = 0$	$(4 \cdot E_1 > E_2)$	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_1 + \varepsilon)}}, 0.0)$	0.0
	$(4 \cdot E_1 \le E_2)$	0.0	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_2 + \varepsilon)}}, 0.0)$
$g_{2,prev} \neq 0$	$(E_1 > 4 \cdot E_2)$	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_1 + \varepsilon)}}, 0.0)$	0.0
	$(E_1 \le 4 \cdot E_2)$	0.0	$max(1. - \sqrt{\frac{(E_D + \varepsilon)}{(E_1 + \varepsilon)}}, 0.0)$

Downmixed signal $d(n)$ is further modified as

$$d(n) = \begin{cases} d(n) + (g_{1,prev} + (g_1 - g_{1,prev})w(n))s_1(n) \\ \quad + (g_{2,prev} + (g_2 - g_{2,prev})w(n))s_2(n) & \text{when } 0 \le n < L/16 \\ d(n) + g_1 s_1(n) + g_2 s_2(n) & \text{when } L/16 \le n < L \end{cases} \quad (5.11\text{-}20)$$

Finally, $d(n)$ is a monaural signal used to input the EVS encoder, which generates interoperable bit streams for the EVS decoder.

5.11.3 Phase compensation (PHA) mode

5.11.3.1 Overview

Phase compensation in time domain is used to modify the input stereo signal. Left and right channels are filtered adaptively by short impulse responses approximating phase differences. Two phase compensation (PHA) sub-modes are defined (denoted "IPD" or "$IPD2$") and adaptively switched on a frame basis. Note that the "$IPD2$" includes two variants (IPD/2 and IPD/4) that are defined in clause 5.11.3.5.3.

In the following the PHA sub-mode in the current frame is denoted M_{PHA}^{curr} and can take values indicated here as strings for the sake of readability: "IPD" or "$IPD2$".

5.11.3.2 Inter spectral difference

The inter spectral difference (ISD) is defined as:

$$ISD(k) = \frac{|S_1(k) - S_2(k)|}{|S_1(k) + S_2(k)|} \quad (5.11\text{-}21)$$

This criterion is used to detect frames where out of phase cancelation may occur. The relative number of bins (resp. N_{ISD}^h and N_{ISD}^l) which verify $ISD(k) > 1.3$ and $ISD(k) < \sqrt{0.9}$ is determined in the 50-8000 Hz frequency range. To minimize complexity, N_{ISD}^h and N_{ISD}^l are computed as follows:

- Initialisation: $N_{ISD}^h = 0;\ N_{ISD}^l = 0$
- Loop: for k = 1 to 160: $N_{ISD}^h \leftarrow N_{ISD}^h + 1$ if $ISD(k)^2 > 1.69$; $N_{ISD}^l \leftarrow N_{ISD}^l + 1$ if $ISD(k)^2 < 0.9$
- Normalisation: $N_{ISD}^h \leftarrow N_{ISD}^h/160$; $N_{ISD}^l \leftarrow N_{ISD}^l/160$;

Note that N_{ISD}^h and N_{ISD}^l can be summarized as:

$$N_{ISD}^h(t) = \frac{total\ number\ of\ bins\ where\ \ ISD(k) > 1.3}{total\ number\ of\ bins} \quad (5.11\text{-}22)$$

$$N_{ISD}^l(t) = \frac{total\ number\ of\ bins\ where\ \ ISD(k) < \sqrt{0.9}}{total\ number\ of\ bins} \quad (5.11\text{-}23)$$

The quantity N_{ISD}^h is smoothed by exponential averaging as follows:

$$\bar{N}_{ISD}^h \leftarrow \bar{N}_{ISD}^h * 0.95 + N_{ISD}^h * 0.05 \quad (5.11\text{-}24)$$

This criterion $\bar{N}_{ISD}^h$ gives the average percentage of bins for which a phase compensation would be desirable, and it is used for the PHA sub-mode selection (see clause 5.11.3.4).

Note that the quantity N_{ISD}^l is used in the phase compensation filter determination (see clause 5.11.3.5.3).

5.11.3.3 Inter-channel phase difference (IPD) and realigned inter-channel correlation (ICCr)

The discrete spectrum is divided into $K - 1$ equal subbands of 2 bins covering intervals $[k_b, k_{b+1} - 1]$, where $b = 1, .., K - 1$ and $K = 80, 160, 240$ at 16, 32 or 48 kHz.

The subband energy is given by:

$$E_1(b) = \sum_{k=k_b}^{k_{b+1}-1} |S_1^{FFT}(k)|^2 \quad (5.11\text{-}25)$$

$$E_2(b) = \sum_{k=k_b}^{k_{b+1}-1} |S_2^{FFT}(k)|^2 \quad (5.11\text{-}26)$$

The inter-channel phase difference (IPD) is defined in frequency domain for each frequency bin in the current frame as:

$$IPD(k) = \angle(S_1^{FFT}(k).\, S_2^{FFT}(k)^*) \quad (5.11\text{-}27)$$

where $\angle$ is the complex argument (angle) and * denotes the complex conjugate.

To reduce complexity, the IPD is determined with no explicit angle calculation and by grouping bin pairs in subbands. The IPD is therefore expressed in complex domain as the unit complex value:

$$\phi_{PHA}(b) = \frac{\sum_{k=k_b}^{k_{b+1}-1} S_1^{FFT}(k) S_2^{FFT}(k)^*}{\left|\sum_{k=k_b}^{k_{b+1}-1} S_1^{FFT}(k) S_2^{FFT}(k)^*\right|} \quad (5.11\text{-}28)$$

The value of $\phi_{PHA}(b)$ is smoothed by exponential averaging with a forgetting factor depending on the subband:

$$\bar{\phi}_{PHA}(b) \leftarrow \alpha(b)\bar{\phi}_{PHA}(b) + \big(1 - \alpha(b)\big)\phi_{PHA}(b) \quad (5.11\text{-}29)$$

where

$$\alpha(b) = \begin{cases} \alpha_{max}, & b < b_{low} \\ \alpha_{min}, & b > b_{high} \\ \alpha_{min} + \frac{b - b_{low}}{b_{high} - b_{low}}, & b_{low} < b < b_{high} \end{cases} \quad (5.11\text{-}30)$$

with $\alpha_{min} = 0.8576958985908941$ and $\alpha_{max} = 0.9440608762859234$. This quantity is further normalized to get a unit complex value:

$$\bar{\phi}_{PHA}(b) \leftarrow \frac{\bar{\phi}_{PHA}(b)}{\sqrt{\Re\big(\bar{\phi}_{PHA}(b)\big) + \Im\big(\bar{\phi}_{PHA}\big) + \epsilon}} \quad (5.11\text{-}31)$$

where $\Re(.)$ and $\Im(.)$ correspond to the real and imaginary parts of a complex value. Note that in the first 10 frames this smoothing of $\phi_{PHA}(b)$ is not applied, i.e. $\bar{\phi}_{PHA}(b) = \phi_{PHA}(b)$.

The phase spectrum $\bar{\phi}_{PHA}(b)$ is obtained after clamping the result (separately for real and imaginary parts), as follows:

$$\Re(\bar{\phi}_{PHA}(b)) \leftarrow \max(\min(\Re(\bar{\phi}_{PHA}(b)), 1), -1) \quad (5.11\text{-}32)$$

$$\Im(\bar{\phi}_{PHA}(b)) \leftarrow \max(\min(\Im(\bar{\phi}_{PHA}), 1), -1) \quad (5.11\text{-}33)$$

The realigned inter-channel correlation (ICCr) is determined as:

$$ICCr = \sqrt{\frac{\sum_{b=1}^{K-1}\sum_{k=k_b}^{k_{b+1}-1} S_1^{FFT}(k) S_2^{FFT}(k)^* e^{jIPD(b)}}{\left(\sum_{b=1}^{K-1}\sum_{k=k_b}^{k_{b+1}-1} |S_1^{FFT}(k)|^2\right)\left(\sum_{b=1}^{K-1}\sum_{k=k_b}^{k_{b+1}-1} |S_2^{FFT}(k)|^2\right) + \epsilon}} \quad (5.11\text{-}34)$$

The quantity ICC_r is smoothed by exponential averaging as follows:

$$\overline{ICCR}_r \leftarrow \overline{ICCR}_r * 0.7 + ICC_r * 0.3 \quad (5.11\text{-}35)$$

5.11.3.4 Selection rule for PHA sub-mode

The selection rule between IPD and IPD/2 sub-modes depends on $\bar{N}_{ISD}^h$:

- if $\bar{N}_{ISD}^h > 0.43$, the current PHA sub-mode is set to IPD ($M_{PHA}^{curr} = "IPD"$) with a hysteresis forcing one previous frame with the same decision.
- if $\bar{N}_{ISD}^h < 0.36$, the current PHA sub-mode is set to IPD/2 ($M_{PHA}^{curr} = "IPD2"$) with a hysteresis forcing one previous frame with the same decision.

5.11.3.5 Determination of filter taps

5.11.3.5.1 General

The filter taps of the previous frame (one set of index 0 or 1 for the left or right channel, resp.) are first updated:

$$h_i^{prev}(n) \leftarrow h_i^{curr}(n), i = 0,1 \quad (5.11\text{-}36)$$

The filer taps in the current frame $h_i^{curr}(n)$ are determined based on the selected sub-mode M_{PHA}^{curr}. A single filter $H(b)$ is first determined in complex frequency domain, before applying an inverse real FFT of length $2K$, windowing, and rescaling.

5.11.3.5.2 Filter taps for IPD

If the current sub-mode is IPD (i.e., $M_{PHA}^{curr} = "IPD"$), filter taps are determined only for the right channel with a filter corresponding in principle to:

$$H_2(k) = \frac{1}{\sqrt{2}} e^{IPD(k)} \quad (5.11\text{-}37)$$

if the filter was defined in frequency domain for each bin. In practice, the filter is shortened using subbands (bin pairs) and the IPD is not explicitly defined, the phase spectrum is directly in terms of real and imaginary parts.

In the IPD sub-mode, the filter for the left :

$$H_1(k) = \frac{1}{\sqrt{2}} \quad (5.11\text{-}38)$$

filter taps for the left channel are not defined:

$$h_1^{curr} = \emptyset \quad (5.11\text{-}39)$$

In practice, the factor $\frac{1}{\sqrt{2}}$ can be omitted in filter tap calculation and applied at the filter stage (see clause 5.11.3.6). To minimize complexity, only one set of filter taps is determined.

More specifically, the filter for the right channel is first estimated in K subbands as follows:

- $H(b) = 1$ for $b = 0, K$ (DC and Nyquist bins)
- $H(b) = \bar{\phi}_{PHA}(b)$ for $b = 1, \dots, K-1$
- Furthermore, the inter-channel level difference (ILD) between left and right channels is used by subband to count the number of subbands N_{ILD} that verify the ILD condition: $E_1(b)/E_2(b) > 15$ or $E_2(b)/E_1(b) > 15$. If $N_{ILD} > 0.1 * K$, the filter value is overwritten as $H(t, b) \leftarrow 1$ in each subband of index b that satisfy this ILD condition.

Then, the impulse response h_2^{curr} is obtained by inverse FFT of length $2K$:

$$h_2^{curr}(n) = FFT^{-1}\{H(b)\} \quad (5.11\text{-}40)$$

5.11.3.5.3 Filter taps for IPD2

If the current sub-mode is IPD2 (i.e., $M_{PHA}^{curr} = "IPD2"$), the filter taps are determined for both left and right channels. In principle, they are determined as:

$$H_1(k) = \frac{1}{\sqrt{2}} e^{-IPD(k)/\xi} \quad (5.11\text{-}41)$$

$$H_2(k) = \frac{1}{\sqrt{2}} e^{IPD(k)/\xi} \quad (5.11\text{-}42)$$

where $\xi = 2$ or 4 (corresponding to IPD/2 or IPD/4). In practice, the factor $\frac{1}{\sqrt{2}}$ can be omitted in filter tap calculation and applied at the filter stage (see clause YY). To minimize complexity, only one set of filter taps is determined, the other step is obtained by folding with no extra inverse FFT.

The actual filter tap determination is defined as follows:

If $ICCR_r^{smooth} < 0.75$ or ($ICCR_r^{smooth} < 0.85$ and $M_{PHA}^{prev} = "IPD2"$

- $H(b) = 1$ for $b \in \{0\} \cup [K_{IPD}^{max}, \dots, K-1]$ with $K_{IPD}^{max} = 50$
- Compute filter for IPD/2:

$$H(b) = \sqrt{\frac{1+\mathcal{R}\left(\bar{\phi}_{PHA}(b)\right)}{2}} + j\sqrt{\frac{1-\mathcal{R}\left(\bar{\phi}_{PHA}(b)\right)}{2}} sign\left(\Im\left(\bar{\phi}_{PHA}(b)\right)\right),\ b = 1, \dots, K-1 \quad (5.11\text{-}43)$$

- If $N_{ISD}^{l} > 0.5$ compute filter for IPD/4:

$$H(b) \leftarrow \sqrt{\frac{1+\mathcal{R}\left(H(b)\right)}{2}} + j\sqrt{\frac{1-\mathcal{R}\left(H(b)\right)}{2}} sign\left(\Im\left(H(b)\right)\right),\ b = 1, \dots, K-1 \quad (5.11\text{-}44)$$

- Furthermore, the inter-channel level difference (ILD) between left and right channels is used by subband to count the number of subbands N_{ILD} that verify the ILD condition: $E_1(b)/E_2(b) > 15$ or

$E_2(b)/E_1(b) > 15$. If $N_{ILD} > 0.1 * K$, the filter value is overwritten as $H(t,b) \leftarrow 1$ in each subband of index b that satisfy this ILD condition.

Else:

Force POC downmix

Then, the impulse response h_2^{curr} for the right channel is obtained by inverse FFT of length $2K$:

$$h_2^{curr}(n) = FFT^{-1}\{H(b)\} \quad (5.11\text{-}45)$$

The impulse response h_1^{curr} for the left channel is derived by folding:

$$h_1^{curr}(0) = h_2^{curr}(0) \quad (5.11\text{-}46)$$

$$h_1^{curr}(n) = h_2^{curr}(2K - n), n = 1, \dots, 2K - 1 \quad (5.11\text{-}47)$$

5.11.3.5.4 Truncation and normalization

The filter for each channel is truncated by windowing:

$$\breve{h}_i(n) = w(n).\tilde{h}_i(n) \quad (5.11\text{-}48)$$

where the window is half of a Hanning window:

$$w(n) = \begin{cases} 1 & n = 0 \\ 1.8 & n = 1, .., L_{PHA} - L_{PHA}^{trans} - 1 \\ 0.5 * \left(1 + cos\frac{2\pi(l+1)}{2P+1}\right) * 1.8 & n = L_{PHA} - L_{PHA}^{trans}, \dots, L_{PHA} - 1 \end{cases} \quad (5.11\text{-}49)$$

where L_{PHA} is the filter length depending on the sampling frequency: L_{PHA} =24, 48, 48 at 16, 32, 48 kHz (respectively); L_{PHA}^{trans} is the length of transition defined as $L_{PHA}^{trans} = \lfloor L_{PHA}/20 \rfloor$ and $\lfloor . \rfloor$ is the rounding to integer towards 0.

Finally, the filter in each channel is normalized as follows:

$$\check{h}_i(n) = \frac{\breve{h}_i(n)}{2\|\breve{h}_i(n)\|}, n = 0, \dots, L_{PHA} - 1 \quad (5.11\text{-}50)$$

where $\|.\|$ is the L2 vector norm.

5.11.3.6 Downmix process

The two input channels are filtered by $\check{h}_i(n), i = 1,2$ separately. Note that if filter taps are not defined in one channel, i.e. $\check{h}_i(n) = \emptyset$, this filter is not applied to this specific channel.

The downmix in PHA mode is obtained as follows by :

- Initialization:

$$d_{PHA}^{fad}(n) = 0, n = 0, \dots, L_{PHA} - 1 \quad (5.11\text{-}51)$$

$$d_{PHA}^{curr}(n) = 0, n = 0, \dots, L - 1 \quad (5.11\text{-}52)$$

- Repeat for two channels $i = 1,2$:
  - The previous end of frame $x_i^{mem}(n), n = 0, .., L_{PHA} - 1$ for each channel is concatenated with the current frame:

$$\tilde{x}_i(n) = [x_i^{mem}(0), \dots, x_i^{mem}(L_{PHA} - 1), x_i(0), \dots x_i(L - 1)] \quad (5.11\text{-}53)$$

  - If filter taps $\check{h}_i(n)$ in the previous frame are defined (i.e., $\check{h}_i^{prev}$ is not $\emptyset$), the crossfade region is filtered with previous filter taps for $n = 0, \dots, L_{PHA} - 1$:

$$d_{PHA}^{fad}(n) \leftarrow d_{PHA}^{fad}(n) + \begin{cases} \frac{1}{\sqrt{2}}\sum_{p=0}^{L_{PHA}-1} \check{h}_i^{prev}(p).\tilde{x}_i(n + L_{PHA} - p) & \text{if } \check{h}_i^{prev} \text{ is not } \emptyset \\ \frac{1}{\sqrt{2}}\tilde{x}_i(n + L_{PHA}) & \text{if } \check{h}_i^{prev} \text{ is } \emptyset \end{cases} \quad (5.11\text{-}54)$$

- The channel is filtered with current filter taps for $n = 0, \dots, L-1$:

$$d_{PHA}^{curr}(n) \leftarrow d_{PHA}^{curr}(n) + \begin{cases} \frac{1}{\sqrt{2}} \sum_{p=0}^{L-1} \breve{h}_i(p).\, \tilde{x}_i(n-p), 0 \le l < L & \text{if } \breve{h}_i \text{ is not } \emptyset \\ \frac{1}{\sqrt{2}} \tilde{x}_i(n + L_{PHA}) & \text{if } \breve{h}_i \text{ is } \emptyset \end{cases} \quad (5.11\text{-}55)$$

The downmix signal is crossfaded with the memory from the previous frame:

$$d_{PHA}(n) \leftarrow d_{PHA}(n) + \begin{cases} \beta_{PHA}(n) d_{PHA}^{curr}(n) + \big(1 - \beta_{PHA}(n)\big) d_{PHA}^{fad}(n), & n = 0, \dots, L_{PHA} - 1 \\ d_{PHA}^{curr}(n) & n = L_{PHA}, \dots, L-1 \end{cases} \quad (5.11\text{-}56)$$

where the fading is linear:

$$\beta_{PHA}(n) = \frac{n+1}{L_{PHA}+1} \quad (5.11\text{-}57)$$

5.11.4 Mode selection rule

The downmix mode (POC or PHA) in the current frame is initially set to the mode of the previous frame.

If $|ITD| > \tau_{ITD}$ (where $\tau_{ITD} = 55, 19, 29$ at respectively 16, 32, 48 kHz)

The mode is set to POC with a hysteresis of 1 frame

Else

The mode is set to PHA with a hysteresis of 1 frame

If $F_{transient} = 1$ or $E_1 > 1000 * E_2$ or $E_2 > 1000 * E_2$ or forced POC

The mode is set to POC (and the hysteresis state is reset)

5.11.5 Handling of mode switching

In a transition frame, i.e, when the downmix mode in the current frame is not the same as in the previous frame, the downmix output is crossfaded between the output of the selected mode and the output of the other mode.

If the current mode is POC, the downmix signal is computed for $n = 0, \dots, L-1$ by selecting the output of the downmix:

$$d(n) = \begin{cases} d_{POC}(n), & \text{if } M_{dmx} = POC \\ d_{PHA}(n), & \text{if } M_{dmx} = PHA \end{cases} \quad (5.11\text{-}58)$$

If the current mode M_{dmx} is different from the previous mode, a crossfading is applied for $n = 0, \dots, L_{fad} - 1$:

$$d(n) \leftarrow \begin{cases} \beta_{dmx}(n) d(n) + \big(1 - \beta_{dmx}(n)\big) d_{PHA}(n) & \text{if } M_{dmx} = POC \\ \beta_{dmx}(n) d(n) + \big(1 - \beta_{dmx}(n)\big) d_{POC}(n) & d_{PHA}(n) \end{cases} \quad (5.11\text{-}59)$$

where

$$\beta_{dmx}(n) = \frac{n+1}{L_{fad}+1}, n = 0, \dots, L_{fad} - 1 \quad (5.11\text{-}60)$$

and L_{fad} is the fading length equal to 20 ms.

6 Functional description of the decoder

6.1 Decoder overview

The IVAS codec's decoder processes the received bitstream and outputs the audio channels in one of the following formats: mono, stereo, objects (ISM), multichannel, SBA (FOA, HOA2, HOA3), MASA, loudspeaker rendering, binaural rendering, or binaural rendering with a room effect.

From the received bitstream, first the IVAS format followed by number of transport channels is decoded. Then the transport channels synthesis is decoded by the decoding tools, namely Single Channel Elements decoder (SCE decoder comprising one core-coder, see clause 6.2.3.2), Channel Pair Elements decoder (CPE decoder comprising two core-coders, see clause 6.2.3.3), and Multichannel Coding Tool decoder (MCT decoder comprising a joint coding of multiple core-coders, see clause 6.2.3.4) while the core-coder is inherited from the EVS codec with additional flexibility and variable bitrate (clause 6.2.2). Finally, the transport channels are processed and upmixed (and rendered in case of internal renderer) in a scene decoder.

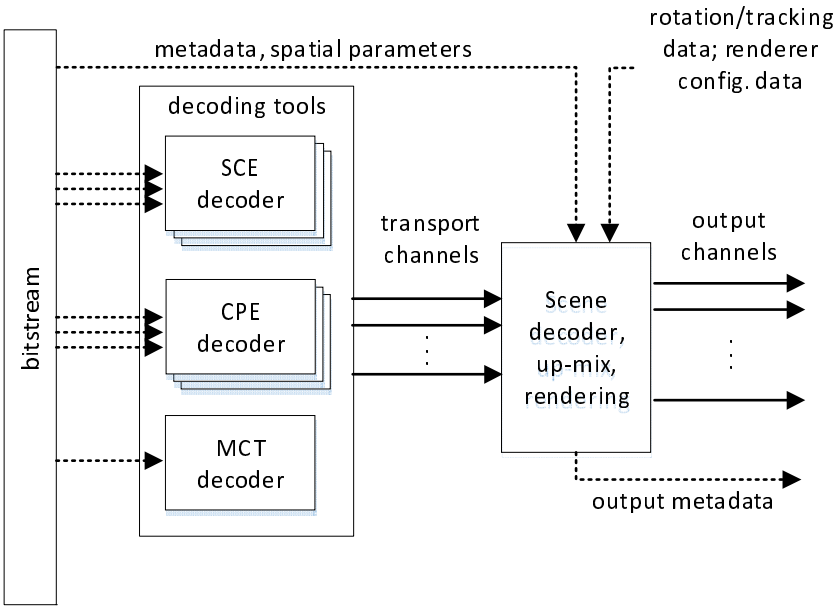


Figure 6.1-1: Decoder Data Flow from IVAS bitstream to output data

The decoder receives all quantized parameters and generates a synthesized signal. Thus, for the majority of encoder operations it represents the inverse of the quantized value to index operations.

As EVS, for the AMR-WB interoperable operation the index lookup is performed using the AMR-WB codebooks and the decoder is configured to generate an improved synthesized signal from the AMR-WB bitstream. For EVS decoding the bitstream decoding functions are equivalent to EVS. IVAS is considered a new bitstream that shall be signalled to the decoder, i.e., whether to operate in EVS (mono) mode or IVAS mode.

The IVAS decoder includes an error concealment scheme for erroneous or lost frames, an adaptive jitter buffer algorithm run on transport channels before the rendering, output signal resampling and internal or external rendering to multiple configurations and options.

6.2 Common processing and decoding tools

6.2.1 Common decoder processing overview

6.2.1.1 High-pass filtering

The high-pass (HP) filtering operation is done on all transport channels (i.e. before rendering) except of the LFE channel. The procedure from clause 5.2.1 is used.

6.2.1.2 Limiter

6.2.1.2.1 Overview

In IVAS, a two-stage distortion limiter placed at the output stage of the decoder provides protection against distortions in the output signal. The limiter is implemented as shown in Figure 6.2-1. It consists of a peak level detection (first stage detection), a strong saturation detection (second stage detection), a calculation of parameters, and an attenuation (IIR-based low-pass) filtering. It is noted that for the multichannel output signal, the distortion may occur in one, several, or all output channels. The attenuation is then to be applied to all output channels regardless if any of them originally contains saturations or not in order to preserve the spatial characteristics of the output signal.

To avoid any processing delay, no look-ahead is used. The limiter is active in the current frame if either the threshold value is exceeded in this frame, or the threshold has been exceeded in frames processed earlier and the system has not yet returned to its default state. Time taken by the system to return to within 1% of its default state (release time) is governed by a heuristic value ranging from 0 to 1. The release heuristic increases on each frame that contains a sample with a value above the threshold and decreases on each frame with all sample values below the threshold. Values of 0 and 1 map to the shortest (0.005 seconds) and longest (1 second) release time, respectively. The goal of this heuristic is to avoid the "pumping" effect when only sharp transients exceed the threshold, but also keep the gain curve smoother if the threshold is exceeded in many frames in a short span of time.

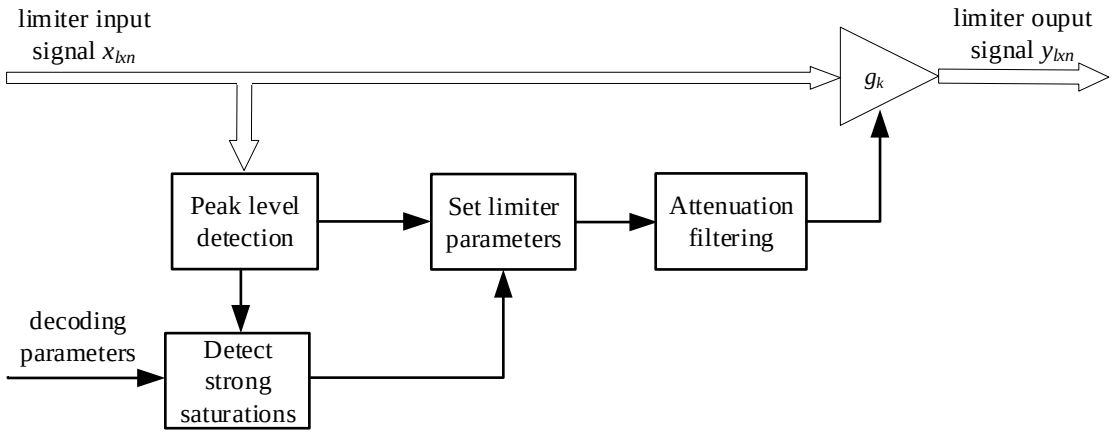


Figure 6.2-1: Schematic block diagram of the IVAS limiter

6.2.1.2.2 Detailed algorithmic description

The input to the limiter at frame with index k is a buffer containing a time-domain signal $x_{l\times n}$ consisting of l samples per each of n channels. The maximum absolute sample (peak) value $x_{k\,max}$ is found in the limiter multichannel input signal $x_{l\times n}$:

$$x_{k\,max} = \max_{\substack{0\le i<l \\ 0\le c<n}} (|x_{ic}|)_{(i,c)} \tag{6.2-1}$$

Based on $x_{k\,max}$ the values of release heuristic r_k and provisional target (control) gain γ_k are calculated for the frame:

If $x_{k\,max} > x_{threshold}$, then:

$$\gamma_k = \frac{x_{threshold}}{x_{k\,max}} \tag{6.2-2}$$

$$r_k = \min\left\{1, r_{k-1} + 4\frac{l}{f_s}\right\} \tag{6.2-3}$$

else, if $x_{k\,max} \leq x_{threshold}$, then:

$$\gamma_k = 1 \tag{6.2-4}$$

$$r_k = \max\left\{1, r_{k-1} - \frac{l}{f_s}\right\} \tag{6.2-5}$$

where the limiter threshold $x_{threshold}$ is a constant equal to 32729 (~-0.01 dBFS).

At this stage an additional heuristic is used to detect strong saturations in the limiter input signal. These are likely to occur when decoding frames containing bit errors, although in exceptional circumstances may also be encountered when processing valid frames. The heuristic is described in 6.2.1.2.3 below. It yields the decision whether strong limiting should be applied in the current frame and updates the provisional target gain γ_k.

The final value of the target gain $g_{k\,target}$ is determined as follows:

$$g_{k\,target} = \begin{cases} \gamma_k, & \text{if strong limiting active} \\ \max\{\gamma_k, 0.1\}, & \text{otherwise} \end{cases} \tag{6.2-6}$$

Time-constants a_{attack} and $a_{k\,release}$ used for smoothing the gain curve in the current frame are calculated as follows:

$$a_{attack} = 0.01^{\frac{1}{f_s t_{attack}}} \tag{6.2-7}$$

$$a_{k\,release} = 0.01^{\frac{1}{f_s t_{k\,release}}} \tag{6.2-8}$$

where t_{attack} and $t_{k\,release}$ describe the attack and release time in seconds that the gain curve takes to reach 99% of its target value. t_{attack} is a constant always equal to 0.005 and $a_{k\,release}$ depends on the release heuristic value in the current frame:

$$t_{k\,release} = 0.005 \cdot 200^{r_k}. \tag{6.2-9}$$

Next, l values for the gain vector G are calculated:

$$g_i = \begin{cases} a_{attack}\left(g_{i-1} - g_{k\,target}\right) + g_{k\,target}, & g_{i-1} > g_{k\,target} \\ a_{release}\left(g_{i-1} - g_{k\,target}\right) + g_{k\,target}, & g_{i-1} \leq g_{k\,target} \end{cases}. \tag{6.2-10}$$

Finally, the limiter output signal $y_{l\times n}$ is created by applying the gain vector to each sample of each limiter input channel via element-wise multiplication:

$$Y_{(i,c)} = (x_{ic} g_i)_{(i,c)}. \tag{6.2-11}$$

Note that due to the lack of look-ahead, the non-zero attack time and the restricted gain reduction range, it is possible that the limiter output contains samples whose absolute value exceeds the limiter threshold. This is accounted for at later stages of the processing chain.

Initial values for recursively defined variables:

- Limiter gain before frame 0: $g = 1$
- Release heuristic before frame 0: $r = 0$

6.2.1.2.3 Handling of strong saturations

The second stage of the limiter from Figure 6.2-1 contains a detection of strong saturations to handle saturations with an unexpectedly high level. A schematic block diagram of its logic is shown in Figure 6.2-2.

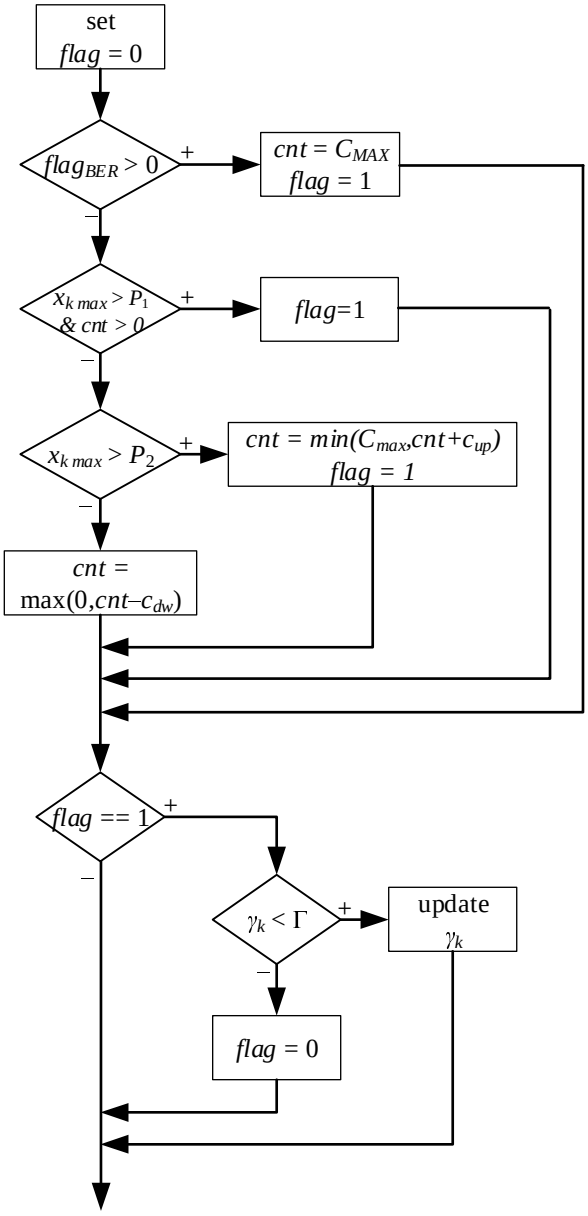


Figure 6.2-2: Schematic block diagram of the strong saturation detection logic

The strong saturation detector is applied to every channel of the limiter input multichannel signal and produces a saturation detection flag $flag$ indicating whether a strong saturation was detected or not, and an updated provisional target (control) gain γ_k. Referring to Figure , the saturation detection starts with initialization, at the beginning of every frame of multichannel signal, of the saturation detection flag: $flag = 0$. The saturation detector then comprises two parts:

- a first calculator which updates a saturation detection counter cnt which stores a metric measuring probability that a strong saturation is present in a current signal synthesis processing frame and produces the saturation detection flag $flag$; and

- a second calculator which (a) decides if additional attenuation is to be applied to the limiter input signal $x_{l\times n}$ and (b) determines an updated provisional target gain γ_k value to control the attenuation to be applied to the limiter input signal $x_{l\times n}$.

The first calculator of the saturation detector starts with a decision whether bit errors, signalled from the decoder by a parameter $flag_{BER}$ were detected in the received bitstream of the current frame or not. If bit errors were detected, indicated by parameter $flag_{BER} == 1$, the saturation detection counter cnt is updated to its maximum value $cnt = C_{MAX}$, $C_{MAX} = 50$, and the saturation detection flag $flag$ is set to its saturation indicating value $flag = 1$. The saturation detection then continues in the second calculator.

If the parameter $flag_{BER} == 0$, indicating that no bit errors are detected in the received bitstream, the saturation detector continues with a second decision using the detected peak value $x_{k\,max}$ from Equation (6.2-1). If the peak value $x_{k\,max}$ is greater than a given threshold P_1 and the saturation detection counter $cnt > 0$, the saturation detection flag is set to its strong saturation indicating value of $flag = 1$. The saturation detection is then continued in the second calculator.

Then, if (a) the parameter $flag_{BER} == 0$, and (b) the peak value $x_{k\,max}$ is equal to or lower than the threshold P_1 and/or the strong saturation detection counter $cnt = 0$, the first calculator continues with a third decision whether the peak value $x_{k\,max}$ is greater than a given threshold P_2. If $x_{k\,max} > P_2$, the strong saturation detection counter cnt is updated to the minimum between C_{MAX} and the sum of $cnt + c_{up}$ and the saturation detection flag is set to $flag = 1$. Here $P_1 = 3 \cdot x_{threshold}$, $P_2 = 10 \cdot x_{threshold}$ and the constant $c_{up} = 20$ represents an incrementation step up. The saturation detection then continues in the second calculator.

Next, if the parameter $flag_{BER} == 0$, (b) the peak value $x_{k\,max}$ is equal to or lower than the threshold P_1 and/or the saturation detection counter $cnt = 0$, and (c) the peak value $x_{k\,max}$ is equal to or lower than another threshold P_2, the first calculator updates the saturation detection counter cnt to the maximum between 0 and the difference $cnt - c_{dw}$ where the constant c_{dw}=1 represents an incrementation step down. The saturation detection then continues in the second calculator.

At this point, the contribution of the first calculator of the strong saturation detector is completed.

In the second calculator of the strong saturation detector, if the saturation detection flag $flag = 0$, the strong saturation detecting method is ended and no additional attenuation is requested at the output of the second stage of the two-stage multichannel limiter.

On the other hand, if the saturation detection flag $flag = 1$, the second calculator of the strong saturation detector continues with a decision using the provisional target gain (aka gain correction factor) γ_k.

If the gain correction factor γ_k is lower than threshold $\Gamma = 0.3$, the second calculator updates its value as follows:

$$\gamma_k := \frac{\gamma_k}{\alpha} \tag{6.2-12}$$

where $\alpha = 3.0$ is an additional correction factor to control the strength of the additional limitation of the provisional target gain γ_k. Consequently, the target gain from the first stage of the limiter is updated as well.

Otherwise, if $\gamma_k \geq \Gamma$, the second calculator sets the saturation detection flag to $flag = 0$ and performs no updating of the provisional target gain γ_k.

6.2.1.3 Time-domain decorrelator

The time-domain decorrelator takes an input audio signal and generates up to 8 outputs that are decorrelated from the input signal as well as from each other. These decorrelated outputs are used by the SBA decoder (subclause 6.4), where they are combined with the transient-containing input signal to fill in energy that was lost in the downmix process. The decorrelators are also used in MASA (subclause 6.5), Multi-channel MASA (subclause 6.7.2) and Parametric Upmix (subclause 6.7.4) processing modes.

The time-domain decorrelator includes a transient processing stage and a decorrelation stage, as shown in Figure 6.2-3. The transient processing stage is used to scale the input and output of the decorrelation stage for the purpose of separating the transient component from the continuous component of the input signal, such that decorrelation is only applied to the continuous component.

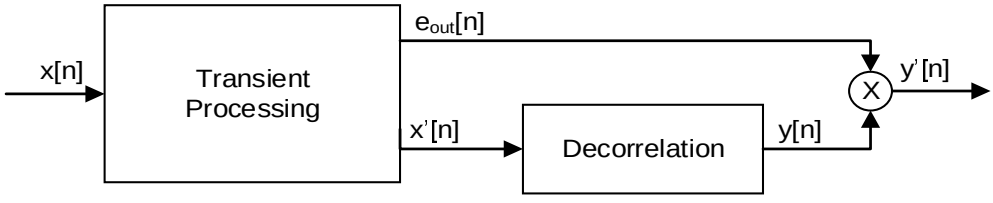


Figure 6.2-3: Time-domain decorrelator block diagram

The transient processing stage begins by filtering the input $x[n]$ with a first-order high-pass filter with a cut-off frequency of 3 kHz to obtain $x_{hp}[n]$. The input signal envelope $e[n] = |x_{hp}[n]|$ is then used to obtain a fast envelope

e_1[n] and a slow envelope $e_2[n]$. The fast envelope $e_1\,[n]$ is obtained by filtering $e[n]$ with an integrator having a 5 ms integration time, whilst $e_2[n]$ is obtained by filtering $e[n]$ with an integrator having 80 ms integration time.

$$e_1[n] = (1-\alpha_1)|x_{hp}[n]| + \alpha_1 e_1[n-1] \tag{6.2-13}$$

and

$$e_2[n] = (1-\alpha_2)|x_{hp}[n]| + \alpha_2 e_1[n-1] \tag{6.2-14}$$

where $\alpha_1 = e^{\frac{-1}{0.005 f_s}}$, $\alpha_2 = e^{\frac{-1}{0.080 f_s}}$ and f_s is the sampling rate in Hz. The fast and slow envelopes are then used to determine a time-varying scaling function $e_{in}[n]$ as follows:

$$e_{in}[n] = \min(c_{duck}(e_{in}[n-1]-1)+1, \alpha_{duck}\frac{e_2[n]}{e_1[n]}) \tag{6.2-15}$$

where α_{duck} is a constant specific to the operating mode and $c_{duck} = e^{\frac{-1}{0.05 f_s}}$.

A second time-varying scaling function $e_{out}[n]$ is computed as

$$e_{out}[n] = \min(c_{duck}(e_{out}[n-1]-1)+1, \alpha_{duck}\frac{e_1[n]}{e_2[n]}). \tag{6.2-16}$$

The input $x'[n]$ to the decorrelation stage is given by $x'[n] = e_{in}[n]x[n-d]$, where d represents a 2 ms delay.

The decorrelation stage consists of a cascade of $k = 0,\ldots,K-1$ first-order all-pass filters for each output channel i. Each filter $y_{ap(i,k)}[n]$ is of the form $y_{ap(i,k)}[n] = g_{ik}x[n] + x[n-m_k] - g_{ik}y_{ap(i,k)}[n-m_k]$. The number of all-pass filters K depends on the number of decorrelator output channels N_{out}:

$$K = \max(2, 2^{\lceil log_2(N_{out})\rceil}) \tag{6.2-17}$$

The gain coefficients for each filter in each channel are given by

$$g_{ik} = 0.4\mathrm{h}_{ik}, \tag{6.2-18}$$

where $h_{ik}$are the coefficients of the 8x8 Hadamard matrix Ψ_{H8}.

$$\Psi_{H8} = \begin{pmatrix} h_{00} & \cdots & h_{07} \\ \vdots & \ddots & \vdots \\ h_{70} & \cdots & h_{77} \end{pmatrix} \tag{6.2-19}$$

The delay m_k of each filter is computed as follows:

$$\mathrm{R}_k = 3^{\frac{k}{4}} \tag{6.2-20}$$

$$m_k = \frac{f_s R_k \tau_{ap}}{\sum_{j=0}^{K-1} R_j}, \tag{6.2-21}$$

where $\tau_{ap} = 20$ ms. The outputs $y_i[n]$ of the decorrelation stage for channels $i = 0..N_{out}-1$ are scaled by e_{out} to produce the decorrelator output $y_i'[n]$:

$$y_i'[n] = y_i[n]e_{out}[n]. \tag{6.2-22}$$

6.2.2 Core-decoder processing

6.2.2.1 Overview

Similar to the core-coder (clause 5.2.2), the core-decoder module in IVAS is based on the EVS codec from [3] while it is supplemented by more flexibility and adaptation for a multichannel processing. In the remaining part of clause 6.2.2, only the differences of IVAS core-decoder wrt. the EVS decoder Specification from [3] are then provided.

The core-decoder module bitrate (one per one core-decoder) is not directly signalled in the bitstream but it is derived from the bit-budgets of all decoder modules in the following steps:

First, an IVAS format signalling is read from the bitstream.

Second, stereo or spatial decoding tools processing (if present) is done and their respective bit-budgets computed.

Similarly, a metadata decoding module (if present) is processed and its bit-budget is computed.

The bit-budgets for IVAS format signalling, the stereo / spatial decoding tools and metadata decoding module are then subtracted from the IVAS total bit-budget.

Finally, the remaining bit-budget is distributed among core-decoders based on the type of the core. The details of the LP-based core decoding are provided in clause 6.2.2.2 while the MDCT-based core-decoder details are provided in clause 6.2.2.3.

6.2.2.2 LP based decoding

6.2.2.2.1 Variable bitrate ACELP decoding

Similar to the encoder, the bit-budget allocated to the ACELP core decoding module can fluctuate between a relatively large minimum and maximum bitrate span with a granularity of 1 bit. The decoder processing is then the same as used at the encoder and described in clause 5.2.2.3.2, subclause Variable bitrate ACELP coding, including the bit-budget allocator, innovation codebook bit-budget distribution and bit-budget distribution in high bitrate ACELP.

6.2.2.2.2 Fast algebraic codebook decoding

The received algebraic codebook index is used to extract the positions and amplitudes (signs) of the excitation pulses and to find the algebraic codevector as described in clause 6.1.1.2.1. of [3]. When the number of bits assigned to decode the codebook index belongs to the fast algebraic codebook configurations from Table 5.2-10, a dedicated decoding processing supporting these configurations is run while this decoding processing simply extends the decoding present algorithms from EVS so it supports small codebook sizes and a longer subframe length.

6.2.2.2.3 AVQ Bit Savings Decoder

6.2.2.2.3.1 Overview

Figure 6.2-4 shows an example of an AVQ decoder for decoding quantized DCT coefficients. The AVQ bit-saving algorithm is integrated in the code converter block in the figure.

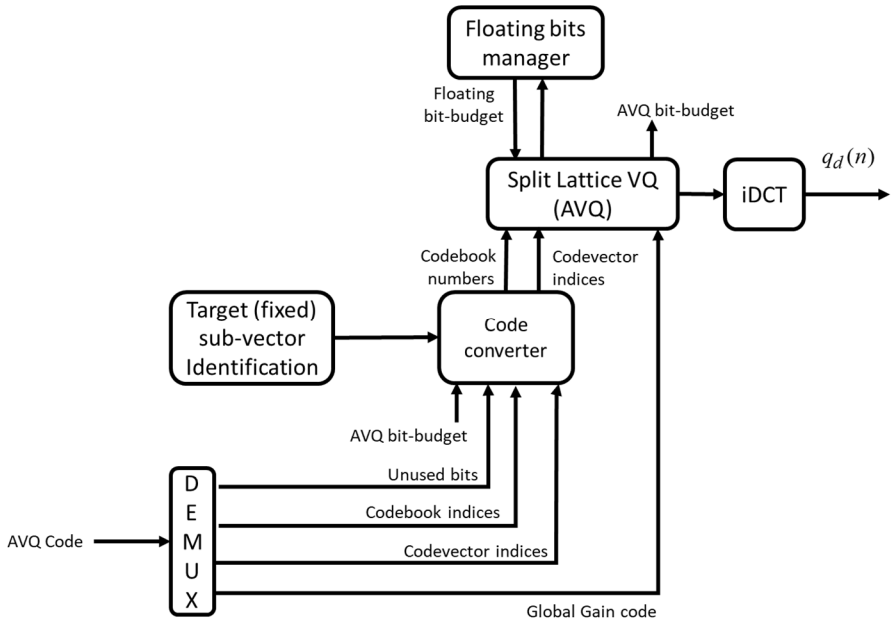


Figure 6.2-4: Overview of AVQ Bit Savings Decoder

6.2.2.2.3.2 Code Converter in Decoder

6.2.2.2.3.2.1 Fundamentals

Code converter converts the number of unused bits to a codebook number (index, indicator) by utilizing the other codebook numbers (indices, indicators) of the sub-vector and the AVQ bit-budget.

Figure 6.2-5 shows a basic functional block diagram of the code converter. The number of unused bits decoder decodes the number of unused bits and outputs it to the codebook index reconstructor. The number of usable bits calculator calculates the number of usable bits by subtracting the number of bits used for the encoded code of the other codebook indices and their corresponding codevector indices from AVQ bit budget inputted from AVQ block and inputs the number of usable bits to the codebook index reconstructor. The codebook index reconstructor reconstructs the codebook number (index or indicator) using the calculated number of usable bits and the decoded number of unused bits. Detailed reconstruction algorithm will be described in the following sub-clause. Finally, the codebook indices generator outputs codebook indices using the reconstructed codebook index for a predetermined position (sub-vector) and inputted encoded code of codebook indices for the other positions (sub-vectors)

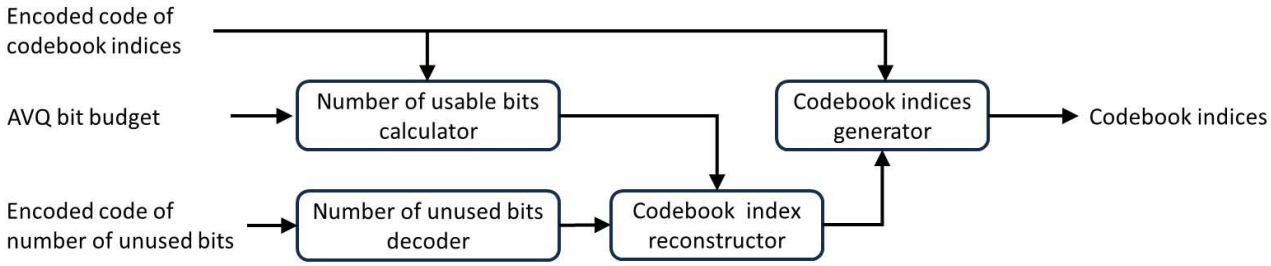


Figure 6.2-5: Functional block diagram of code converter in the decoder

6.2.2.2.3.2.2 Selection of Targeted Sub-Vector

The targetted sub-vector for the code conversion is in the same manner as explained in the encoder clause 5.2.2.3.2.5.2.2

6.2.2.2.3.2.3 Case of Selecting the 3rd Sub-Vector

Figure 6.2-6 shows code conversion processing flow for the case of selecting the 3rd sub-vector as the targeted sub-vector. The processing flow is the same as the code conversion processing flow in the encoder side. Since Figure 6.2-8 includes the step of decoding codebook index using the decoded number of unused bits, it is explained as AVQ code decoding process rather than code conversion process. Detailed process of decoding codebook index (number) using the decoded number of unused bits will be described in section 6.2.2.2.3.3. Other processing steps are the same with those described in clause 5.2.2.3.2.5.2.3. In the figure, Th1=30 bits, Th2=9 bits and Th3=79 bits.

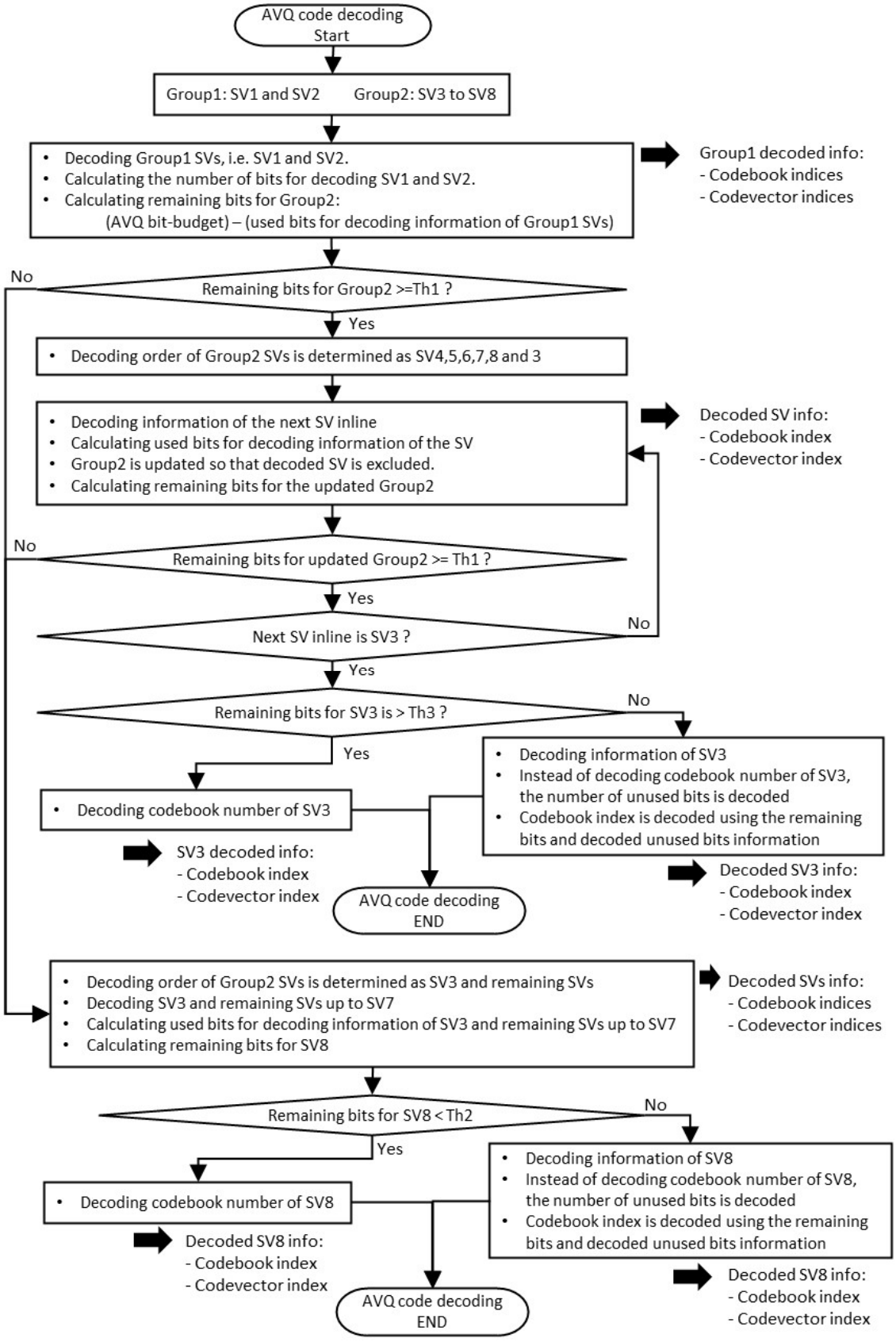


Figure 6.2-6: Flow of AVQ code decoding (in the case of selecting SV3 as target SV)

6.2.2.2.3.2.4 Case of Selecting the 8th Sub-Vector

Figure 6.2-7 shows the code conversion processing flow for the case of selecting the 8th sub-vector as the targeted sub-vector. The flow is the same as the code conversion process in the case of targeting SV8 in the encoder side. Since Figure 6.2-8 includes the step of decoding codebook index (number) using the decoded number of unused bits, it is explained as AVQ code decoding process rather than code conversion process. Detailed process of decoding codebook index (number) using the decoded number of unused bits will be described in clause 6.2.2.2.3.3. Other steps are the same as those described in clause 5.2.2.3.2.5.2.4. In the figure, Th2=9 bits and Th3=79 bits.

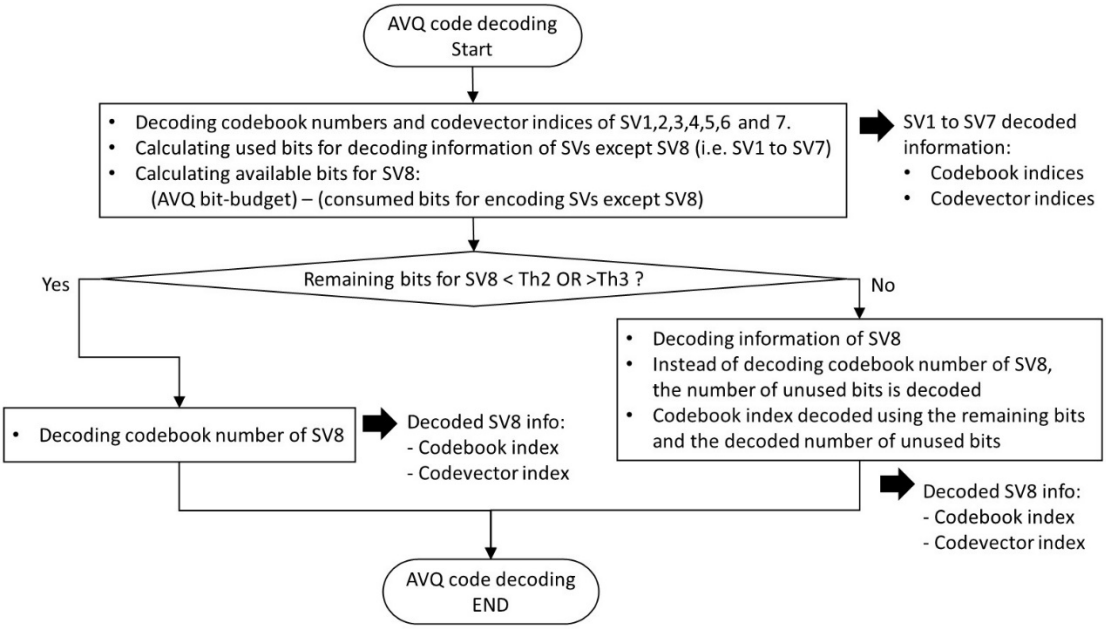


Figure 6.2-7: Flow of AVQ code decoding (in the case of selecting SV8 as target SV)

6.2.2.2.3.3 Decoding Codebook Index using Decoded Unused Bit

Figure 6.2-8 shows a flow diagram of decoding SVd parameters. The flow is the same as the one for unused bit encoding except the last parts in which the SVd parameters are decoded after decoding the number of unused bits. Therefore, the processing steps are the same steps described in clause 5.2.2.3.2.5.3. In the last two blocks, using the decoded number of unused bits and the number of remaining bits for SVd, the number of bits for SVd is reconstructed, and the codebook index (number) and the codevector index is identified.

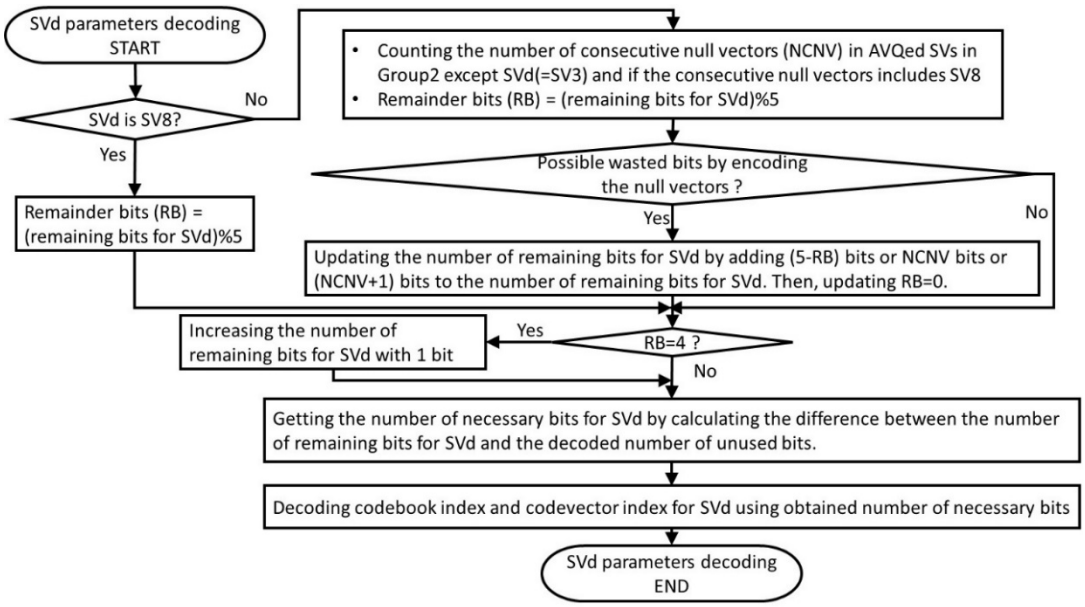


Figure 6.2-8: Flow of decoding SVd parameters

The process of updating the number of remaining bits for SVd code is shown in Figure 6.2-9. This is also the same with the one in the encoder side, which is shown in Figure 5.2-13.

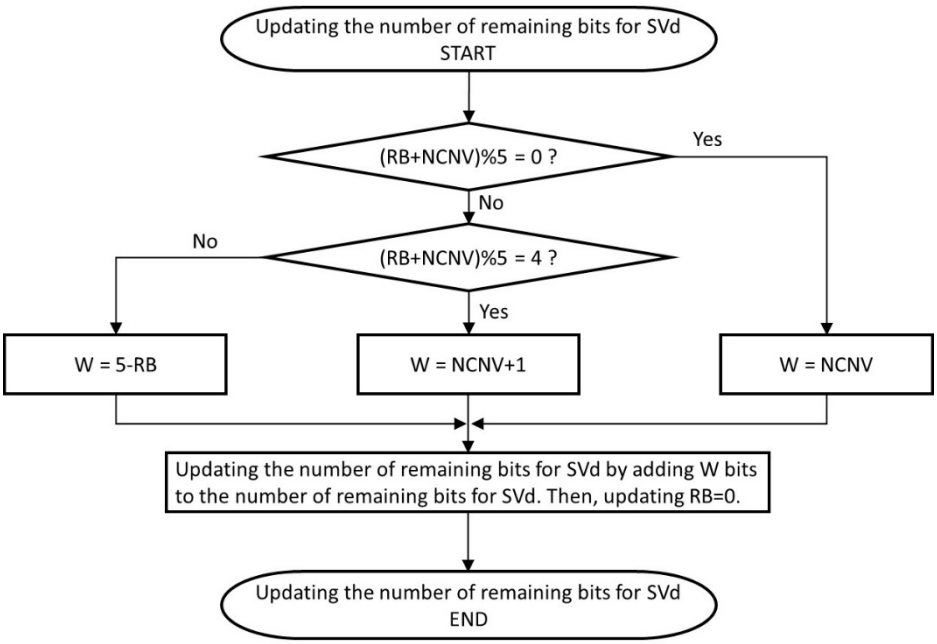


Figure 6.2-9: Process of updating the number of remaining bits for SV3

6.2.2.2.4 Bass Post-Filter

The bass post-filter (BPF) in LP-based decoding is operated in the same way as in EVS ([3], clause 6.1.4.2), with the modification that the filter gain is never estimated at the encoder and thus never written to/read from the bitstream. That means that equation (1544) of [3] is never applied. Additionally, the BPF is switched off in the first active frame after an inactive (DTX) frame if the element mode is IVAS_CPE_DFT or IVAS_CPE_MDCT. In DFT-based stereo decoding, the BPF signal is generated as in EVS, but the signal is added in the DFT domain, see 6.3.2.3.5.

6.2.2.2.5 GSC core decoding

If the IVAS bitstream decoder detects the presence of a frame that is classified as having UNCLEAR content, i.e., neither SPEECH nor MUSIC, the GSC submode F_{tfsm} is decoded, but only if the bitrate available is above 9.2 kbps. Otherwise, the GSC submode flag F_{tfsm} is set to 0. This is shown in the schematic diagram in Figure 6.2-10.

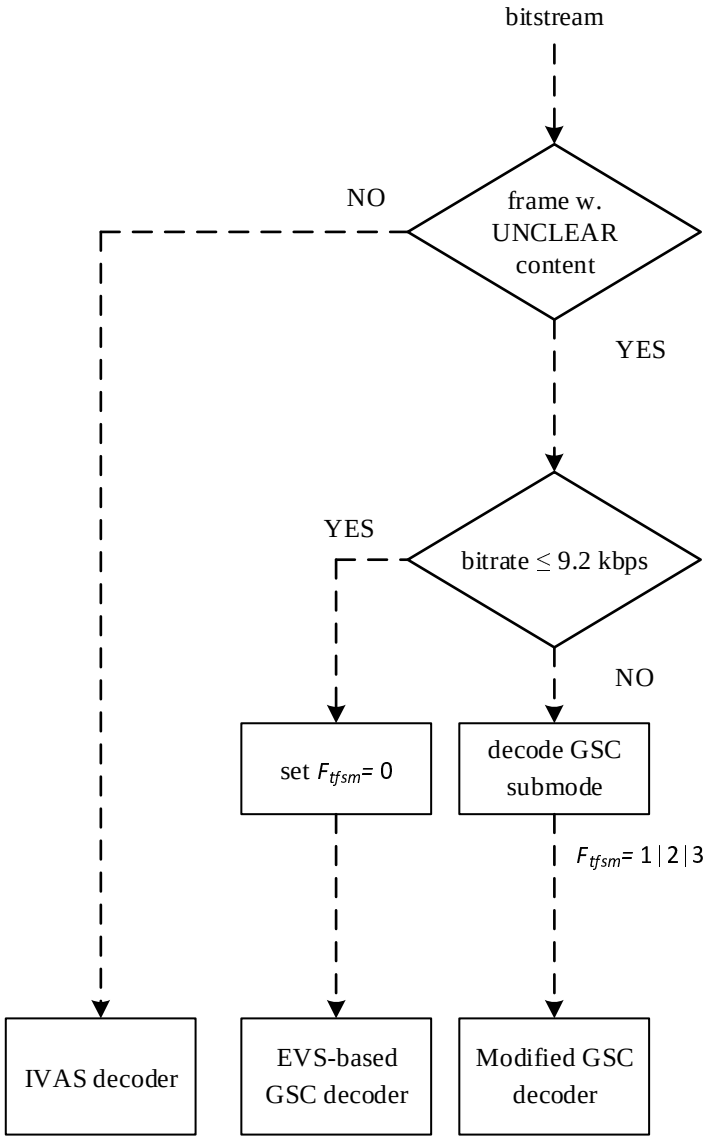


Figure 6.2-10: Decoding of the GSC sub-mode

When the submode flag F_{tfsm} is zero the EVS-based GSC decoder is used to decode the remaining part of the bitstream. Otherwise, a modified version of the GSC decoder is used to generate the output synthesis signal as shown in in Figure 6.2-10. The modifications applied to the EVS-based GSC decoder are essentially the replica of the modifications applied to the GSC encoder, described in Figure 5.2-15 in clause 5.2.2.3.2.6. In particular, the modifications involve bitrate distribution between the time-domain coder and the frequency-domain coder, allocation of bits per frequency bands, re-construction of the combined time-domain and frequency-domain excitation signal. This is illustrated in Figure 6.2-11.

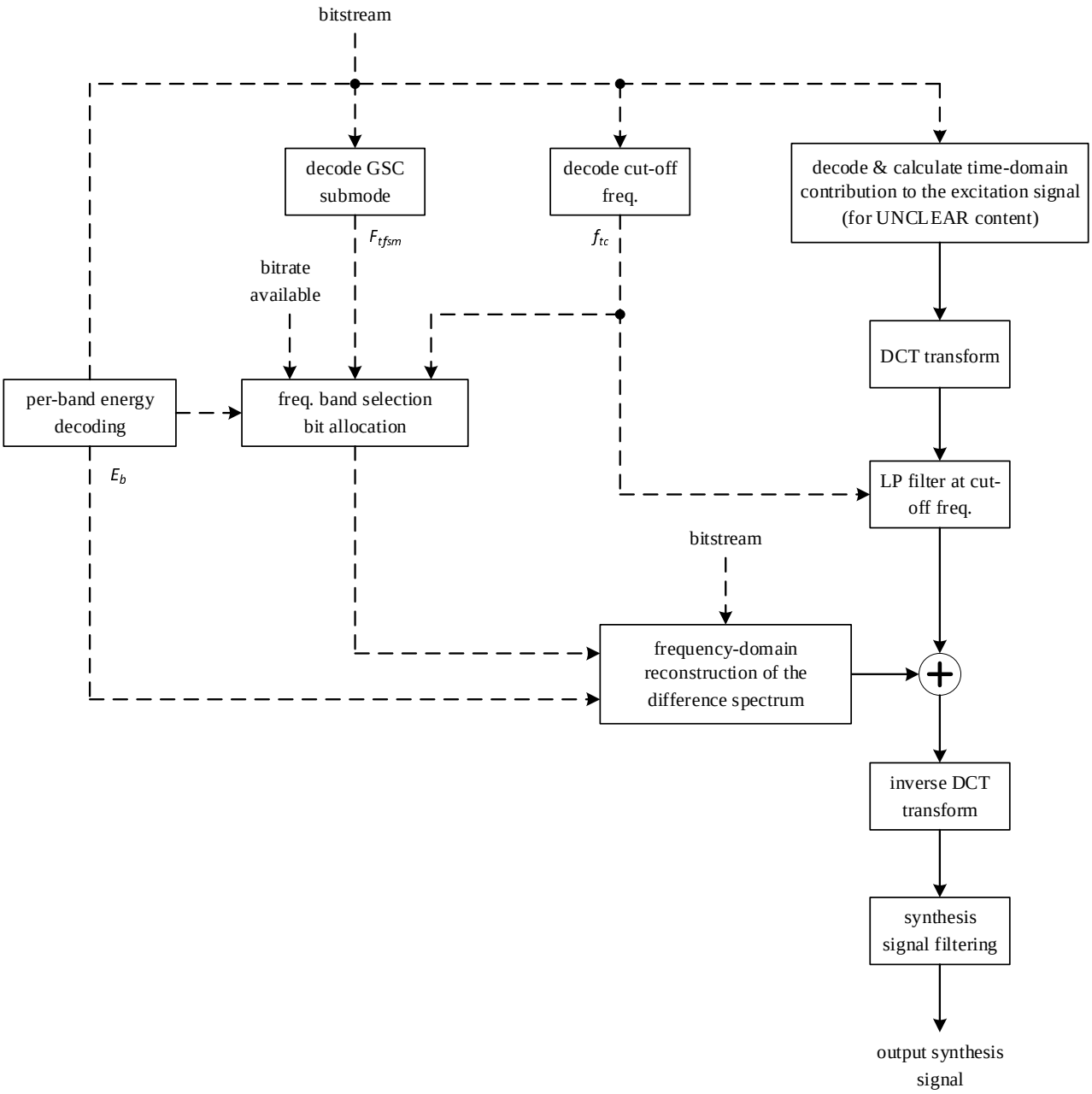


Figure 6.2-11: Modified GSC decoder

Note, that the time-domain contribution of the excitation signal may consist of four, two or one subframes. The time-domain contribution of the excitation signal always includes the contribution from the adaptive codebook (ACB) and may include the contribution from the algebraic codebook (FCB), depending on the bitrate available and on the decoded GSC submode F_{tfsm}. This is described in detail in clause 5.2.2.3.2.6.

Once the decoding of the time-domain contribution of the excitation signal is finished it is converted into frequency domain using DCT transform. The transformed signal is then low-pass filtered at a cut-off frequency corresponding to the decoded GSC submode. The GSC decoder then reconstructs the frequency-domain contribution of the excitation signal. This is done by decoding the selected frequency bands and the allocation of bits per frequency bands from the bitstream. Per-band energy is then decoded from the bitstream as well and converted into per-band scaling gains. The per-band scaling gains are then applied to the reconstructed spectrum. This completes the reconstruction of the frequency-domain contribution of the excitation signal. The reconstructed frequency-domain contribution of the excitation signal is then added to the time-domain contribution in frequency domain. Finally, an inverse DCT transform is applied on the combined spectrum to obtain the excitation signal in time domain. The reconstructed excitation signal in time domain is then filtered using the synthesis filter to get the output synthesis signal.

6.2.2.2.6 Bandwidth extension in time domain

At 1.75 kbps the SWB TBE mode LSFs are transmitted with 20 bits. The spectral envelope is dequantized in three stages.

The first stage consists of 4-bit 6-dimensional VQ which is used to produce a 6th order quantized first sub vector $\hat{L}^1$. The codebook for this stage is stored in `cb_LSF_BWE.` The second stage consists of 8-dimensional Lattice VQ with 15 bits which is used to produce an 8th order quantized second sub vector $\hat{L}^2$. The quantized first sub vector is extended to an 8th order vector by inserting two zeroes for the 7th and 8th vector components, thereby forming an 8th order zero extended quantized first sub vector. The zero extended quantized first sub vector is then added to the quantized second sub vector $\hat{L}^2$ to give the quantized combined LSF sub vector $\hat{L}^{12}$

$$\hat{L}^{12} = \hat{L}^1 + \hat{L}^2$$

The final quantization stage uses the 8 coefficients of the quantized combined LSF sub vector $\hat{L}^{12}$ to predict the last two LSF coefficients (vector components 8 and 9) with an optimal predictor vector $\tilde{\rho}$ comprising 8 prediction coefficients. In all there are two different predictor coefficient sets, which are stored in the table `LastCoefPred_1bit,` of which one set of prediction coefficients is used to predict the 9th and 10th LSF coefficients (vector components 8 and 9). The predictor set selected for use in the prediction stage is signaled with a 1-bit flag. Each predictor coefficient set consists of an 8-coefficient vector $\widetilde{\rho_8}$ for predicting the 9th LSF coefficient and an 8-coefficient vector $\widetilde{\rho_9}$ for predicting the10th LSF coefficient. The 10th order quantized LSF vector $\hat{L}^3$ is formed by concatenating the quantized combined LSF sub vector $\hat{L}^{12}$with the predicted LSF coefficients $\hat{L}^3_{8,9}$. The LSF coefficients $\hat{L}^3_{8,9}$ are each LSF predicted according to

$$\hat{L}^3_8 = \widetilde{\rho_8} * \hat{L}^{12} \text{ and } \hat{L}^3_9 = \widetilde{\rho_9} * \hat{L}^{12}$$

and where $*$ denotes the vector scalar product operator.

The final 10th order SWB TBE SHB LSF vector is formed by adding a mean vector.

$$\hat{L} = \hat{L}^3 + L^{mean}$$

Finally, the resulting SHB LSF $\hat{L}$ vector is sorted into an increasing order of magnitude of vector components. Then the vector is reversed to a decreasing order of magnitude for use in the SHB synthesis.

6.2.2.2.7 Multimode FD bandwidth extension coding

6.2.2.3 MDCT based decoding

6.2.2.3.1 General

The IVAS MDCT based decoding follows the MDCT Coding Mode decoding in EVS, as described in clause 6.2 of [3]. Differences compared to EVS are described in the subsequent clauses.

6.2.2.3.2 Variable bitrate MDCT decoding

The HQ Core decoder has been adapted for IVAS operation to handle varying bit rate from 16.4 kbps up to 48 kbps, the same way as the HQ Core encoder as described in clause 5.2.2.3.3.2. The fine structure decoder bit rate is obtained from the core bit rate R_{core} to match the bit distribution used by the encoder. For sub-mode HVQ (clause 5.3.4.2.5 of [3]) the number of peaks is calculated the same way as in equation (5.2-184).

6.2.2.3.3 TCX entropy decoding

The TCX entropy decoding in IVAS follows the TCX entropy coding in EVS as described in clause 6.2.2.3.1 of [3]. For lower complexity, the low-level arithmetic coding routines are however replaced by an optimized arithmetic decoding routine (range coder).

The arithmetic decoder is described by the following pseudo-code. For the IVAS operation modes, the routine `rc_uni_dec_read_symbol_fast()`replaces the routine `ari_decode()`, which is described in in clause 6.2.2.3.1 of [3]. It takes as input arguments the cumulative frequency table *cum_freq[]*, the size of the alphabet *cfl*, the symbol frequency table *sym_freq[]* and the variable tot_shift. The symbol frequency table *sym_freq[]* can be derived from the cumulative frequency table *cum_freq[]* by simply determining the delta of two neighbored cumulative

frequencies *sym_freq[i] = cum_freq[i+1] - cum_freq[i]*. The variable *tot_shift* indicates the total frequency as a power of 2.

```
void rc_uni_dec_init( int16_t max_available_bits )
{
    int16_t i;

    uint32_t rc_low = 0;
    uint32_t rc_range = 0xFFFFFFFF;

    int16_t bit_count = 0;
    int16_t max_allowable_bit_count = max_available_bits + 30;
    int16_t bit_error_detected = 0;
    for ( i = 0; i < 4; i++ )
    {
        rc_low = ( rc_low << 8 ) + rc_uni_dec_read( rc_st_dec );
    }
}

uint16_t rc_uni_dec_read_symbol_fast( uint16_t cum_freq_table[], uint16_t sym_freq_table[], uint16_t
alphabet_size, uint16_t tot_shift )
{
    uint16_t sym_begin;
    uint16_t sym_end;
    uint16_t sym_middle;
    uint32_t low;
    uint32_t range;
    uint16_t ceil_log2_alphabet_size;
    uint16_t step;
    uint32_t reversed_low;

    low = rc_low;
    range = rc_range;
    range >>= tot_shift;

    if ( low >= ( range << tot_shift ) )
    {
        bit_error_detected = 1;
        rc_range = 0xFFFFFFFF;
        rc_low = rc_range;

        return 0; /* return the minimum valid value for the output */
    }

    sym_begin = 0;
    sym_end = alphabet_size;

    ceil_log2_alphabet_size = 31 - norm_l( alphabet_size - 1 );
    reversed_low = ( range << tot_shift ) - low;

    for ( step = 0; step < ceil_log2_alphabet_size; step++ )
    {
        sym_middle = ( sym_begin + sym_end ) >> 1;

        if ( range * ( ( 1 << tot_shift ) - cum_freq_table[sym_middle] ) >= reversed_low )
        {
            sym_begin = sym_middle;
        }
        else
        {
            sym_end = sym_middle;
        }
    }
    rc_range = range;
    rc_uni_dec_update( rc_st_dec, cum_freq_table[sym_begin], sym_freq_table[sym_begin] );

    return sym_begin;
}

void rc_uni_dec_update( cum_freq, sym_freq )
{
    rc_low -= cum_freq * rc_range;
    rc_range *= sym_freq;

    if ( rc_range < 0x01000000 )
    {
        rc_low = ( rc_low << 8 ) + rc_uni_dec_read( rc_st_dec );
```

```
        rc_range <<= 8;
        if ( rc_range < 0x01000000 )
        {
            rc_low = ( rc_low << 8 ) + rc_uni_dec_read( rc_st_dec );
            rc_range <<= 8;
        }
    }

    return;
}

int16_t rc_uni_dec_finish( )
{
    int16_t total_bit_count;
    int16_t bits;

    bits = norm_l( rc_range >> 24 ) - 22;
    bits++;

    total_bit_count = ( bit_count - 32 ) + bits;

    return total_bit_count;
}

int16_t rc_uni_dec_virtual_finish( )
{
    return bit_count + norm_l( rc_range >> 24 ) - 53;
}

int16_t rc_uni_dec_read( )
{
    int16_t byte_read;
    uint16_t *shifted_bit_buffer;

    shifted_bit_buffer = bit_buffer + bit_count;
    bit_count += 8;

    if ( bit_count > max_allowable_bit_count )
    {
        bit_error_detected = 1;
        rc_range = 0xFFFFFFFF;
        rc_low = rc_range;

        return 0;
    }

    byte_read = ( (int16_t) shifted_bit_buffer[0] << 7 ) | ( (int16_t) shifted_bit_buffer[1] << 6 )
|
                ( (int16_t) shifted_bit_buffer[2] << 5 ) | ( (int16_t) shifted_bit_buffer[3] << 4 )
|
                ( (int16_t) shifted_bit_buffer[4] << 3 ) | ( (int16_t) shifted_bit_buffer[5] << 2 )
|
                ( (int16_t) shifted_bit_buffer[6] << 1 ) | ( (int16_t) shifted_bit_buffer[7] );

    return byte_read;
}
```

6.2.2.3.4 PLC method selection

In case of a packet loss when the last decoded good frame was coded with the HQ MDCT mode, the preliminary signal analysis described in 5.4.3.1 [4] is performed. The PLC method selection 5.4.3.2 [4] is amended in IVAS operation in the following way:

- An output sampling rate $f_{s,out}$ equal to 8000 Hz is only supported in EVS operation, meaning that the PLC method specified in An output sampling rate $f_{s,out}$ equal to 8000 Hz is only supported in EVS operation, meaning that the PLC method specified in [4] 5.4.3.3, 5.4.3.4 is applied.

- Otherwise, let the expression $evs\_mode\_selection$ denote the decision to use the PLC algorithm specified in [4] 5.4.3.6, and let the expression $ivas\_mode\_selection$ be defined according to

$$ivas\_mode\_selection = T_c < 56 \vee c < 0.85$$

  where $\vee$ is the ‘inclusive or’ operator, c is the correlation parameter and T_c is the pitch computed as in [4] 5.4.3.1.2.

- In case of EVS operation, $evs\_mode\_selection$ corresponds to the decision to use the PLC algorithm specified in [4] 5.4.3.6.
- In case of IVAS operation,
  - if the expression
    $$evs\_mode\_selection \wedge ivas\_mode\_selection$$
    is true, the PLC algorithm specified in [4] 5.4.3.6 is used. Here, ∧ denotes logical 'and'.
  - Otherwise, the PLC algorithm specified in [4] 5.4.3.5 is used.

6.2.2.3.5 Phase ECU enhancements

If the PLC method selection described in clause 5.2.2.1 results in the usage of [4] 5.4.3.5, the Phase ECU frame loss concealment method is used. The basis for this method is the same as in [4] 5.4.3.5, with the enhancements described in this clause. When the Phase ECU is activated, a frequency domain analysis of the previously decoded synthesis is performed as described in [4] 5.4.3.5.2, rendering a frequency spectrum $X_F(k)$ and a magnitude spectrum $|X_F(k)|$. Tonal components of $X_F(k)$ are found by identifying peaks of the spectrum using a refined peak picking method. Which results in a list of assumed tone(s) with the centre(s) located in bins, $k_p(j),\ j = 0, \dots, N_{peaks} - 1$, further assuming one unique tone per found peak.

The refinement consists of a threshold adjustment of the peak finder used for locating potential tones, the significance level is set to 0.93 (0.97 is used for EVS). This reduces the number of found peaks as it avoids classifying sidelobes around tones as false tones, this improves performance in clean tones.

Additionally, the interpolation of the found peaks is made more accurate by using the complex information in the frequency spectral estimate $X_F(k)$ when available for the three bins used for interpolation. That is, the centre peak and the two surrounding bins all have complex valued $X_F(k)$. In the other cases the real-valued parabolic interpolation from EVS is used.

The higher interpolation accuracy is achieved using the following equation.

$$k'_p(j) = K_{jacob}\ RE\left\{\frac{X_F(k_p(j)-1) - X_F(k_p(j)+1)}{2\,X_F(k_p(j)) - X_F(k_p(j)-1) - X_F(k_p(j)-1)}\right\} \quad (6.2\text{-}23)$$

where K_{jacob} is scale factor dependent on the frame length and window type use for the spectral analysis of the prototype frame. Where analysis found that K_{jacob} =1.1453 gave the best performance. Resulting in the peaks frequency of

$$f_{k_p(j)} = \left(k_p(j) + k'_p(j)\right)\ C \quad (6.2\text{-}24)$$

where C is the coarse resolution in Hz/bin for the spectral analysis, in this case $C = 32.25$.

The result of the refined peak picking method is the set of peak locations at fractional frequencies $k'_p(j), j = 0, \dots, N_{peaks} - 1$, which is represented by an integer peak location $k_p(j)$ and the peak neighbourhood $k_p(j) - \delta_1, \dots, k_p(j) - \delta_2$, where δ_1, δ_2 is defined as in [4] 5.4.3.5 equation (190). The bins of the spectrum that are not covered by the peaks $k_p(j)$ and their peak neighbourhood is defined as the noise component of the spectrum. A relative energy E_{NSR} in the range $[0,1]$ between the noise component of the spectrum and the peak component of the spectrum is calculated.

$$E_{NSR} = \frac{\sum_{k \notin G_{peak}(j)} |X_F(k)|^2}{\sum_{k=0}^{N-1} |X_F(k)|^2} \quad (6.2\text{-}25)$$

A noise attenuation factor $g_{PHECU,noise}$ is set depending on E_{NSR} according to

$$g_{PHECU,noise} = \begin{cases} 0.5, & E_{NSR} < 0.03 \\ 1, & E_{NSR} \geq 0.03 \end{cases} \quad (6.2\text{-}26)$$

The noise attenuation factor is applied to the noise component of the spectrum. Once a time domain substitution frame $x_{ph}(n)$ has been obtained as described in [4] 5.4.3.5.3, an MDCT frame completion is performed.

With MDCT frame completion the substitution frame, $x_{ph}(n)$, is combined with already decoded signal from previous frame (or previously recreated, in case of burst errors) through copying and overlap adding of the different signal segments depending on if both signals are available or not for those segments. For the initial part of the MDCT frame (4ms) only the previously decoded (or recreated) signal is available and therefore copied into a processing buffer. The second part (1.625ms) where both already decoded signal and the substitution frame are available are combined with overlap add to the processing buffer. Finally, the remaining part (30.375ms) of the substitution frame is copied into the MDCT processing buffer. The processing buffer is then processed using the MDCT analysis window, followed by the time domain aliasing (TDA) step described in the encoder [4] 5.3.2.2. This result is then used to recreate an approximation of the lost frame using normal decoder side processing [4] 6.2.4.1. That is, first applying the inverse time domain aliasing (ITDA) step, applying the synthesis MDCT window and finally performing the overlap add (OLA) operation to combine the MDCT memory from the previous frame with the approximation of the lost frame. Thereby producing an approximation of the lost time domain frame and an update of the MDCT memory to be used for the next frame.

6.2.2.3.6 Long term prediction processing

LTP post filtering is used in the similar way as described in clauses 5.1.14.1.1.1 and 6.9.2 in [3]. LTP post filtering is used for post-processing TCX20 and TCX10/5 frames at bitrates below 96 kbps. Different to [3] there is no LTP gain reduction at 48 kbps.

The parameters of the LTP filter, being the LTP gain and the pitch gain, may change and are updated in regular update intervals. The framing of 20 ms is chosen as the regular update interval. LTP filter coefficients are derived from the LTP parameters in the same way as in [3]; the LTP filter coefficients change whenever the LTP parameters change.

If a delay compensation of D_{LTP} is needed, then the first D_{LTP} samples of s_{LTP} are obtained as in [3].

If the LTP gain is non-zero in the previous and in the current frame, it means that the LTP post filtering is active in the previous and in the current frame. If the LTP is active in the previous and in the current frame and if the LTP parameters have changed, OAO smoothing is used for the transition in the initial subinterval of 5 ms. OAO smoothing is more efficient than the smoothing in [3] that uses zero input response.

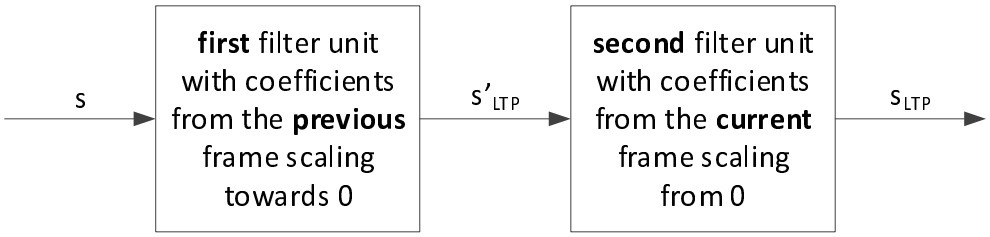


Figure 6.2-12: Block diagram of the LTP post filtering

OAO smoothing is accomplished by two filter units. The block diagram for the LTP post filtering that uses OAO smoothing is shown in Figure 6.2-12. The first filter unit filters the input signal $s(n)$ using the coefficients according to the LTP parameters associated to the previous frame:

$$s'_{LTP}(n) = s(n) + c_{k-1}(n)\left(\sum_{i=1-l_{filt}}^{l_{filt}} b_{k-1,i}s(n-i) - \sum_{j=d_{LTP}'^{(prev)}-l_{filt}+1}^{d_{LTP}'^{(prev)}+l_{filt}} a_{k-1,j}s'_{LTP}(n-j)\right) \quad (6.2\text{-}27)$$

for $n = D_{LTP}..D_{LTP} + \frac{L_{out}}{4} - 1$.

Further:

$$s'_{LTP}(n) = s_{LTP}(n)$$

for $n = D_{LTP}-l_{filt}..D_{LTP} - 1$, which is possible as s_{LTP} is already available from the previous LTP post filtering for these samples and for these samples s_{LTP} is equivalent to s'_{LTP} used in the previous frame with $g_{LTP}^{(prev)}$ instead of $c_{k-1}(n)$.

Furthermore:

$$s'_{LTP}(n) = s(n)$$

for $n = D_{LTP} + \frac{L_{out}}{4} .. D_{LTP} + \frac{L_{out}}{4} + l_{filt} - 1$.

In Equation (6.2-27), $c_{k-1}(n)$ changes from the LTP gain in the previous frame towards 0 when n increases:

$$c_{k-1}(n) = \frac{L_{out} - 4(n - D_{LTP})}{L_{out}} g_{LTP}^{(prev)}$$

The filter coefficients are the same as used in [3]:

$$b_{k-1,-i} = h_{LTP}^{\left(i_{filt}^{(prev)}, 0\right)}(p_{res} i)$$

$$b_{k-1,i+1} = h_{LTP}^{\left(i_{filt}^{(prev)}, p_{res}\right)}(p_{res} i)$$

$$a_{k-1,d_{LTP}'^{(prev)}-j} = h_{LTP}^{\left(i_{filt}^{(prev)}, f_{LTP}'^{(prev)}\right)}(p_{res} j)$$

$$a_{k-1,d_{LTP}'^{(prev)}+1+j} = h_{LTP}^{\left(i_{filt}^{(prev)}, p_{res} - f_{LTP}'^{(prev)}\right)}(p_{res} j)$$

for $i, j = 0 .. l_{filt} - 1$, but according to OAO filtering and different to [3] they are scaled with $c_{k-1}(n)$ as shown in Equation (6.2-27). As in [3] the FIR filters are symmetric:

$$h_{LTP}^{\left(i_{filt}^{(prev)}, p_{res}\right)}(p_{res} i) = h_{LTP}^{\left(i_{filt}^{(prev)}, 0\right)}\left(p_{res}(i+1)\right) \text{ and } h_{LTP}^{\left(i_{filt}^{(prev)}, p_{res}\right)}\left(p_{res}\left(l_{filt} - 1\right)\right) = 0.$$

The second filter unit filters the output of the first filter unit $s'_{LTP}(n)$ using the coefficients according to the LTP parameters associated to the current frame:

$$s_{LTP}(n) = s'_{LTP}(n) + c_k(n)\left(\sum_{i=1-l_{filt}}^{l_{filt}} b_{k,i} s'_{LTP}(n-i) - \sum_{j=d'_{LTP}-l_{filt}+1}^{d'_{LTP}+l_{filt}} a_{k,j} s_{LTP}(n-j)\right) \quad (6.2\text{-}28)$$

for $n = D_{LTP} .. D_{LTP} + \frac{L_{out}}{4} - 1$.

In Equation (6.2-28), $c_k(n)$ changes from 0 to the LTP gain in the current frame when n increases:

$$c_k(n) = \frac{4(n - D_{LTP})}{L_{out}} g_{LTP}$$

The filter coefficients are the same as used in [3]:

$$b_{k,-i} = h_{LTP}^{(i_{filt}, 0)}(p_{res} i)$$

$$b_{k,i+1} = h_{LTP}^{(i_{filt}, p_{res})}(p_{res} i)$$

$$a_{k,d'_{LTP}-j} = h_{LTP}^{(i_{filt}, f'_{LTP})}(p_{res} j)$$

$$a_{k,d'_{LTP}+1+j} = h_{LTP}^{(i_{filt}, p_{res} - f'_{LTP})}(p_{res} j)$$

for $i, j = 0 .. l_{filt} - 1$, but according to OAO filtering and different to [3] they are scaled with $c_{k-1}(n)$ as shown in Equation (6.2-28). As in [3] the FIR filters are symmetric:

$$h_{LTP}^{(i_{filt}, p_{res})}(p_{res} i) = h_{LTP}^{(i_{filt}, 0)}\left(p_{res}(i+1)\right) \text{ and } h_{LTP}^{(i_{filt}, p_{res})}\left(p_{res}\left(l_{filt} - 1\right)\right) = 0.$$

where $\frac{L_{out}}{4}$ is the length of the initial subinterval (within the current frame k of length L_{out}) where the OAO smoothing is used.

6.2.2.4 Background noise estimation

6.2.2.4.1 General

Background noise envelope is estimated by analyzing the decoded mono downmix (core) signal during speech inactivity. The information about speech inactivity is inferred from the VAD flag, f_{VAD}. The VAD flag is deduced directly from the information in the bitstream and it's set as follows

$$f_{VAD} = \begin{cases} 0 & \text{if SID frame OR INACTIVE} \\ 1 & \text{otherwise} \end{cases} \quad (6.2\text{-}29)$$

This means that the VAD flag is set to 0 if the current frame is a SID frame or if the decoded frame type is INACTIVE (see clause 5.1.13 in [3] frame. In all other cases the VAD flag is set to 1.

6.2.2.4.2 Power spectrum compression

Let the decoded mono downmix (core) signal be denoted as $m_d(n)$ where $n = 0,..,N-1$ and $N = 320$ is the length of the decoded frame. The decoded mono downmix (core) signal is converted to the frequency domain with the DFT transform. The process of conversion to the freqency domain is illustrated in the schematic diagram in Figure 6.2-13.

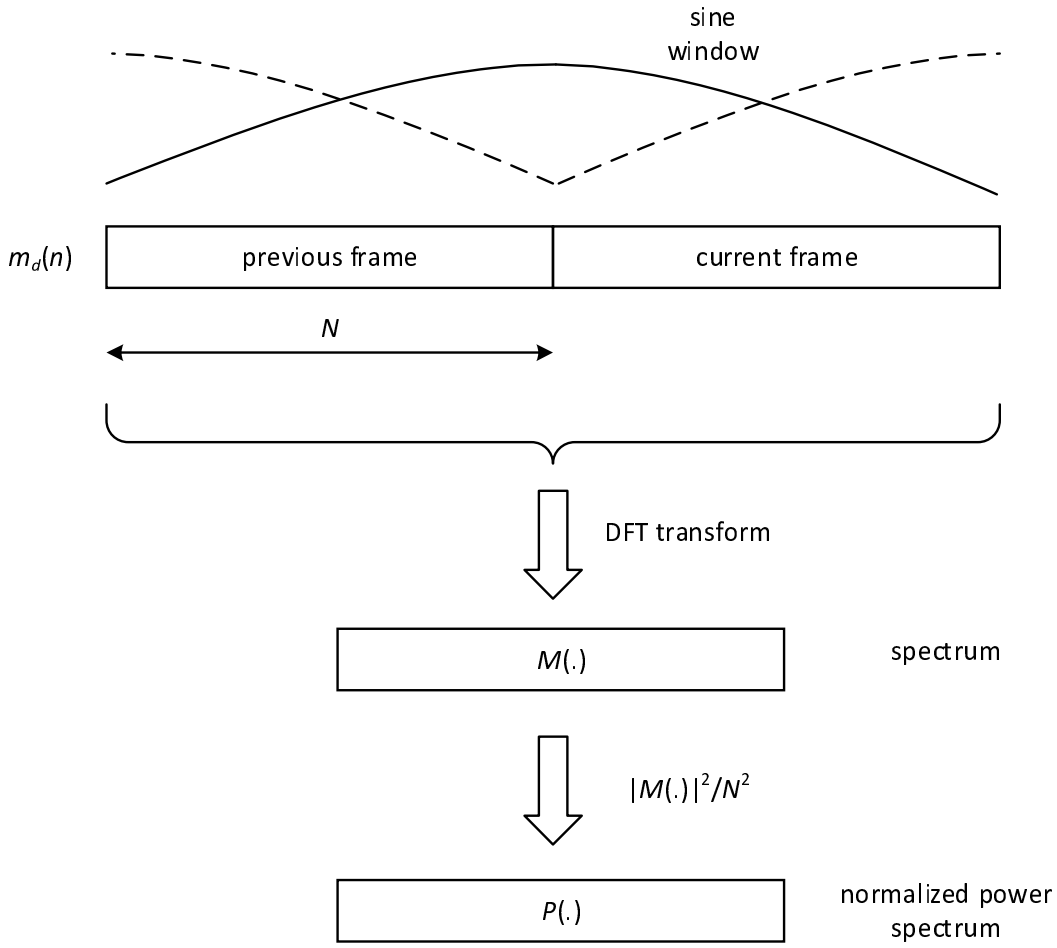


Figure 6.2-13: Conversion of Mono Downmix to Frequency Domain

The input to the DFT transform consists of the current frame and the previous frame. Therefore, the total length of the DFT transform is $2N$. The decoded core signal is first windowed with a normalized sine window. The sine window is defined as

$$w_s(n) = sin\left(\pi \frac{n+0.5}{2N}\right), \quad n = 0,\ldots,2N-1 \quad (6.2\text{-}30)$$

The sine window is normalized with

$$w_{sn}(n) = \frac{w_s(n)}{\sqrt{\frac{1}{2N}\sum_{n=0}^{2N-1} w_s^2(n)}}, \quad n = 0,\ldots,2N-1 \tag{6.2-31}$$

The decoded core signal is then windowed with the normalized sine window as

$$m_w(n) = m_d(n)w_{sn}(n), \quad n = 0,\ldots,2N-1 \tag{6.2-32}$$

The windowed signal is then transformed with the DFT transform as follows

$$M(k) = \sum_{n=0}^{2N-1} m_w(n) \cdot e^{-j2\pi\frac{kn}{2N}}, \quad k = 0,\ldots,2N-1 \tag{6.2-33}$$

Since the decoded core signal is real the spectrum is symmetric and only the first half, i.e. the first N spectral bins, are taken into account when calculating a power spectrum. This is done as follows

$$P(k) = \frac{1}{N^2}|M(k)|^2, \quad k = 0,\ldots,N-1 \tag{6.2-34}$$

Note, that the power spectrum is normalized to get the energy per sample. The normalized power spectrum is then compressed in the frequency domain by compacting spectral bins in frequency bands. Since the total number of samples in each frame of the decoded core signal is $N = 320$ the length of the FFT transform is $2N = 640$. Let's denote the total number of frequency bands as B. The process of compacting spectral bins in frequency bands is illustrated in Figure 6.2-14.

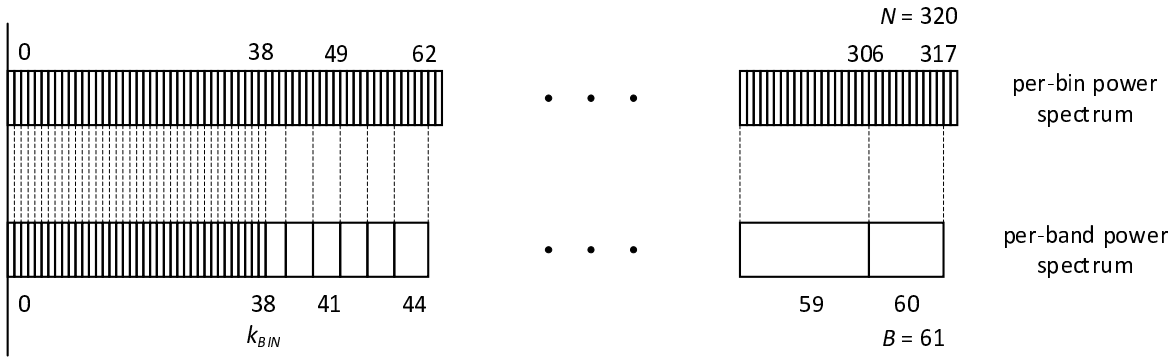


Figure 6.2-14: Power spectrum compression

During the compacting process 320 bins of the normalized power spectrum spanning the range of 0 Hz to 8 kHz are compressed into $B = 61$ frequency bands. In this partitioning scheme single-bin partitions are defined up to $f_{BIN} = 950$ Hz. Let's denote the index corresponding to this frequency as k_{BIN}. The last frequency index for bin-wise partitioning is set to $k_{BIN} = 38$. For low frequencies up to k_{BIN} no spectral compression is done and the bin-wise power spectrum is simply copied to the band-wise (compressed) power spectrum. This can be expressed as

$$N(k) = P(k), \quad k = 0,\ldots,k_{BIN} \tag{6.2-35}$$

For frequency bins above k_{BIN} the bin-wise power spectrum is compressed by means of spectral averaging. This is done by first calculating the variance in each frequency band

$$N_0(b) = \frac{1}{(k_{high}(b)-k_{low}(b)+1)}\sum_{k=k_{low}(b)}^{k_{high}(b)} P(k), \quad b = k_{BIN}+1,\ldots,B-1 \tag{6.2-36}$$

where b determines the frequency band and $\langle k_{low}(b), k_{high}(b)\rangle$ is a set of frequency bins corresponding to the bth band. The assignment of frequency bins to frequency bands is defined in 6.2-1.

Table 6.2-1: Power spectrum partitioning

band	lower bound k_{low}	upper bound k_{high}	middle point k_{mid}
0	0	0	-
1	1	1	-
2	2	2	-
...	...	...	-
37	37	37	-
38	**38**	**38**	**-**
39	39	41	40
40	42	45	43
41	46	49	47
42	50	53	51
43	54	57	55
44	58	62	60
45	63	67	65
46	68	72	70
47	73	78	75
48	79	84	81
49	85	91	88
50	92	98	95
51	99	106	102
52	107	115	111
53	116	124	120
54	125	134	129
55	135	146	140
56	147	174	160
57	175	210	192
58	211	254	232
59	255	306	280
60	307	317	312

6.2.2.4.3 Compensation for the loss of variance

The spectral averaging process in eq. (6.2-36) tends to reduce the variance of the background noise. To prevent the loss of variance a compensation method is in place adding random gaussian noise to the compressed power spectrum. This is done as follows. First, the variance in each frequency band is calculated as follows

$$\sigma(b) = \frac{1}{(k_{high}(b) - k_{low}(b) + 1)} \sum_{k=k_{low}(b)}^{k_{high}(b)} [P(k) - N_0(b)]^2 , \quad b = k_{BIN} + 1, \ldots, B - 1 \qquad (6.2\text{-}37)$$

The generated normal noise has zero mean and variance calculated in eq. (6.2-37) in each frequency band. Let's denote the generated gaussian noise as $\mathcal{N}(0, \sigma_b^2)$. The addition of the gaussian noise to the compressed power spectrum can then be expressed as

$$N(b) = N_0(b) + \mathcal{N}(0, \sigma_b^2), \; b = k_{BIN} + 1, \ldots, B - 1 \qquad (6.2\text{-}38)$$

The values of the compressed power spectrum are lower-limited to the threshold of 10^{-5}. The addition of the gaussian noise to the compressed power spectrum is only performed after the intialization period. This is explained later in this document.

6.2.2.4.4 Spectral smoothing

The compressed power spectrum is smoothed in the frequency domain by means of non-linear IIR filtering. Each frequency band is filtered with an IIR filter having a different weight, also called the forgetting factor. The weight is proportional to the ratio of energies between the compressed power spectrum in the current frame and the smoothed version. The energy of the compressed power spectrum in the current frame is calculated as

$$E_N = \sum_{b=0}^{B-1} N(b) + 10^{-7} \qquad (6.2\text{-}39)$$

Let's denote the smoothed compressed power spectrum as $\tilde{N}(b)$ with $b = 0, \ldots, B - 1$. The energy of the smoothed compressed power spectrum in the current frame is calculated as

$$\tilde{E}_N = \sum_{b=0}^{B-1} \tilde{N}(b) \qquad (6.2\text{-}40)$$

The energy ratio between the compressed power spectrum in the current frame and the smoothed version is then calculated as

$$r_{enr} = \frac{E_N}{\tilde{E}_N + \varepsilon} \tag{6.2-41}$$

where $\varepsilon = 10^{-7}$ is a small bias added to avoid the division by zero.

The IIR filtering operation on the compressed power spectrum depends on the VAD flag. As a general rule, the smoothing is stronger during inactive segments and weaker during active segments. For inactive segments, when the VAD flag is 0, the IIR smoothing is done as follows

$$\tilde{N}^{[m]}(b) = \begin{cases} 0.8 \cdot \tilde{N}^{[m-1]}(b) + 0.2 \cdot N^{[m]}(b), & \text{if } b \le k_{BIN} \text{ AND } N^{[m]}(b) < \tilde{N}^{[m-1]}(b) \\ 1.05 \cdot \tilde{N}^{[m-1]}(b), & \text{if } N^{[m]}(b) \ge 2\tilde{N}^{[m-1]}(b) \\ 0.95 \cdot \tilde{N}^{[m-1]}(b) + 0.05 \cdot N^{[m]}(b), & \text{otherwise} \end{cases} \tag{6.2-42}$$

where the index *m* in brackets has been added to denote the current frame. In the first line of eq. (6.2-42) fast downward update of the compressed power spectrum is performed in single-bin partitions. In the second line of eq. (6.2-42) only slow upward update is performed for all bands of the compressed power spectrum. The third line of eq. (6.2-42) represents is the default IIR filter configuration with $\alpha = 0.95$ for all cases other than those described by the two conditions in eq. (6.2-42).

For active segments, when the VAD flag is set to 1, the IIR smoothing is done as follows. If the energy ratio r_{enr} is lower than 0.5 the total energy of the compressed power spectrum in the current frame is significantly lower than the smoothed version. In this case, the smoothed compressed power spectrum is updated as follows

$$\begin{aligned} \tilde{N}^{[m]}(b) &= r_{enr} \cdot \tilde{N}^{[m-1]}(b) + (1 - r_{enr}) \cdot N^{[m]}(b), \\ b &= 0, \ldots, B-1 \text{ if } N^{[m]}(b) < \tilde{N}^{[m-1]}(b) \end{aligned} \tag{6.2-43}$$

Thus, in all bands where significant energy drop is detected in the current frame the energy of the smoothed compressed power spectrum is updated rather quickly, in proportion to the energy ratio r_{enr}.

If the energy ratio r_{enr} is higher or equal than 0.5 the smoothed compressed power spectrum is updated only at frequency bands above 2275 Hz. This corresponds to $b \ge 50$. First, short-term average of the compressed power spectrum is calculated as

$$\tilde{N}_{ST}^{[m]}(b) = 0.9 \cdot \tilde{N}_{ST}^{[m-1]}(b) + 0.1 \cdot N^{[m]}(b), \quad b = 50, \ldots, B-1 \tag{6.2-44}$$

where $\tilde{N}_{ST}^{[0]}(b) = 0$ for $b = 50, \ldots, B-1$. Note that the short-term compressed power spectrum is updated in every frame, regardless of r_{enr}. The smoothed compressed power spectrum is updated in frames where $r_{enr} \ge 0.5$ with

$$\tilde{N}^{[m]}(b) = 0.7 \cdot \tilde{N}^{[m-1]}(b) + 0.3 \cdot \tilde{N}_{ST}^{[m]}(b), \quad b = 50, \ldots, B-1 \text{ if } \tilde{N}_{ST}^{[m]}(b) < \tilde{N}^{[m-1]}(b) \tag{6.2-45}$$

Only downward update is allowed but the update is slower compared to the case when $r_{enr} < 0.5$.

Note, that the update of the smoothed compressed power spectrum is not performed during the initialization period. This is explained in the next section.

6.2.2.4.5 Initialization period

The background noise estimation process requires proper initialization. In the initialization process the smoothed compressed power spectrum is updated with a successive IIR filter. Let's first introduce the counter of consecutive frames where the smoothed compressed power spectrum has been updated as c_{CNI}. The counter c_{CNI} is initialized to 0 at the beginning of the decoding process. Let's also introduce a binary flag f_{CNI} for signaling whether the initialization process has been finished. This flag is also initialized to 0 at the beginning of the decoding process. The counter c_{CNI} and the flag f_{CNI} are updated with a simple state machine described in Figure 6.2-15.

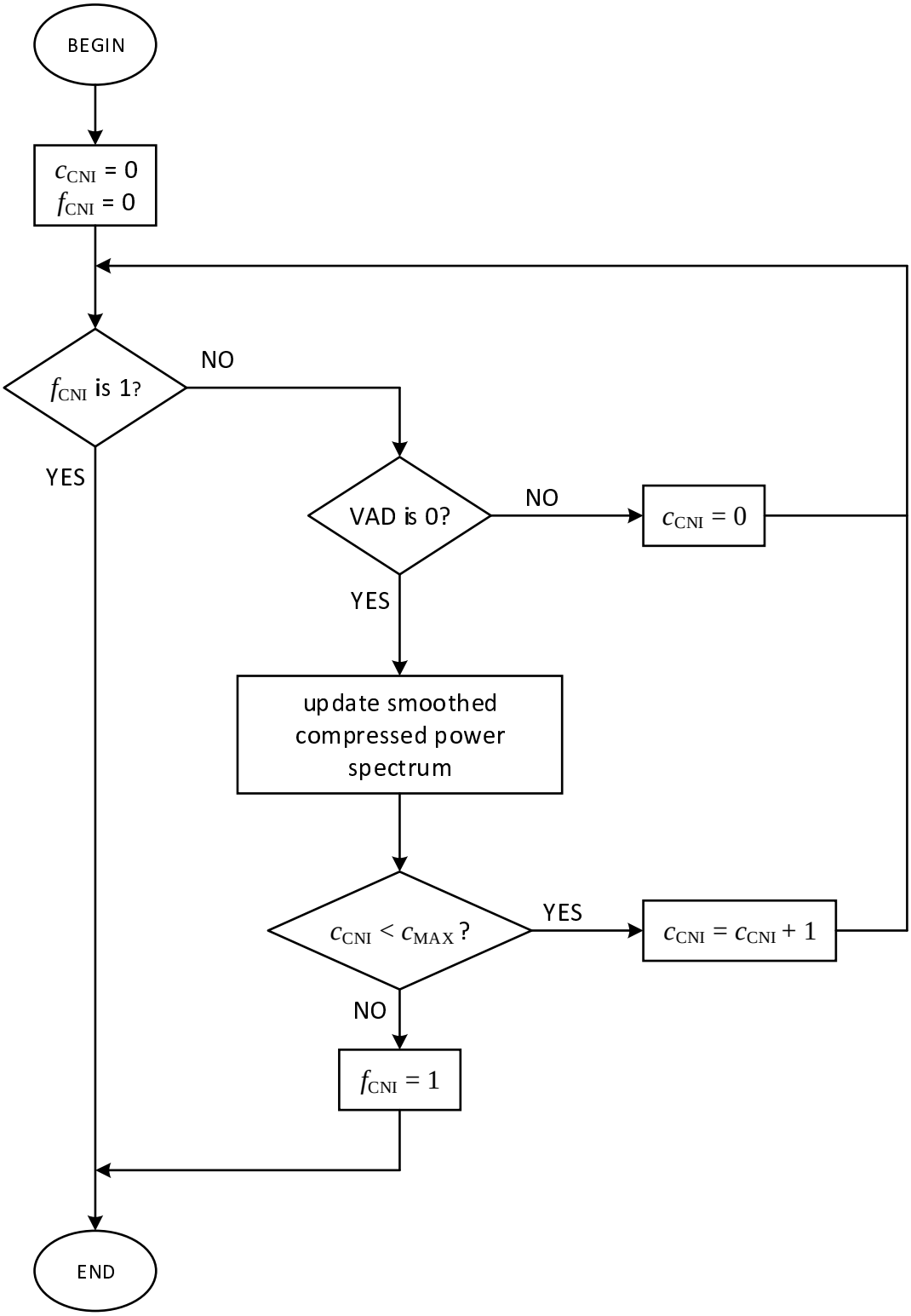


Figure 6.2-15: Initialization of the background noise estimation process

The initialization process is only finished when the smoothed compressed power spectrum has been updated in a sufficient number of consecutive inactive frames. This is controlled by the parameter c_{MAX} whic is set to $c_{MAX} = 5$. The smoothed compressed power spectrum is used for stereo comfort noise injection, described in clause 6.3.2.3.8.

During the initialization period the smoothed compressed power spectrum is updated with a successive IIR filter as follows

$$\widetilde{N}^{[m]}(b) = \left(1 - \frac{1}{c_{CNI}+1}\right) \cdot \widetilde{N}^{[m-1]}(b) + \frac{1}{c_{CNI}+1} \cdot \widetilde{N}_{ST}^{[m]}(b), \quad b = 0, \ldots, B-1 \qquad (6.2\text{-}46)$$

with $\tilde{N}_{ST}^{[0]}(b) = 0$ for $b = 0, \ldots, B-1$. Thus, the forgetting factor $\alpha = 1/(c_{CNI}+1)$ is proportional to the number of initialization frames. With this procedure the smoothed compressed power spectrum contains meaningful spectral information about the background noise even during the initialization period. In case of DTX operation it is possible to use the smoothed compressed power spectrum as an estimate of the background noise even before the initialization period is finished.

6.2.2.4.6 Power spectrum expansion

Similarly to the process of power spectrum compression described in clause 6.2.2.4.2 the inverse operation is the expansion of the smoothed and compressed background noise power spectrum. For low frequencies, up to k_{BIN}, no expansion takes places and the band-wise compressed power spectrum is copied to the bin-wise (expanded) power spectrum. That is

$$\tilde{P}(k) = \tilde{N}(k), \;\; k = 0, \ldots, k_{BIN} \tag{6.2-47}$$

For frequencies higher than k_{BIN} the band-wise compressed power spectrum is expanded by means of linear interpolation in the logarithmic domain. This is done by first calculating the multiplicative increment β_{mult}

$$\beta_{mult}(b) = \exp\left(\frac{log\left(\tilde{N}(b)\right) - log\left(\tilde{N}(b-1)\right)}{k_{mid}(b) - k_{mid}(b-1)}\right), \;\; b = k_{BIN}+1, \ldots, B-1 \tag{6.2-48}$$

where b determines the frequency band and $k_{mid}(b)$ the middle bin of the bth band. The expanded power spectrum is then calculated for all $b = k_{BIN}+1, \ldots, B-1$ as follows

$$\tilde{P}(k) = \tilde{N}(b-1) \cdot [\beta_{mult}(b)]^{\left(k - k_{mid}(b-1)\right)}, \;\; k = k_{mid}(b-1)+1, \ldots, k_{mid}(b) \tag{6.2-49}$$

Note, that the frame index [m] has been omitted for simplicity.

6.2.2.5 Switching coding modes

6.2.2.5.1 Improved handling of envelope stability parameter in HQ MDCT

In the decoding of the HQ MDCT core coding mode, the noise-fill is partly controlled by the envelope stability es, corresponding to $S^{[m]}$ as defined in equation (1839) in clause 6.2.3.2.1.3.2.3 of [3]. The envelope stability is determined based on the spectral envelope representation present only when operating the HQ MDCT core. To handle transitions to HQ MDCT from the ACELP and MDCT based TCX core coding modes, an estimation of es is done.

The core coding modes are selected by the encoder and signalled in the bitstream to the decoder. If both the mode of the current frame and the mode of the previous frame are HQ MDCT, the envelope stability es is determined according to:

$$es = \begin{cases} 1 - stab\_trans(I), & D_{LP}^{MDCT} < 2.571757 \\ stab\_trans(I), & D_{LP}^{MDCT} \geq 2.571757 \end{cases} \tag{6.2-50}$$

where $stab\_trans(I)$ is a lookup table and the index I is found by $I' = [\![|D_{LP}^{MDCT} - 2.571757| / 0.103108]\!]$ and clamping the index I' to the range $[0,9]$. This is a discretely sampled sigmoid function which maps D_{LP}^{MDCT} to the range $[0,1]$. D_{LP}^{MDCT} is a long-term estimate of the log energy variation determined according to:

$$D_{LP}^{MDCT} = \alpha D^{MDCT} + (1-\alpha)(D_{LP}^{MDCT})^{[-1]} \tag{6.2-51}$$

where $\alpha = 0.1$ is a low-pass filter coefficient and D^{MDCT} is calculated as:

$$D^{MDCT} = \sqrt{\frac{1}{N_{bands}} \sum_{b=0}^{N_{bands}-1} (I[b] - I^{[-1]}[b])^2} \tag{6.2-52}$$

where N_{bands} is the number of bands, $I[b]$ are the log energy indices corresponding to $I_M(b)$ in equation (1166) in clause 5.3.4.2.1.1 of [3] and D^{MDCT}, D_{LP}^{MDCT} corresponds to $D_M^{[n]}$, $\tilde{D}_M^{[n]}$ in equations (1837) and (1838) in clause 6.2.3.2.1.3.2.3 of [3].

If the current core coding mode is HQ MDCT and a transition occurred from the either the ACELP mode or MDCT based TCX mode to HQ MDCT, meaning the current mode is not equal to the previous mode, D_{LP}^{MDCT} cannot be calculated since $D_{LP}^{MDCT[-1]}$ and the log energy indices $I^{[-1]}[b]$ are outdated or do not exist. In this case D_{LP}^{MDCT} is estimated on the energy stability $E_{\Delta,LP}^{[-1]}$ and shape stability $\theta_{LP}^{[-1]}$ values from the previous frame:

$$D_{LP}^{MDCT} = D_{LP,est}^{MDCT} = P_1 + P_2\,\theta_{LP}^{[-1]} + P_3 E_{\Delta,LP}^{[-1]} \quad (6.2\text{-}53)$$

where P_1, P_2, P_3 are constants, set as $P_1 = 2.93$, $P_2 = -2.20$ and $P_3 = 0.741$ using a linear regression model. The energy stability of the current frame $E_{\Delta,LP}$ is determined in all core coding modes by using the definition as the long-term estimate of the absolute log energy difference according to:

$$E_{\Delta,LP} = \beta E_{\Delta} + (1-\beta)E_{\Delta,LP}^{[-1]} \quad (6.2\text{-}54)$$

$$E_{\Delta} = \left| E_{log} - E_{log}^{[-1]} \right| \quad (6.2\text{-}55)$$

$$E_{log} = \log_2 \frac{1}{L_{out}} \sum_{k=0}^{L-1} \hat{x}(;k)^2 \quad (6.2\text{-}56)$$

where $\hat{x}(;k)$ is the output synthesis of the current frame, L_{out} denotes the output synthesis frame length and β is a low-pass filter coefficient. The shape stability θ_{LP} of the current frame corresponds to $\bar{\theta}^{[0]}$ in equation (1542) in clause 6.1.4.2 of [3] and is calculated according to:

$$\theta_{LP} = \gamma\theta + (1-\gamma)\theta_{LP}^{[-1]} \quad (6.2\text{-}57)$$

where γ is a low-pass filter coefficient. In ACELP and TCX based MDCT core coding modes, the stability factor θ is computed based on a distance measure between adjacent LP filters as shown in clause 6.1.1.3.2 of [3]. In the HQ MDCT mode, the LP filter is not available and the stability factor for the current frame is estimated based on D_{LP}^{MDCT} and $E_{\Delta,LP}$ according to:

$$\theta = \theta_{est} = Q_1 + Q_2 D_{LP}^{MDCT} + Q_3 E_{\Delta,LP} \quad (6.2\text{-}58)$$

where Q_1, Q_2 and Q_3 are constants derived from a linear regression model and set as $Q_1 = 1.093$, $Q_2 = -5.84 \cdot 10^{-5}$ and $Q_3 = 0.125$.

Depending on whether a transition has occurred, the determined or the estimated envelope stability es is used in the decoding of the HQ MDCT frame.

6.2.2.6 DTX/CNG operation

6.2.2.6.1 General

DTX/CNG operation in the decoder works analogously to the encoder, i.e. most operations are the same as in EVS. General differences are described in this clause, while differences that are specific to an encoding mode are described in the respective encoding mode subclause.

6.2.2.6.2 FD-CNG VQ global gain dequantization

When decoding the global gain value for the FD-CNG MSVQ, instead of equation (1973) in clause 6.7.3.1 of [3], Equation (6.2-59) below is used:

$$g_{FD-CNG}^{[SID]} = \frac{I_{g,FD-CNG} - 45}{1.5} \quad (6.2\text{-}59)$$

6.2.2.6.3 First stage VQ decoding for FD-CNG

The MSVQ first stage was quantized as described in encoder clause 5.2.2.3.5.3 by employing a storage and complexity efficient mid-removed, upscaled and DCT-II(M =24) transformed structure. In the decoder the reverse process is applied to provide a single first stage FD-CNG domain vector $V_{0,idx_{rx}}(i)$, $i \in [0 \ldots M-1]$, of length M =24 based on the received $idx_{rx} \in [0 \ldots N_{st1} - 1]$. The decoder side inverse transformation is performed using the normalized IDCT-II(M =24) and the resulting signal is subsequently downscaled by the factor $FDCNG_{dctScale}$(= 0.4202880859375) and then added to the common mid-vector $N_{FD-CNG,midQ}^{dB}$ to produce vector $V_{0,idx_{rx}}$.

The received first stage MSVQ index idx_{rx} (denoted $I_{\mathrm{MSVQ,FD-CNG}}(0)$ in [3]) is first used to obtain a segment number $segm \in [0 \ldots N_{segm} - 1]$. The auxiliary vector $N_{c,segm} = \{16, 33, 50, 128\}$ containing the cumulative number of first stage vectors per segment allows the decoder to establish the segment number $segm$ and the local segment index idx as:

$$segm = \underset{s \in 0 \ldots N_{segm}-1}{\operatorname{argmax}} \; idx_{rx} < N_{c,s} \tag{6.2-60}$$

$$idx = \begin{cases} idx_{rx}, & segm = 0 \\ idx_{rx} - (N_{c,(segm-1)}, & segm > 0 \end{cases} \tag{6.2-61}$$

The segment number $segm$ is used to locate the tabled 8-bit mantissa vector $m_{idx,segm,col}$, with $idx \in \left[0 \ldots N_{cand,segm} - 1\right]$ and with coefficient indexes $col \in \left[0 \ldots N_{trunc,segm} - 1\right]$, and truncation lengths given by $N_{trunc,segm} = \{8, 10, 16, 18\}$. The integer shift factor vector for scaling the columns of each segment is denoted $exp_{segm,col}$, $col \in \left[0 \ldots N_{trunc,segm} - 1\right]$, the shift factors are common for each segment and is used to scale the columns of the compactly stored mantissas $m_{idx,segm}$ by a value of $2^{exp_{segm,col}}$. The truncation lengths of each segment are even, allowing the decoder to fetch the mantissas pairwise as 16-bit words, apply a bitmask to each coefficient and shift each coefficient according to the respective shift factor $exp_{segm,col}$ and $exp_{segm,col+1}$. The overall MSVQ first stage synthesis model is thus given by:

$$V_{0,\text{idx\_rx}} = N^{dB}_{FD-CNG,midQ,i} + FDCNG_{dctScale} \cdot IDCT_{M=24}\left(m_{idx,segm,col} \cdot 2^{exp_{segm,col}}\right) \\ , where\ col \in \left[0 \ldots N_{trunc,segm} - 1\right] and\ i \in [0 \ldots M-1] \tag{6.2-62}$$

In the case the FD-CNG decoder is in WB-operation mode, the first stage FD-CNG domain vector V_0 is truncated to M_{WB} =21 coefficients.

The final total MSVQ FD-CNG domain vector output $\widehat{N}^{dB}_{FD-CNG}$ is constructed using the generic stage summation Equation(1972) in clause 6.7.3.1.1 of [3], with $V_{0.\text{idx\_rx}}$ corresponding to $V_0(j = idx_{rx}, i)$,in [3], however the number of higher MSVQ stages may be different than in [3], and will vary depending on the actual IVAS SID format.

6.2.3 Common audio decoding tools

6.2.3.1 General

This clause describes common audio coding tools used for decoding of the transport channels. Similar to the encoder tools, the individual decoder tools can be used simultaneously multiple times similarly as a combination of different tools can be used simultaneously to decode different numbers of transport channels. The exact set-up depends on the actual IVAS format, IVAS total bitrate, and its actual mode and it is described in particular IVAS format decoder related clauses.

6.2.3.2 Single Channel Element (SCE) decoder

The Single Channel Element (SCE) decoder is a decoding tool that decodes one transport channel. It is based on one core-decoder module. Its block diagram is shown in Figure 6.2-16.

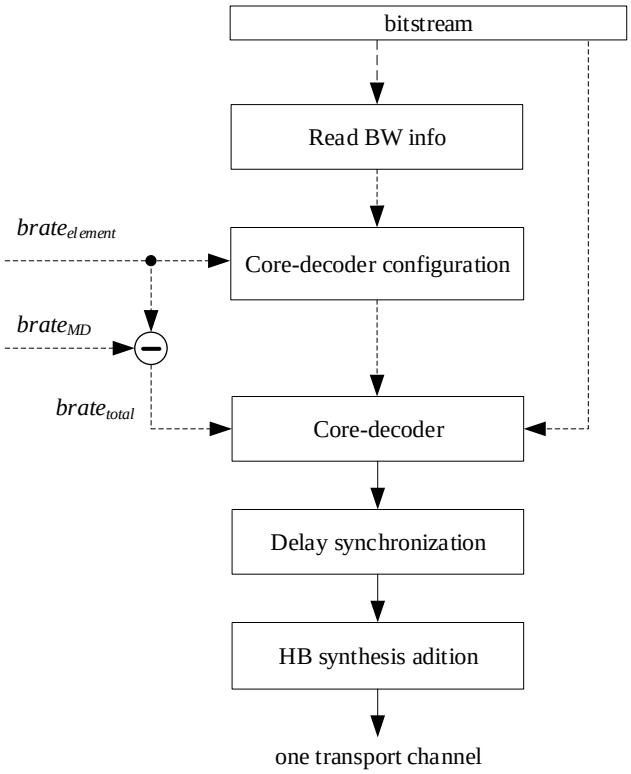


Figure 6.2-16: Block diagram of the Single Channel Element (SCE) decoder

As seen from Figure 6.2-16, the SCE decoder tool consists of the following modules:

(a) The audio bandwidth information is read from the bitstream.

(b) Core-decoder configuration module: high-level parameters like core-decoder internal sampling-rate, core-decoder nominal bitrate are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-decoder set-ups.

(c) One variable bitrate core-decoder (clause 6.2.2).

(d) Delay synchronization: the low-band synthesis is synchronized to be in line between different IVAS elements so it is delayed by $3.25 - 1.25 = 2.0$ ms.

(e) The high-band synthesis from BWE modules is synchronized between IVAS elements/formats so it is delayed by $3.25 - 2.3125 = 0.9375$ ms.

Note that the core-decoder module (c) depends on the core-decoder total bitrate $brate_{total}$ which is a difference between the element bitrate $brate_{element}$ and bitrate related to encode the metadata $brate_{MD}$ (if present) as defined in (5.2-241).

6.2.3.3 Channel Pair Element (CPE) decoder

The Channel Pair Element (CPE) decoder is a decoding tool that decodes two transport channels. It is based on one or two core-coder module(s) supplemented by stereo coding tools. Its block diagram is shown in Figure 6.2-17.

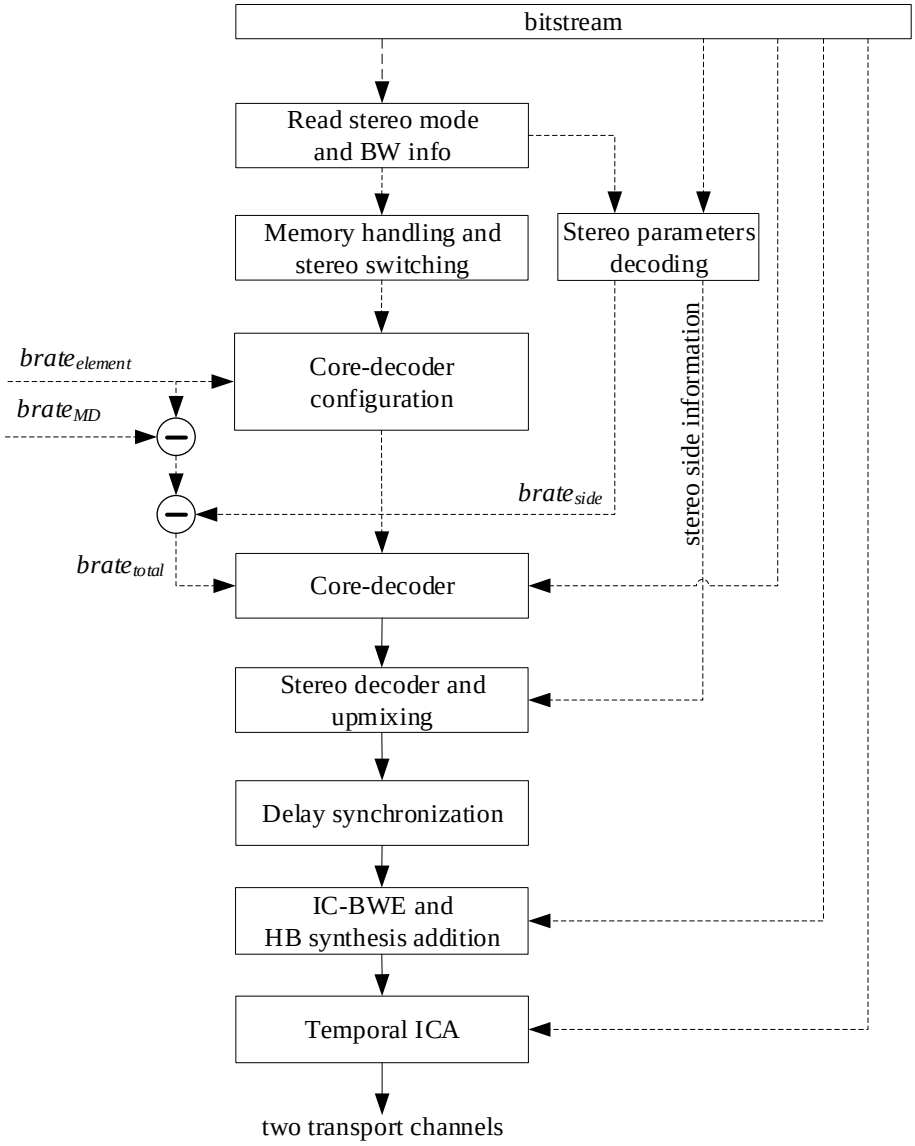


Figure 6.2-17: Block diagram of the Channel Pair Element (CPE) decoder

As seen from Figure 6.2-17, the CPE decoder consists of the following modules:

(a) The stereo mode and audio bandwidth information are read from the bitstream.

(b) Memory handling and stereo switching (clause 6.3.4) controls the switching between DFT stereo, TD stereo and MDCT stereo coding modes and comprises dynamic allocation/deallocation of static memory of stereo modes data structures depending on the current stereo mode. Also, the TD stereo mode is set in this module.

(c) Core-decoder configuration, one per transport channel: high-level parameters like core-decoder internal sampling-rate, core-decoder nominal bitrate etc. are set. Note that it depends on the element bitrate, $brate_{element}$, which is constant at one IVAS total bitrate, $brate_{IVAS}$, and ensures that there is no frequent switching between different core-decoder set-ups.

(d) One or two variable bitrate core-coder(s) (clause 6.2.2) performed sequentially on one or two downmixed channels.

(e) Stereo decoder and upmixing module in case of DFT stereo (clause 6.3.2.3) or TD stereo (clause 6.3.2.2) while the stereo parameters are decoded in the stereo parameters decoding module.

(f) Delay synchronization: the low-band synthesis is synchronized to be in line between different IVAS elements so it is delayed depending on the stereo mode: the delay is $3.25 - 3.125 = 0.125$ ms in case of DFT stereo, $3.25 - 1.25 = 2.0$ ms in case of TD stereo and MDCT stereo.

(g) Inter-Channel BWE (IC-BWE) decoder module (clause 6.3.2.1) decodes high-frequencies in DFT stereo and TD stereo modes and add the HB synthesis to the low-band synthesis.

(h) Temporal Inter-Channel Alignment (ICA) decoder module to time-align the two transport channels (clause 6.3.2.1).

Note that the core-decoder module (d) depends on the core-decoder total bitrate $brate_{total}[n], n = 1, 2$. The sum of total bitrates of both transport channels is then a difference between the element bitrate $brate_{element}$ and bitrates related to decode the metadata $brate_{MD}$ (if present) and stereo side information $brate_{side}$ (if present) as defined in (5.2-242).

6.2.3.4 Multichannel coding tool (MCT) decoder

6.2.3.4.1 General overview

The decoding process of MCT is very similar to the MDCT-stereo based decoder described in clause 6.3.3. After the decoding of the side bit information and the MCT parameters from the bitstream, the number of bits allocated to each channel can be determined with the decoded bitrate ratios and the exact same procedure as the encoder, described in clause 5.2.3.4.6. Then, inverse quantization of the spectra of the jointly coded channels, followed by the noise filling as described takes place. Afterwards, IGF or stereo IGF is applied. From the bitstream signaling, it is also known if there are channel-pairs that were jointly coded. The inverse processing should start with the last channel-pair block formed in the encoder in order to convert back to the original whitened spectra of each channel. For each channel pair the inverse stereo processing is applied base on the stereo mode and the band-wise M/S decision. For all channels that were involved in channel pair blocks and were jointly coded the spectrum is de-normalized to the original energy level based on the $\widehat{ILD}(k)$ energy normalization values that were sent from the encoder. Finally, TNS and SNS is applied to reconstruct the original spectra of the individual channels and the inverse MDCT transforms the signals to the time domain.

6.2.3.4.2 Core and SNS parameter decoding

For the side parameter decoding, first the channel mode flag bit is read setting the flag to REGULAR or IGNORE. That determines whether decoding of the bitstream and respective processing for a channel takes place or is skipped. Then, the decoding of the core parameter side bits -including the SNS decoding-is applied for each active channel (excluding the LFE channels if it exists) and the process is the same as for MDCT-stereo which is described in clauses 6.3.3.2.10-6.3.3.2.3. The active channels are assigned to $\frac{N}{2}$ CPEs, which are configured as MDCT-stereo CPEs per default. As the encoding of the core side parameters are written to the bitstream before the MCT channel pairing process, the decoded parameters represent the parameters for each active channel in sequential order $0, \cdots, N-1$.

6.2.3.4.3 MCT parameter decoding

The MCT parameters that are decoded needed to perform the inverse MCT processing, are the following in the order they are being read from the bitstream:

- The number of selected channel pairs $Block\_cnt$
- If $Block\_cnt \neq 0$ then for each active channel the $\widehat{ILD}$ is read, meaning the quantized level differences to the mean energy level, otherwise $\widehat{ILD}$ are set to *0*.
- The flag for signaling downscaling with bit *0* and upscaling with bit *1* for the normalization of each channel as described in the encoder subclause 5.2.3.4.4.2.
- Finally, the pairwise processing data is decoded. More precisely, for each channel pair block from $0, \cdots, Block\_cnt - 1$:
    - The channel pair index is read, which discloses the selected channels for the particular channel pair block as described in Table 5.2-28
    - The stereo information for the joint processing of the particular channel pair, which include:
        - The stereo mode for the core coding bands
        - The respective MS mask if the stereo mode is band-wise MS
        - If IGF is active the stereo mode for the IGF bands
        - The respective MS mask if the IGF encoding mode is band-wise MS

6.2.3.4.4 Bitrate distribution

The number of bits allocated to each channel must be known before fully decoding the bitstream and the spectral coefficients for each channel.

The bit ratio for each channel $\hat{r}_i$is decoded from the bitstream. After the decoding of the core coding parameters and the MCT side data, the number of bits $bits_{remaining}$ remaining to decode the spectral coefficients of each channel is known. The exact same procedure described for the encoder in subclause 5.2.3.4.6 is applied to retrieve the number of bits allocated to each channel $bits_i$.

6.2.3.4.5 Spectral data decoding and noise filling

Spectral data for each active channel are decoded using the range decoder described in 5.2.2.3.3.3 with the respective number of assigned bits $bits_i$. Then inverse quantization and noise filling is applied as for MDCT -stereo and as is described in the respective clauses 6.3.3.3.1.

6.2.3.4.6 Application of IGF and stereo IGF

IGF is applied on the whitened, ILD compensated spectral domain.

If from the decoded MCT parameters it is established, that either there are no channel pair blocks, i.e. $Block\_cnt = 0$, or some of the active channels are not part of a channel-pair block, then single channel IGF is applied as is described in clause 6.2.2.3.8 of [3] and IGF subclause of IVAS.

If there are channel-pair blocks, then the same conditions apply as described in MDCT-stereo and clause 6.3.3.3.2. If either the core stereo encoding mode or the IGF stereo encoding more is not DUAL_MONO, stereo IGF as described in 6.3.3.3.2.2 is applied. Otherwise, the IGF is applied to each channel separately.

6.2.3.4.7 MCT decoding

6.2.3.4.7.1 Inverse M/S processing of channel pairs

If there are channel pair blocks present, then for each block the inverse M/S processing is applied depending on the core stereo encoding mode and the IGF stereo encoding mode (if IGF is active) as described in the MDCT-stereo decoder clause in 6.3.3.4.1.

6.2.3.4.7.2 ILD de-normalization

To retrieve the original whitened signals that were input to the MCT at encoder side, the de-normalization towards the initial energies of the signals with use of the decoded quantized ILDs $\widehat{ILD}$ is applied. That is summarized in the following:

If $\widehat{ILD} \neq 0$ and if upscaling is flagged

$$\hat{a}_i = \frac{\overline{ILD}(i)}{ILD_{RANGE}} \tag{6.2-63}$$

if downscaling is flagged then the normalization factor is derived from

$$\hat{a}_i = \frac{ILD_{RANGE}}{\overline{ILD}(i)} \tag{6.2-64}$$

where $i = 0,1, \cdots, N_{ACTIVE} - 1$.

Then the spectrum of each channel is de-normalized by scaling with the de-normalization factor $\hat{a}_i$.

6.2.3.4.8 Noise shaping

As in MDCT stereo, after the retrieval of the whitened signals of all active channels, temporal and noise shaping is applied to reconstruct the original spectra. As described in clause 6.3.3.5, the TNS may be applied either prior or after the scaling of the MDCT spectrum with the SNS coefficients.

Detailed description of the SNS decoding and spectral shaping can be found in clause 6.3.3.5.2.

Detailed description of the TNS can be found in subclause 6.2.2.3.10 of [3].

6.2.3.4.9 Inverse MDCT transform to time domain

Finally, the signals are transformed back to the time domain using an inverse MDCT transform. In clause 6.2.4 of [3], the details of the processing including the inverse MDCT, overlap-add and windowing are described.

6.2.3.4.10 MCT PLC

As all MCT channel are coded in TCX MDCT mode, the according PLC selection and methods are used as described in clause 5.4.2 of [4].

All MCT-specific parameters – which in non-PLC frames are read from the bitstream and decoded as described in 6.2.3.4.3 are simply copied from the last received frame and re-used for the MCT decoder processing. Therefore, channel pair selection and bitrate distribution between channels will remain fixed until the next active frame is received.

6.2.4 Common spatial metadata decoding tools

6.2.4.1 Direction metadata decoding tools

6.2.4.1.1 Overview

The direction metadata parameters are part of the spatial metadata whose main configuration parameters have been presented in table 5.2-29. They consist of the azimuth and elevation values for each TF tile. The following clauses will present first the dequantization procedures and then the associated decoding for raw and entropy-based decoding respectively.

6.2.4.1.2 Direction dequantization tools

6.2.4.1.2.1 Joint azimuth-elevation dequantization

The joint azimuth elevation decoding presents the method for obtaining the azimuth and elevation values from one integer index. The codebook is a spherical grid, as defined at the encoder in clause 5.2.4.3.2.1 and there are spherical grids defined for 1 to 11 bits. The dequantization procedure has as input one integer index and the number of bits of the spherical grid for which the index has been generated.

For a 0-bit grid both the dequantized elevation and dequantized azimuth are zero. For a 1-bit grid the dequantized elevation is zero and the dequantized azimuth is zero for the index 0 and -180 for the index 1. For a 2-bit grid the dequantized elevation is zero and the azimuth is dequantized according to the procedure described in 6.2.4.1.2.2. For a grid defined for 3-11 bits the spherical index of the resulting point in the spherical grid, idx_{sph}, was formed by enumerating the points on the grid starting from the frontal direction at zero elevation and zero azimuth, enumerating the point on the circle for elevation zero, then the points on the circle corresponding to the first positive elevation value, then the points on the circle corresponding to the first negative elevation value, then second positive, second negative and so on. For the McMASA case, there is only the upper part of the spherical grid, therefore the index was created by enumerating the points from the consecutive circles in the upper hemisphere of the grid. The deindexing and dequantization procedures are summarized below:

1. Calculate the cumulated index offsets according to the format (McMASA / other case using full spherical grid)
2. Find the first cumulated index offset that is larger than the index to be deindexed and keep the track if it corresponds to an upper hemisphere circle or to a lower hemisphere circle; assign the elevation sign according to the position upper (positive) or lower (negative) hemisphere
3. The circle index (in absolute value) is the quantized elevation index, id_θ
4. Calculate the difference between the index and the offset obtained at previous point and assign it as quantized azimuth index
5. Dequantize the elevation index by multiplying the elevation index with the quantization step, $\boldsymbol{\Delta_\theta}$, corresponding to the number of bits used in the grid (see table 5.2-31)
6. Dequantize the azimuth index, id_ϕ using the procedure from clause 6.2.4.1.2.2.

6.2.4.1.2.2 Azimuth dequantization

The azimuth index, id_ϕ, obtained in the joint dequantization of the elevation and azimuth is used to obtain an azimuth value, ϕ, in the interval $-180 \leq \phi < 180$.

In the generic case the dequantized azimuth value is obtained as $\phi = id_\phi * \Delta_{\phi\, id_\theta} + shift$, where the value of $shift$ depends on the quantization index of the elevation, id_θ, and it is given by equation (5.2-265).

For the case of McMASA the absolute value of the decoded azimuth is in addition decompanded using the inverse compading function from clause 5.2.4.3.2.2:

find k = 0:4 such that $\hat{\phi}_{abs} \leq pB[k+1]$
calculate $decomp(\hat{\phi}_{abs}) = \left(pA[k] + \frac{pA[k+1]-pA[k]}{pB[k+1]-pB[k]}(\hat{\phi}_{abs} - pB[k])\right)$
The pA and pB arrays are defined in clause 5.2.4.3.2.2 and the choice of their values depend on the dequantized value of the elevation.

After decompanding, the dequantized azimuth value is assigned the same sign as the dequantized azimuth before the decompanding.

6.2.4.1.3 Direction metadata raw decoding

The raw decoding of direction metadata uses the joint azimuth-elevation dequantization tools presented in 6.2.4.1.2.1. It relies on the spherical grid definitions that are given for 1 to 11 bits presented in table 5.2-31 and for 16 bits. The grid corresponding to 16 bits is the grid used in the MASA metadata input format. The raw decoding transforms the spherical index into azimuth and elevation values. It can be used as single decoding method or in combination with other entropy encoding methods, for all the TF tiles or only for part of the TF tiles as presented next.

6.2.4.1.4 Direction metadata entropy decoding tools

6.2.4.1.4.1 Quasi uniform decoding

The decoding function, *decode_quasi_uniform(alphabet_size)* is defined in pseudo-code as:

```
bits = floor(log2(alphabet_sz))
thresh = 2 ^ (bits + 1) - alphabet_sz
value = read_bits(bits)
if (value >= thresh)
        value = (value * 2 + read_bits(1)) - thresh
return value
```

6.2.4.1.4.2 Extended Golomb Rice decoding

The extended Golomb-Rice decoding can be performed with the following pseudo-code, given as input the alphabet size and the extended GR parameter, that indicates the limit of the most significant bits.

```
Decode_extended_GR():
    msb_size = ( alph_size + ( 1 << gr_param ) - 1 ) >> gr_param;
    if ( msb_size <= 3 )
        value = decode_quasi_uniform( index, alph_size );
    else
        msb = 0;
        while ( ( msb < msb_size - 1 ) && ( read_bits(1) != 0 ) )
            msb++;

        if ( msb == msb_size - 1 )
            lsb = decode_quasi_uniform( alph_size - ( ( msb_size - 1 ) << gr_param ) );
        else
            lsb = 0;
            for ( i = 0; i < gr_param; i++ )
                lsb = ( lsb << 1 ) + read_bits(1);

        value = ( msb << gr_param ) + lsb;
```

6.2.4.2 Diffuseness and energy ratio decoding methods

6.2.4.2.1 Diffuseness decoding

This decoding method is the counter part of the encoding method described in clause 5.2.4.4.3.

If the *nbands* is equal to one, only one diffuseness value is transmitted is coded in binary code on 3 bits.

Otherwise, the *dif_use_raw_coding* 1-bit flag is read.

If it is equal to 0, entropy coding was used.

For *nbands* < 8, the next 1-bit *dif_have_unique_value* is read. If this flag is 1 only one value is decoded using the *decode_quasi_unform()* function using an alphabet of 8. The unique decoded diffuseness value is then copied to all the nbands. If *dif_have_unique_value* is zero, all diffuseness values consist only of two consecutive values. First the minimal diffuseness value is decoded *decode_quasi_uniform()* function using an alphabet of 8, then for each nbands an offset of 0 or 1 and coded on 1-bit binary code is decoded. The final diffuseness is then:

```
Diffuseness[b] = diff_min + offset[b]
```

For *nbands* ≥ 8, a Huffman coding strategy was used, and first an average diffuseness is decoded from a 3-bit binary code. The average diffuseness is adjusted for each frequency bands by decoding a unary code with leading 1 and terminating 0, and this up to 9 leading value, where the value 9 is amended by a 2-bit binary code. The decoded value is then used to update the average diffuseness value as follows:

```
Diffuseness[b] = diff_avq + (value[b]+1)/2 if value[b] % 2== 1
Diffuseness[b] = diff_avq - (value[b]/2) if value[b] % 2== 0
```

In of case *dif_use_raw_coding* is equal to 1, all nbands diffuseness values are decoded using *decode_quasi_unform()* function using an alphabet of 8.

As in the encoding, the minimum coded diffuseness index is computed. It is used to set the dif_index_max_ec using the subsequent entropy coding 1. If the minimum diffuseness is above 6, dif_index_max_ec = 7, otherwise dif_index_max_ec = 5.

6.2.4.2.2 Diffuseness and energy ratio decoding with two concurrent directions

In the case of two sets of directional parameters, the diffuseness (or the equivalent diffuse-to-total energy ratio) and the direct-to-total energy ratio parameters are decoded from the combined scheme described in clause 5.2.4.4.4. The process described here is the inverse of the encoding process. The decoding depends on if the metadata codec is in Ho-DirAC mode or in MASA format mode.

In case of Higher-order Dirac, a diffuseness ratio and direct-to-total ratio are decoded. First, diffuseness ratio index is decoded with methods in clause 6.2.4.2.1. Then, based on the diffuseness ratio index, the number of bits used to quantize the direct-to-total, dfRatio_bits, is determined. This is done with an equivalent function as in clause 5.2.4.4.4.1 fulfilling table 5.2-35. The direct-to-total ratio indices are then decoded from the bitstream using the diffuseness decoding method in clause 6.2.4.2.1 with the same adjustments as detailed in clause 5.2.4.4.4.2.

Next, diffuseness ratio is reconstructed by selecting the correct reconstruction level from a ROM table (detailed in table 5.2-33) based on the decoded diffuseness ratio index. This is followed with dequantizing direct-to-total with uniform scalar dequantizer between 0.0 and 1.0 with number of levels determined by the value of dfRatio_bits for the frequency band.

In case of MASA format mode, the following steps are done for each frequency band. First, diffuseness ratio index (of quantized diffuseness ratio r_{diff}) is decoded with methods in clause 6.2.4.2.1. Then, based on the diffuseness ratio index, the number of bits used to quantize the distribution factor r_{df}, dfRatio_bits, is determined. This is done with an equivalent function as in clause 5.2.4.4.4.1 fulfilling table 5.2-36. The distribution factor ratio indices are then decoded from the bitstream using the diffuseness decoding method in clause 6.2.4.2.1 with the same adjustments as detailed in clause 5.2.4.4.4.2.

Next, diffuseness ratio r_{diff} is reconstructed by selecting the correct reconstruction level from a ROM table (detailed in table 5.2-33) based on the decoded diffuseness ratio index. This is followed with dequantizing r_{df} with uniform scalar dequantizer between 0.5 and 1.0 with number of levels determined by the value of dfRatio_bits for the frequency band.

With r_{diff} and r_{df} decoded and reconstructed, both direct-to-total ratios are decoded with the equations

$$r_{dir1} = \frac{r_{df}}{1 - r_{diff}}$$
$$r_{dir2} = \left(1 - r_{diff}\right) - r_{dir1}$$

These two direct-to-total ratios for each frequency band are provided as an output from the metadata decoding.

In case of MASA format mode, modified direct-to-total energy ratios are generated from the decoded direct-to-total ratios for each time-frequency tile with two directional parameter sets using the process detailed in clause 5.2.4.4.4.3. The initial bits assigned (i.e., the quantization spatial resolution) for the decoding of the direction parameter values are based on the modified direct-to-total energy ratios instead of the initial decoded direct-to-total energy ratios. The obtained (as part of the encoded bitstream) direction parameter values are then decoded in clause 6.2.4.3 based on the quantization spatial resolution provide by the modified direct-to-total energy ratios.

6.2.4.3 Direction metadata decoding methods

6.2.4.3.1 Direction metadata decoding overview

This subclause describes the tools used for the spatial direction parameters decoding. They are lower level functions and procedures that are combined to form the main decoding tool of the spatial direction parameters. They include both raw or fixed rate and entropy or variable rate decoding tools, the main focus being the 3 decoding blocks corresponding to the encoding blocks EC1, EC2, EC3.

6.2.4.3.2 Decoding EC1

6.2.4.3.2.1 Introduction

A first 1-bit flag is read. If its value if 1 the raw decoding will be performed, otherwise, the frame-wise entropy decoding will follow.

6.2.4.3.2.2 Raw decoding

If the raw coding method was signalled, the direction azimuth and elevation are decoded from the lowest to the highest frequency band, and then for each time blocks.

In case of 2D scene, the elevation angle is set to zero and the azimuth codebook is selected depending of the decoded diffuseness index, using *decode_quasi_uniform()* function.

In case of 3D, both elevation and azimuth are decoded jointly reading and decode a spherical index per frequency band and time block.

6.2.4.3.2.3 Frame-wise entropy decoding

The direction parameters are decoded form the lowest transmitted frequency band *start_band* to the highest *nbands-1*. For each frequency band the previously diffuseness index is used to deduce the direction quantization resolution, except for the higher IVAS total bitrates for MASA, 384 kbps and 512 kbps, where the maximal resolution is used corresponding to a diffuseness index of 0.

If the diffuseness index is above the threshold *diffuseness_index_max_ec*, the direction parameters are raw coded in the bitstream, and the spherical indices are read and decoded for each time block according to the associated resolution in case of 3D. In case of 2D, *decoder_quasi_uniform()* is used to decode the *nblocks* azimuth indices associated to the derived resolution and elevation angles are set to zero.

For the other frequency bands with diffuseness indices below and equal to *diffuseness_index_max_ec*, the direction parameters are entropy coded.

The average elevation angle is first decoded in case of 3D, using *decode_quasi_uniform()* followed by *deorder()*. The extended GR parameter between 0 and 4 is then decoded using *decode_quasi_uniform()*. In case extended GR parameter is equal to 4, all the elevation quantized distanced from the average elevation angle zero. The average

elevation is then projected for each band and block to the respective codebook and angular resolution. Along with the diffuseness index, it will be used to select the codebook and alphabet of the azimuth. If the extended GR parameter is below 4, the distance in index from the projected elevation index is decoded using *decode_extended_gr().* The distance index is reordered before being added to the projected average elevation index and used for the inverse quantization of the elevation angle.

The azimuth angles are decoded in a second step, by decoding the average azimuth index using *decode_quasi_uniform()* and *deorder()*. The extended GR parameter between 0 and 5 is then decoded using *decode_quasi_unform().* In case the GR parameter is 5, all the azimuth distance index are set to zero, and the projected average azimuth indices are computed for each frequency bands and used for each band and block for the inverse quantization. For the GR parameter below 5, the extended_GR_decode() followed by the *reorder()* to adjust the projected average azimuth index, which later on used for the inversion quantization.

6.2.4.3.3 Decoding EC2

When the EC2 method is signalled in the bitstream, the next bits that are read give the information, for each sub band, if it was encoded with raw encoding or with entropy encoding. Next, for all entropy encoded sub bands the entropy encoded indices corresponding to the directional parameters are read, decoded and the number of read bits are counted. For the sub bands that were raw encoded, the number of bits corresponding to the original bit allocation according to the table 5.2-32 are counted. The number of bits read from the entropy coded data and the number of bits corresponding to the raw encoded sub bands are summed together and the sum is compared to the number of bits available for the considered frame and direction. If the sum is larger than the number of bits available, the bit allocation for the sub bands that have been marked as being raw encoded is gradually reduced by 1 bit for each TF-tile, until the resulting number of bits for the encoding of the directional parameters equals $bits\_allowed$. The gradual transition to the lower resolution is done mirroring the process done at the encoder in 5.2.4.5.2. After the second quantization resolution is obtained, the corresponding encoded indices of the raw encoded sub bands are read and decoded.

6.2.4.3.4 Decoding EC3

When the EC3 method is signalled in the bitstream, the bit allocations for the direction parameters need to be reduced by the same procedure as at the encoder and presented in clause 5.2.4.5.3.1. The EC3 decoding flow is presented below:

1. Calculate the allowed number of bits of the last sub band according to the predefined order
2. If the above calculated number of allowed bits is zero
    a. *last_band* = *nbands*-2
3. Else
    a. *last_band* = *nbands*-1
4. End
5. For each sub band *b = 1:last_band*
    a. Select the current sub band index according to the predefined order $o(b)$
    b. Calculate the allowed bits for current sub band $bits\_allowed(o(b)) = \sum_{j=1}^{M} bits\_dir\_red[o(b)][j]$
    c. If *nblocks* == 1 *use_common_direction* = 0; end
    d. Find the maximum value of the spatial direction parameters bit allocation for the sub frames of the current sub band
    e. If the maximum bit allocation over the sub frames is less or equal to 1 *use_common_direction* =1
    f. If the maximum bit allocation over the sub frames is between 1 and a threshold (3), read one bit and store it in *use_common_direction*; then reduce 1 bit from the bit allocation of the TF tile with maximum number of bits for the current sub band
    g. Update the elevation alphabets for the current bit allocation; for McMASA case only positive elevation values are counted.
    h. If *use_common_direction* == 1
        i. Read the direction parameters using one common direction and the differences in azimuth with respect to it.
    i. Else
        i. If (*nblocks* == 1) and the bit allocation of the first subframe is less than or equal to a threshold (5)

1. *fixed_rate* = 1

ii. Else

1. Read one bit and store it in fixed_rate

iii. End if

iv. If fixed_rate == 1

1. Use raw decoding for the current bit allocation for each TF tile of the sub band

v. Else

1. Decode the elevation values as a combination of a method giving the same elevation value for all sub frames, and a Golomb Rice decoding method of parameter 0 or 1, the information about the Golomb Rice parameter being transmitted on one bit; decoding of the elevation is done only for the sub frames where the bit allocation is larger than 2 bits, otherwise, the elevation is set to value zero.
2. Decode the azimuth using a combination of three decoding variants;

a. If at least one bit allocation for the TF tiles of the current band is less or equal to 1

i. For the non-degenerate case where the number of bits is 0 or 1, use Golomb Rice decoding with order equal to 1 for the TF tiles where the bit allocation is 2 bits, and a Golomb Rice with order equal to 2 for the rest of the cases.

b. Else

i. Decode the azimuth values using a combination between direct Golomb Rice coding with parameter 1 or 2 and a minimum removed Golomb Rice coding with parameter 0 or 1. The Golomb Rice parameter is read in both cases on one bit.

c. End if

vi. End if

j. End if

k. Count the number of bits read at the current sub band

l. If less than the allowed number of bits have been read

i. Redistribute them to the next sub band, following the same procedure as in step 1.e.i of the EC3 encoder flow

m. End if

n. If more than the allowed number of bits have been read

i. Greedily remove them from the bit allocation of next sub band.

o. End if

6. End for

The predefined order of taking the sub bands is deduced the same way as for the encoder presented in clause 5.2.4.5.3.1.

For the decoding of the azimuth values for the McMASA case, instead of the uniform distribution of the codewords, the codebook is {-135, -110, -30, 0, 30, 110, 135} and has the following indexes respectively {6,4,2,0,1,3,5}.

6.2.4.4 Coherence decoding

6.2.4.4.1 Spread coherence decoding

6.2.4.4.1.1 Spread coherence decoding for bitrates up to 256 kbps

The spread coherences are decoded sub band wise. Two different cases are observed: when the number of subframes *nblocks* is 4 and when the number of subframes is 1.

When the number of subframes is 4, the spread coherences have been encoded using a DCT transform for which only the first and second coefficients are transmitted to the decoder. The number of bits on which the DCT coefficient of order 0 is encoded is a function of the quantization index of the weighted average of energy ratio of the sub band as presented in table 5.2-40. The actual codewords' values depend on the variance of the decoded azimuth across the sub band, as presented in the same table from clause 5.2.4.6.1. Using the codebook length information, one or two joint codewords containing information of the DCT coefficients of order 0 for all sub bands, or for half and half of the sub bands, are read and decoded from a product code.

The DCT coefficients of order 1 are decoded with an average removed Golomb Rice decoder of order 0. The average index is decoded with a Huffman code. The decoded and average adjusted indexes are transformed using the inverse of the function in equation (5.2-269) and the resulting positive index is used in a codebook selected from the table 5.2-42. The codebook selection index is given in the Table 5.2-41 where the codebook index is specified for each sub band case of the 5, 8, 12, 18, or 24 sub bands. The second and third DCT coefficients are set to zero. The 4x4 inverse DCT transform is applied on the vector of decoded values resulting in the 4 decoded spread coherence values corresponding to each sub frame of the current sub band.

When the number of sub frames is 1, the number of codewords that have been used at the uniform scalar quantization of the spread coherence of each sub band are calculated. They are a function of the corresponding quantized energy ratio index, similarly to the encoder side. One bit is read to indicate whether a Golomb Rice decoder should be used or a product code. In the Golomb-Rice case, a minimum removed decoding approach is applied by reading first the Golomb Rice order, then the minimum index, followed by the difference indexes with respect to the minimum. The decoded indexes are added the decoded minimum value. The resulting indexes are uniformly scalar decoded into the spread coherence values.

6.2.4.4.1.2 Spread coherence decoding for 384 kbps and 512 kbps

The quantized spread coherence values were quantized at the encoder on 3 and 4 bits at 384 kbps and 512 kbps respectively. The same decoding method is used irrespective of the number of sub frames. Except for the number of bits, the decoding of the spread coherence parameters for these two bitrates follows the same flow, the one presented below:

1. Calculate the quantization step
2. For each subframe
    a. Read 1 bit signalling if minimum removed or average removed Golomb Rice coding was used
    b. If average removed method
        i. Read the average index on the corresponding number of bits (3 or 4)
        ii. Read the Golomb Rice order on 1 bit (1 or 0)
        iii. For each sub band
            1. Read the Golomb Rice (positive) codeword for the difference index
            2. Transform it using the inverse of the function from equation (5.2-269)
            3. Add the average index
            4. Reconstruct the quantized spread coherence value
        iv. End for

c. Else

i. Read the minimum index on the corresponding number of bits (3 or 4)

ii. Read the Golomb Rice order on 1 bit (1 or 0)

iii. For each sub band

1. Read the Golomb Rice codeword for the difference index

2. Add the minimum index

3. Reconstruct the quantized spread coherence value

iv. End for

d. End if

3. End for

6.2.4.4.2 Surround coherence decoding

6.2.4.4.2.1 Surround coherence decoding for bitrates up to 256 kbps

One surround coherence value is decoded for each sub band. In addition, for each sub band a significance measure of the surround coherence of the sub band is calculated, which is applicable across the TF tiles of the sub band. The significance measure is calculated according to the formula from equation 5.2-270 from the mirroring encoder clause. The significance measure is then quantized. Its quantization index is used to specify the codebook for the surround coherence of the corresponding sub band, according to table 5.2-43. The surround coherence decoded indexes are used in their corresponding codebooks.

As special case, if the sum of the number of codewords of the codebooks of all sub bands is zero, there is no need to read the surround coherence and it is set to zero. Otherwise, one bit is read to indicate the coding method. If the Golomb Rice method is signalled, a minimum removed Golomb Rice decoder is used. The Golomb Rice order is then read, followed by the minimum index decoded with a Golomb Rice decoder of order zero. The differences to the minimum index are decoded with a Golomb Rice decoder of variable order (0 or 1) that has been previously read. The decoded indexes, adjusted with the minimum value are transformed into decoded surround coherence values using the codewords from their corresponding codebooks.

If the product code method is signalled, a first check on the total number of sub bands is performed. If it is larger than 8, then a maximum surround coherence index value is read, with a Golomb Rice decoder of order 0. If the predefined number of codewords of each sub band codebook is larger than the maximum index value plus one unit, the corresponding number of codewords is assigned the maximum index value plus one. If the number of sub bands is less or equal to 8, the number of codewords remains unchanged. With these settings for the number of codewords, one or two joint product codes are decoded and the indexes for the decoded surround coherence values are obtained.

6.2.4.4.2.2 Surround coherence decoding for 384 kbps and 512 kbps

For the higher bitrates, 384 kbps and 512 kbps, the decoding of the surround coherence is used only for MASA format. The significance measure is calculated as in equation (5.5-4) and it takes into account each sub band and each sub frame. The decoding flow is similar to the one in clause 6.2.4.4.2.1, but an array of indexes is decoded for each sub band and each sub frame. The decoded indexes are then transformed into decoded surround coherence values using their corresponding codebooks.

6.2.4.5 DTX decoding

DTX decoding, common to SBA-DirAC and MASA, is the dual part of the DTX encoding described in clause 5.2.4.7. In the case of non-active frames and for the SID frames, 5 bands are transmitted for MASA and only 2 for SBA-DirAC. Only one direction per parameter band is transmitted, and a time resolution of 20 ms is used, i.e. only 1 time block is used. The total payload allocated to the SID sound field parameters is a maximum of 2.8 kbps, corresponding to the total SID bit rate of 5.2 kbps minus the 2.4 kbps SID of core-coding.

For MASA, one signalling bit is read to signal in case of 2 transport channels to know if DFT stereo or discrete CPE MDCT coding is used for the core-coding. Another bit is read to know whether the transmitted soundfield parameters are for a 2D or 3D scene.

First the diffuseness indices are decoded for each parameter bands using the *decode_quasi_uniform()* function with an alphabet of 4. In case of SBA-DirAC, the diffuseness indices are sufficient to decode the rest of the SID payload, while in MASA the same heuristic as done at the encoder side and described in clause 5.2.4.7 must be performed.

The directions, one DoA per parameter band and frame, are decoded in a second step. It is achieved by decoding the spherical indices in 3D coded by raw coding and decoding the azimuth angle by using the function *decode_quasi_uniform()* in case of 2D.

To avoid abrupt transitions between the new directions decoded in the current SID frame and the previous directions, and to increase the temporal resolution from 20 ms to the 5 ms resolution used in the upmix, a smoothing is performed at the time block resolution, i.e. 5 ms, by interpolating the two sets of directions by successively adding the Cartesian coordinate of the new direction to the previous direction at each time block step of the frame, and this for each parameter band. The 4 resulting Cartesian coordinates per parameter band associated to the 4 time blocks are transformed into polar coordinates for further processing.

6.2.5 Common filter bank operation tools

6.2.5.1 Complex Low-delay Filter Bank (CLDFB) analysis

The CLDFB analysis is mostly the same as in EVS. It is described in greater detail in Ref. [3], clause 5.1.2. Specifically, the transformed signal $U\ \ (l,k)$ is calculated from the time-domain signal $u\ \ (n)$as

$$U\ \ (l,k) = \sum_{n=0}^{N-1} p(n) u_d(kS-n) \exp\left(j\frac{\pi}{L}(l+0.5)(n-\frac{D}{2})\right). \qquad (6.2\text{-}65)$$

Here, $l = 0, 1, \ldots, L-1$ is the CLDFB frequency band index where $L = 60$, $N = 600$ is the length of the FIR prototype filter p, $S = 60$ is the stride, k is time-slot index, D = 299 is the analysis-synthesis (sample-by-sample).

In IVAS, the prototype function differs from EVS. The delay is increased to 5 ms. The prototype function is plotted in Figure 6.2-18.

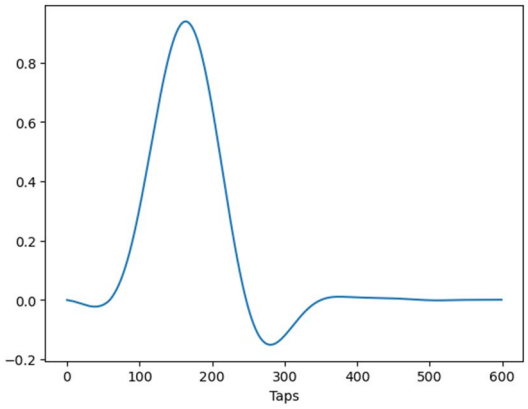


Figure 6.2-18 CLDFB prototype function in IVAS

6.2.5.2 Complex Low-delay Filter Bank (CLDFB) synthesis

The CLDFB synthesis is the inverse transform of the analysis in clause 6.2.5.1. It is described in greater detail in Ref. [3], clause 6.9.3. The change prototype function and delay follow from clause 6.2.5.1.

6.2.6 CLDFB domain decorrelator

6.2.6.1 Overview

The decorrelators in the CLDFB domain are implemented as IIR-filters with pre-delay with a distinct filter for each CLDFB band. For each channel of a number of channels $n_{ch,decorr}$ to be correlated a different filter set is chosen to produce orthogonal signals, i.e. signals that are mutually decorrelated. The decorrelated signals are energy adjusted to closely match the energy of the input signals.

An onset detection and removal of onset portions in a signal that shall not be decorrelated is applied to the signals to be correlated and optionally the onset portions are added back to the final decorrelated signal after the energy adjustment of the decorrelated signals when a flag f_{onsets} is set to 1.

The signal flow for decorrelating an arbitrary input signal Y_{proto} that consists of the CLDFB signal of one time slot of all channels to be correlated is depicted in Figure 6.2-19.

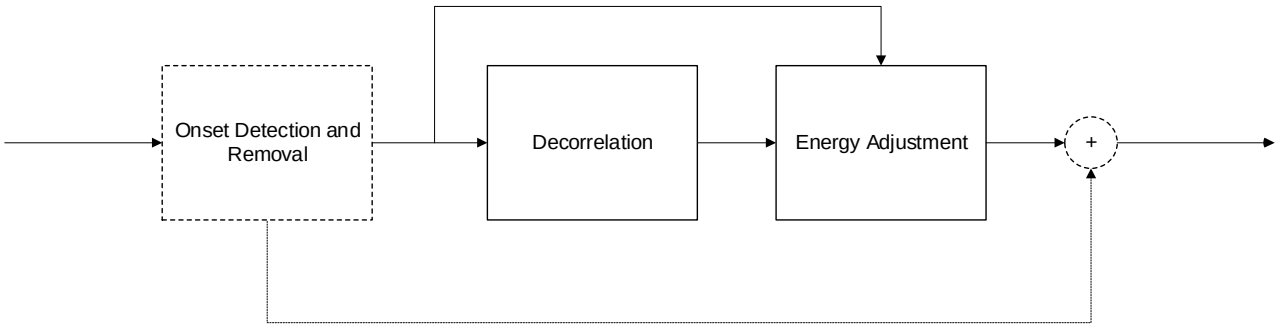


Figure 6.2-19: CLDFB Decorrelator signal flow

The processing is done only up to a maximum decorrelation band determined from the maximum decorrelation band $n_{B,decorr,init}$ set by the module using the decorrelator and the output band $n_{B,S}$ width in terms of CLFDB frequency bands:

$$n_{B,decorr} = \min(n_{B,S}, n_{B,decorr,init}) \tag{6.2-66}$$

6.2.6.2 Onset detection and removal

The onset detector estimates a set of onset filters g_{onsets} that describe the amount of the input signal belonging to the onset. The number of CLDFB bands for which the onset filter is calculated $n_{B,onsets}$ can differ from the number bands to be decorrelated depending on the module that uses the decorrelator.

The onset filters are calculated using the energy of the input signal P_{input}:

$$P_{input,ch}(n) = \left|Y_{proto,ch}(n)\right|^2, 0 \le ch \le n_{ch,decorr}, 0 \le n \le n_{B,onsets} \tag{6.2-67}$$

The minimum and maximum power envelopes are calculated:

$$\begin{aligned} Env_{max,ch}(n) &= \max\left(\alpha Env_{max,ch}{}^{[-1]}(n), P_{input,ch}(n)\right) \\ Env_{min,ch}(n) &= \min\left(\beta Env_{min,ch}{}^{[-1]} + (1-\beta)Env_{max,ch}(n), Env_{max,ch}(n)\right)' \\ 0 \le ch \le n_{ch,decorr}&, 0 \le n \le n_{B,onsets} \end{aligned} \tag{6.2-68}$$

Where: $^{[-1]}$ denotes the value in the previous CLDFB time slot

$\alpha = 0.95$ is the maximum envelope smoothing factor

$\beta = 0.995$ is the minimum envelope smoothing factor

The final onset filter is calculated:

$$g_{onsets,ch}(n) = \max\left(0, \min\left(1, \frac{4Env_{min,ch}(n)}{Env_{max,ch}(n)}\right)\right),\ 0 \le ch \le n_{ch,decorr}, 0 \le n \le n_{B,onsets} \quad (6.2\text{-}69)$$

The onset portion Y_{onsets} and the final signal to be correlatd $Y_{decorr,in}$ are calculated:

$$\begin{aligned} Y_{onsets,ch}(n) &= (1 - g_{onsets,ch})(n)Y_{proto,ch}(n) \\ Y_{decorr,in,ch}(n) &= g_{onsets,ch}(n)Y_{proto,ch}(n) \end{aligned},\ 0 \le ch \le n_{ch,decorr}, 0 \le n \le n_{B,decorr} \quad (6.2\text{-}70)$$

At initialization, the previous power envelopes $env_{max,ch}{}^{[-1]}$, $env_{min,ch}{}^{[-1]}$ are set to zero.

6.2.6.3 De-correlation

6.2.6.3.1 Overview

The decorrelation filters consist of real-valued IIR filters with a final complex phase rotation and a pre-delay. The decorrelation frequency region is divided into subsets of CLDFB bands, where each subset has the same length for the IIR filters and the same pre delay.

The subsets are calculated at initialization and the calculation starts with the following vector of split frequency bands:

$$n_{B,split} = \{0,7,22,0\}$$

The final split frequency vector and the number of sub sets $n_{sets,decorr}$ used in the decorrelation is determined by the following pseudo code:

```
n_sets,decorr = 0
for k=1,2,3:
    n_sets,decorr = n_sets,decorr + 1
    if n_B,split(k) ≥ n_B,decorr:
        n_B,split(k) = n_B,decorr
        break
```

The CLDFB band sub sets k are then:

$$k_s = n_{B,split}(s), ..., n_{B,split}(s+1)\text{-}1 \quad (6.2\text{-}71)$$

6.2.6.3.2 Derivation of filter coefficients from lattice coefficients

The filter coefficients define stored as lattice coefficients with the filter lengths L_f are defined as:

$$L_f(n) = \begin{cases} 15, n \in k_0 \\ 6, n \in k_1 \\ 3, n \in k_2 \end{cases} \quad (6.2\text{-}72)$$

The filter coefficients are defined as lattice coefficients in Tables 6.2-2, 6.2-3, 6.2-4.

Table 6.2-2: lattice coefficients $\Phi_{x,ch}$ for set k_0

ch	$\Phi_{x,ch}$
0	{0.795329, 0.502700, 0.204456, 0.416566, 0.459648, 0.270454, -0.201944, 0.027997, 0.067811, -0.052627, -0.038779, -0.057387, 0.020480, 0.367697, -0.593705}
1	{0.533667, 0.202500, -0.001953, 0.195081, -0.184458, -0.233867, 0.228100, -0.329293, -0.338714, -0.079700, 0.052389, -0.009468, 0.178807, 0.190843, -0.478877}
2	{0.044022, 0.788498, 0.133529, -0.173657, 0.545391, 0.681431, 0.332868, 0.294685, 0.325261, 0.047617, 0.157401, 0.116272, 0.218980, -0.189248, -0.317401}
3	{-0.753681, 0.772385, -0.507384, 0.276980, -0.692775, 0.232302, -0.354759, -0.015620, -0.223059, 0.310506, -0.035034, 0.085783, -0.499998, 0.215260, 0.201415}
4	{-0.495551, -0.301660, 0.196510, 0.326147, -0.594364, 0.314921, 0.668671, 0.076643, 0.045711, -0.124790, -0.203272, -0.297190, 0.125806, -0.179483, -0.201757}
5	{0.161128, 0.396050, 0.266897, 0.356586, 0.488145, -0.056254, 0.139280, -0.296405, -0.112844, 0.037405, -0.367425, -0.216292, -0.277360, 0.389420, 0.115115}
6	{-0.240449, -0.271015, 0.426720, -0.011059, 0.151813, 0.253490, 0.225764, 0.498716, -0.136377, 0.443004, -0.305017, -0.031310, -0.010765, 0.170349, 0.496478}
7	{-0.139875, -0.241998, -0.104850, 0.294343, -0.067728, -0.492202, -0.487610, 0.036395, 0.109393, 0.396155, -0.352845, -0.205913, -0.082999, -0.463033, -0.309296}
8	{0.025886, -0.092456, -0.125139, -0.156117, -0.004887, 0.178440, 0.089586, 0.044827, 0.238219, -0.312120, -0.390688, -0.178543, 0.454418, 0.387012, -0.388874}
9	{-0.197797, 0.035540, 0.455388, -0.054410, 0.380035, 0.290964, 0.048804, 0.078637, 0.221740, -0.217548, 0.121289, -0.396681, -0.218482, -0.127265, -0.269507}
10	{-0.344218, -0.465038, -0.421415, -0.026031, 0.221547, 0.361993, -0.348243, 0.294983, 0.366175, 0.070663, -0.086050, 0.252129, 0.156066, -0.062800, 0.408972}
11	{0.242461, -0.301764, -0.066160, 0.388651, -0.462227, -0.158880, 0.230796, -0.093179, 0.047076, 0.073402, -0.335018, 0.022940, 0.354611, 0.072391, 0.019473}
12	{0.336900, -0.480534, 0.170267, 0.259663, -0.393576, -0.348588, -0.108962, 0.278842, 0.385490, 0.362249, -0.318739, 0.362305, 0.288936, 0.291204, -0.278019}
13	{-0.449302, -0.288513, -0.413973, -0.405279, -0.295152, -0.245110, 0.002530, 0.287890, 0.348229, -0.178354, -0.206517, 0.351081, -0.482205, 0.021360, -0.492207}
14	{0.048642, 0.453282, 0.109160, 0.232599, -0.473781, 0.140502, 0.352527, -0.098606, 0.147172, -0.055797, 0.107739, -0.231026, 0.357310, 0.348031, 0.232404}
15	{-0.324788, 0.162480, 0.057647, -0.060734, -0.009742, -0.224185, -0.282355, -0.065443, 0.064697, 0.280370, -0.284906, -0.470501, 0.019484, -0.442308, 0.377214}
16	{0.447384, -0.290498, 0.345528, 0.370207, -0.313120, 0.119592, 0.300014, 0.406995, -0.277922, 0.447039, 0.194824, 0.157703, -0.223402, -0.147167, 0.379073}
17	{-0.216342, 0.137967, -0.397180, 0.073905, -0.273110, -0.443037, -0.168327, 0.346264, 0.037543, 0.065387, 0.163901, -0.122523, 0.365477, -0.316321, 0.117273}
18	{-0.301282, 0.169625, -0.336466, 0.269914, -0.420160, -0.331296, 0.498523, -0.393009, -0.462184, -0.323097, 0.470977, -0.359463, 0.264315, 0.216797, 0.493400}
19	{0.151489, -0.321653, 0.464413, -0.355673, 0.420401, 0.184297, -0.302128, 0.136536, -0.252849, 0.193906, 0.298775, 0.238808, -0.386298, -0.181999, -0.077326}
20	{0.124840, -0.263394, 0.389606, 0.004502, 0.339804, 0.415204, 0.377751, 0.400221, 0.352426, -0.003808, -0.184530, 0.433348, -0.237554, 0.147684, 0.407210}
21	{0.067616, -0.249313, -0.354010, 0.320937, 0.426000, 0.309576, -0.189689, -0.209447, 0.158967, -0.081929, 0.386828, 0.178582, -0.407143, 0.222189, -0.149097}

Table 6.2-3: lattice coefficients $\Phi_{x,ch}$ for set k_1

ch	$\Phi_{x,ch}$
0	{ 0.633692, 0.681207, -0.049418, 0.286715, 0.146022, 0.135402}
1	{ -0.410145, -0.206766, -0.656968, -0.101746, 0.436299, 0.339818}
2	{ -0.131383, -0.773746, -0.301627, 0.327561, 0.332227, 0.205858}
3	{ 0.717602, -0.552370, -0.150136, 0.054556, 0.239519, -0.648477}
4	{ -0.721848, 0.189377, 0.068185, 0.006216, 0.077025, 0.038678}
5	{ 0.423415, -0.108658, 0.432050, -0.414641, 0.277840, 0.418486}
6	{ 0.077811, -0.283063, -0.357008, 0.004634, -0.442640, -0.372315}
7	{ 0.403336, 0.243023, 0.314367, 0.444513, 0.347517, 0.496043}
8	{ 0.108408, 0.469382, -0.170286, 0.326310, 0.172157, 0.440334}
9	{ 0.058417, 0.339673, -0.194965, 0.491219, -0.281296, -0.043120}
10	{ -0.482487, -0.335005, -0.336159, 0.196469, -0.164623, 0.442491}
11	{ -0.135190, -0.354385, 0.452133, -0.311221, -0.347640, 0.498342}
12	{ -0.288999, 0.376431, -0.177924, 0.195542, 0.333872, -0.152697}
13	{ 0.053230, 0.457767, 0.442476, 0.235254, -0.345159, -0.286098}
14	{ 0.155438, 0.405393, 0.217073, -0.107960, 0.010698, 0.368987}
15	{ -0.498358, -0.495816, -0.215479, -0.093869, 0.320276, -0.013842}
16	{ 0.489406, 0.470814, -0.065834, 0.346871, 0.027279, 0.150086}
17	{ 0.484013, -0.497391, 0.168796, -0.493841, -0.173528, 0.334676}
18	{ 0.264235, -0.424651, -0.314926, 0.253086, 0.397381, -0.491565}
19	{ -0.453727, -0.463358, -0.019128, 0.000344, 0.315432, 0.472345}
20	{ 0.095139, 0.283375, -0.225088, -0.119762, -0.476871, 0.037525}
21	{ 0.336951, 0.494511, -0.062603, 0.177652, 0.463892, 0.489286}

Table 6.2-4: lattice coefficients $\Phi_{x,ch}$ for set k_2

ch	$\Phi_{x,ch}$
0	{ 0.018977, -0.212205, 0.422719}
1	{ -0.400657, -0.106890, -0.024589}
2	{ 0.140005, 0.279582, 0.032357}
3	{ 0.632535, 0.578535, -0.734606}
4	{ 0.017182, 0.013244, -0.027715}
5	{ -0.353356, -0.482160, -0.491265}
6	{ 0.457024, 0.165122, 0.469723}
7	{ -0.195705, 0.440105, -0.477366}
8	{ 0.360186, -0.490565, 0.484623}
9	{ -0.173791, 0.007543, 0.278186}
10	{ 0.434416, 0.060363, -0.193717}
11	{ -0.033709, 0.496222, 0.002939}
12	{ -0.480848, -0.109552, -0.023198}
13	{ 0.324679, -0.292075, -0.356148}
14	{ -0.366595, 0.380917, -0.301741}
15	{ 0.110318, 0.383789, 0.303984}
16	{ -0.499685, -0.349584, 0.334749}
17	{ -0.020224, -0.430078, -0.154705}
18	{ -0.371129, 0.334080, 0.346913}
19	{ -0.166781, -0.229089, 0.117956}
20	{ 0.341292, 0.490463, 0.493655}
21	{ -0.367726, 0.426528, -0.045774}

The reflection coefficients are converted into the direct form filter coefficients $a_{l,ch}(n)$ and $a_{l,ch}(n)$ according to:

$$\begin{aligned} a_{l,ch}(n) &= \alpha_{p,l,ch}(n) \\ b_{l,ch}(n) &= \left(a_{L_f(n)-l,ch}(n)\right), 0 \le l < L_f(n), p = L_f(n) \end{aligned} \quad (6.2\text{-}73)$$

Where $\alpha_p(l)$ are filter coefficients for a filter of order p, given the following recursion:

$$\begin{aligned} \alpha_{p,1,ch}(n) &= 1 \\ \alpha_{p,p,ch}(n) &= \Phi_{p-1,ch}(\mathrm{n}) \\ \alpha_{p,i,ch}(n) &= \alpha_{p-1,i} + \Phi_{i-1,ch}(\mathrm{n})\alpha_{p-1,p-i,ch}(n) \end{aligned} \quad , 1 \le i \le p-1, 1 \le p \le L_f(n) \quad (6.2\text{-}74)$$

Where: $\Phi_{i-1,ch}(\mathrm{n})$ are the lattice coefficients belonging to the set band n belongs to from tables 6.2-2, 6.2-3, 6.2-4.

6.2.6.3.3 Decorrelator IIR Filtering

The delayed subband samples are obtained as:

$$Y_{decorr,in,delay,ch}(n) = \begin{cases} Y_{decorr,in,ch}(n-7), n \in k_0 \\ Y_{decorr,in,ch}(n-2), n \in k_1 \\ Y_{decorr,in,ch}(n-1), n \in k_2 \end{cases} \quad (6.2\text{-}75)$$

Where $Y_{decorr,in,ch}(n)$ for $n < 0$ contains the buffered values of previous time slots. The delayed subband samples are filteres as:

$$Y_{decorr,ch}(n) = e^{j\Delta\phi_{ch}(n)} \frac{1}{a_{0,ch}(n)} \left(\sum_{l=0}^{L_f(n)} b_{l,ch}(n) Y_{decorr,in,delay,ch}(n-l) - \sum_{l=1}^{L_f(n)} a_{l,ch}(n) Y_{decorr,ch}(n-l) \right) \quad (6.2\text{-}76)$$

The final phase rotation values are defined as:

$$\Delta\phi_{ch}(n) = \Delta\phi_{ch,init}(n - k_s(0)) \quad (6.2\text{-}77)$$

Where: $\Delta\phi_{ch,init}$ are the initialization phase values from table 6.2-5

$k_s(0)$ is the index of the first band of the split band set n belongs to

Table 6.2-5: initialization phase values $\Delta\Phi_{ch,init}$

ch	$\Delta\phi_{ch,init}$
0	{ 1.802519, 0.922986, 1.813685, 1.272828, 0.856928, 0.366571, 1.531249, 1.318158, 0.123812, 0.897173, 0.958696, 1.256384, 0.179677, 0.668918, 1.440292, 1.573058, 1.396481, 1.191463, 0.444143, 1.666942}
1	{ 1.273955, 1.747171, 1.408330, 1.002782, 1.559302, 1.782992, 1.474896, 0.813181, 1.457724, 0.588531, 1.384302, 0.156493, 0.600048, 1.661632, 0.538958, 0.645429, 0.565237, 0.024684, 0.264229, 0.062140}
2	{ 1.235343, 0.851725, 1.820211, 0.116148, 0.972111, 0.488703, 1.777672, 1.452170, 0.814134, 1.272649, 1.281416, 0.101871, 0.897888, 0.199760, 0.085732, 1.686579, 0.964558, 0.057281, 0.910252, 1.662302}
3	{ 0.955234, 0.834348, 1.672478, 1.324896, 0.444544, 1.721172, 0.153356, 1.602240, 0.171880, 1.169774, 0.543628, 1.409581, 1.763724, 1.686754, 1.210390, 0.402691, 0.983618, 0.862997, 1.220409, 0.890061}
4	{ 0.031641, 0.461590, 1.719550, 1.357698, 1.112262, 1.166531, 0.246097, 1.387325, 0.177485, 1.446268, 0.799476, 1.667227, 1.723465, 1.505920, 0.245874, 1.155854, 0.831394, 0.677194, 0.568871, 1.652070}
5	{ 0.019803, 1.197794, 0.635553, 0.531682, 0.878194, 0.048050, 0.080480, 1.566743, 0.724210, 0.853668, 1.741191, 0.698465, 1.553550, 0.130290, 0.688346, 1.331091, 0.599759, 1.125466, 1.764818, 1.042879}
6	{ 1.486589, 1.627971, 1.871181, 0.102359, 0.035021, 1.403176, 1.468675, 0.190347, 0.553282, 1.031227, 1.232390, 1.255724, 1.504443, 0.683526, 0.600958, 1.746936, 1.529243, 1.448196, 0.646850, 0.116053}
7	{ 1.283295, 0.355220, 1.380620, 1.858453, 0.818804, 0.219006, 0.476292, 0.420029, 1.291187, 0.568738, 1.174088, 0.628805, 1.753154, 1.459582, 1.354449, 1.755790, 0.441757, 0.856240, 1.647962, 0.686353}
8	{ 1.395289, 0.699934, 0.239310, 0.239535, 0.958190, 0.748780, 0.513784, 1.666344, 1.461995, 1.599060, 0.893107, 0.341873, 1.387703, 1.808363, 0.676542, 1.424958, 0.310574, 0.836247, 1.011101, 1.686200}
9	{ 1.823432, 0.994827, 1.635555, 0.684380, 1.017029, 1.440371, 1.694641, 0.607132, 1.197331, 0.862161, 0.666449, 1.047956, 0.159627, 0.043131, 1.251515, 1.618724, 0.216906, 0.152250, 0.471610, 0.744260}
10	{ 0.576810, 1.632177, 1.556912, 1.866317, 0.568088, 1.541817, 1.726725, 0.275154, 0.814958, 0.863399, 1.333040, 0.148277, 0.197893, 1.048665, 1.158090, 1.692225, 0.884294, 0.289619, 0.380633, 1.728234}
11	{ 1.433213, 1.749505, 1.533837, 0.669701, 0.372580, 1.052390, 1.116645, 0.181320, 1.139126, 0.222671, 0.604393, 1.811797, 1.743315, 1.368792, 1.861434, 0.751908, 0.159811, 1.566503, 0.443273, 1.667530}
12	{ 1.083060, 1.243136, 0.717777, 0.675019, 0.690490, 0.672228, 1.060789, 0.423566, 1.198457, 0.485768, 0.993953, 0.443540, 0.361702, 1.552042, 0.863562, 1.517677, 1.061899, 0.691413, 1.642818, 1.756590}
13	{ 0.278323, 0.790363, 0.172303, 0.417138, 0.009343, 0.783325, 1.369303, 1.041067, 0.467102, 0.992773, 1.525170, 0.871213, 0.243906, 1.542036, 0.449148, 0.843633, 0.191800, 1.614246, 1.038188, 1.415620}

14	{ 0.551081, 0.382599, 1.410121, 0.102084, 0.137286, 0.671081, 0.254860, 1.758068, 1.079013, 0.129143, 1.410873, 0.150485, 0.601119, 0.760737, 0.975905, 0.223261, 0.710162, 1.677048, 0.996836, 1.849865}
15	{ 1.536222, 0.089016, 0.960881, 0.388690, 0.379955, 1.002223, 1.271420, 1.410632, 0.254397, 1.535559, 1.133703, 1.305280, 1.466565, 0.274167, 0.399688, 1.359638, 1.766289, 1.401348, 1.310883, 0.261030}
16	{ 1.314825, 1.538635, 1.317986, 1.243167, 1.749461, 1.689706, 0.024853, 0.634754, 1.036317, 1.828101, 1.676951, 0.023606, 0.857000, 0.076471, 1.622198, 0.254469, 1.451625, 1.720881, 0.763812, 0.186982}
17	{ 0.056994, 0.590507, 0.375291, 1.609261, 0.607721, 0.026355, 0.483366, 0.823931, 0.792878, 0.163577, 0.753588, 0.730789, 0.135991, 1.031660, 1.554135, 1.192863, 0.016693, 0.125796, 1.017920, 1.591773}
18	{ 0.575956, 0.112943, 0.249506, 1.399570, 0.053241, 1.410759, 0.251638, 1.059086, 0.025315, 1.422914, 1.030412, 0.848758, 0.317396, 1.375456, 1.116858, 1.682310, 0.279550, 0.325974, 0.937704, 1.744329}
19	{ 0.447773, 1.024286, 1.001528, 1.863684, 1.278323, 0.860699, 1.346331, 1.692596, 0.022627, 1.033613, 0.546354, 0.395804, 1.486546, 1.381045, 1.312260, 0.245976, 1.607429, 1.818793, 0.964359, 1.496598}
20	{ 0.669967, 1.535929, 1.841878, 0.979127, 0.614002, 1.879218, 0.512531, 1.167061, 0.081697, 1.773427, 1.535668, 0.757729, 0.220395, 1.538243, 1.281162, 0.302159, 0.889871, 0.798522, 1.476288, 1.665941}
21	{ 0.915365, 1.394094, 0.757041, 0.350064, 1.199679, 1.319499, 1.128405, 0.632337, 0.790673, 0.461582, 1.693343, 1.537442, 0.346527, 0.433782, 1.754552, 0.550903, 0.686724, 0.764433, 1.792750, 1.489998}

6.2.6.4 Energy adjustment

The powers of the decorrelator input and output signals are computed:

$$P_{decorr,in,ch}(n) = \left|Y_{decorr,in,ch}(n)\right|^2 \tag{6.2-78}$$

$$P_{decorr,ch}(n) = \left|Y_{decorr,ch}(n)\right|^2 \tag{6.2-79}$$

The powers are smoothed:

$$\bar{P}_{decorr,in,ch}(n)=(\gamma \bar{P}^{[-1]}{}_{decorr,in,ch}(n)+(1-\gamma)P_{decorr,in,ch}(n)) \tag{6.2-80}$$

$$\bar{P}_{decorr,ch}(n)=(\gamma \bar{P}^{[-1]}{}_{decorr,ch}(n)+(1-\gamma)P_{decorr,ch}(n)) \tag{6.2-81}$$

Where: $^{[-1]}$ denotes the value in the previous CLDFB time slot

$\gamma = 0.8$ is the smoothing factor

The adjustment gains $g_{ener}(n)$ is calculated as:

$$g_{ener,ch}(n) = \begin{cases} \sqrt{\frac{1.5\bar{P}_{decorr,in,ch}(n)}{\bar{P}_{decorr,ch}(n)}}, & if\ 1.5\bar{P}_{decorr,in,ch}(n) < \bar{P}_{decorr,ch}(n) \\ \min\left(2, \sqrt{\frac{\bar{P}_{decorr,in,ch}(n)}{1.5\bar{P}_{decorr,ch}(n)}}\right), & if\ 1.5\bar{P}_{decorr,ch}(n) < \bar{P}_{decorr,in,ch}(n) \\ 1, & else \end{cases} \tag{6.2-82}$$

And are applied to the decorrelated signals:

$$Y_{decorr,adj,ch}(n) = g_{ener,ch}Y_{decorr,ch}(n) \tag{6.2-83}$$

6.2.6.5 Final output

The final decorrelator output is constructed as:

$$Y_{decorr,ch}(n) = \begin{cases} Y_{decorr,adj,ch}(n) + Y_{onsets,ch}(n), & n < n_{B,decorr} \wedge f_{onsets} = 1 \\ Y_{decorr,adj,ch}(n), & n < n_{B,decorr} \wedge f_{onsets} = 0 \\ Y_{proto,ch}(n), & n_{B,decorr} \le n \le n_{B,S} \end{cases} \tag{6.2-84}$$

6.2.7 Common tools for decoding with time scale modification (JBM)

6.2.7.1 Overview

The IVAS Jitter Buffer Management (JBM) [5] allows for fine grain adjustment of the play out delay by generating time scale modified (TSM) versions of a multi-channels signal, i.e. provide decoded frames that are longer or short in duration than the default frame length. The IVAS JBM splits the decoding and reconstruction/rendering into the steps transport channel and metadata decoding, the multi-channel time scale modification of the transport channels, resulting in a time scale modified version of the transport channels with specific duration, the adaption of the metadata and other rendering/reconstruction parameters to the time scale modified duration of the IVAS frame, and the reconstruction and rendering adapted to the new time scale modified frame length. The IVAS JBM decoding process performs a number of processing steps to provide the processed (output) signal based on the input audio signal representation (the encoded IVAS frame), where the time scale modification is performed on the intermediate audio signals, i.e. the transport channels which are generated by the first processing step of decoding the transport channels and meta data, and performs the second processing, i.e. the reconstruction/rendering of the output signal based on the time scaled intermediate audio signals. The reconstruction and rendering is adapted to the time scale modification as are the parameters needed for the reconstruction, and the meta data needed for the reconstruction.

The decoder shall keep track of the number of already rendered samples, time slots, and sub frames.

6.2.7.2 Transport channel buffer management

The time scale modified transport channels are stored in the transport channel buffer. Depending on the IVAS format and the method of rendering/reconstruction the output processing has a specific time granularity. For all output processing involving the CLDFB filter bank this time granularity is 1.25 ms, for the TD binaural object rendering and the CREND rendering it is 5 ms. For other rendering methods or cases where the transport channel acts as a simple output buffer the granularity is set to 1.25 ms to ease rate switching to other rendering methods. Due to the nature of the time scale modification the resulting frame length does not necessarily has to be an integer multiple of the reconstruction or output time granularity. The transport channel buffer management take this into account by buffering residual samples that do not fit into a multiple of the time granularity and forms the next block of samples to be processed by starting with the buffered samples and appending the new (time scale modified) samples.

The number of residual samples L_{res} in the current frame is:

$$L_{res} = \mathrm{mod}\left(L_{res}^{[-1]} + L_{TSM}, L_{ts}\right) \quad (6.2\text{-}85)$$

Where: L_{ts} is the length of one rendering/reconstruction slot in samples

$L_{res}^{[-1]}$is the number of residual samples from the previous frame

L_{TSM} is the frame length in number of samples of the time scale modified frame

The number of slots n_{TS} (each of the duration of the specific time granularity) to be rendered in the current frame is therefore:

$$n_{TS} = \frac{L_{res}^{[-1]} + L_{TSM} - L_{res}}{L_{ts}} \quad (6.2\text{-}86)$$

And the number of samples to be rendered L_{rend} in the current frame is:

$$L_{rend} = \frac{L_{res}^{[-1]} + L_{TSM} - L_{res}}{L_{ts}} \quad (6.2\text{-}87)$$

The transport channel buffer may also be used to keep track of the number of already rendered samples, time slots, and subframes.

Figure 6.2-20 depicts this residual buffering for a shortened frame where the shortened TC frame is of length L_{TSM}, the portion marked 1 is the residual portion from the previous frame of length $L_{res}^{[-1]}$, the portion marked 2 is the residual portion of the current frame of length L_{res}, and the last TS has the index $m = n_{TS} - 1$.

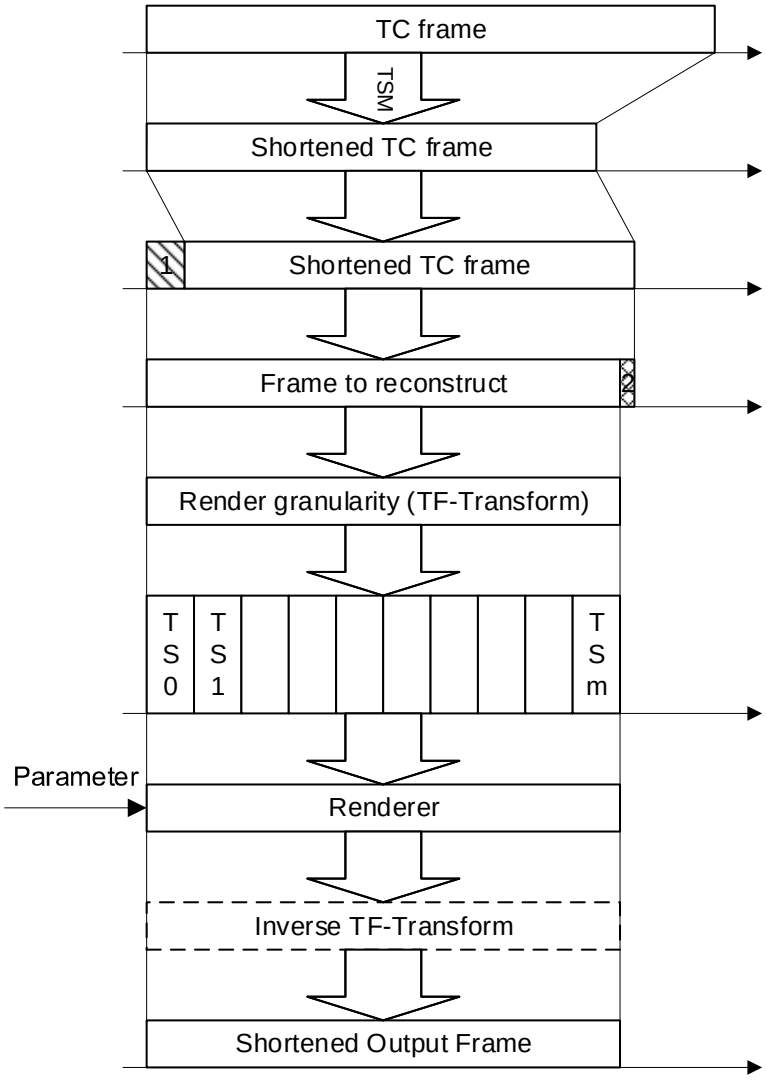


Figure 6.2-20: **TSM transport channel buffer management**

6.2.7.3 Time scale modification

6.2.7.3.1 Extension to multi-channel processing

The channels to be time scale modified are interleaved to form a single interleaved time-domain vector of the channel signals prior to the signal based adaptation. Sub clauses 5.4.3.1, 5.4.3.2, 5.4.3.3, 5.4.3.5, 5.4.3.6, and 5.4.3.7 of [5] are carried out on the interleaved signal. For this all parameters in those clauses that relate to a length in time are multiplied by the number of channels n_{TC} to be modified, the only exception being the Similarity Measurement complexity Parameters in Table 4 of [5].

Prior to the feeding into the TSM transport channel buffer the time scale modified channel signals are deinterleaved.

6.2.7.3.2 Energy estimation

For IVAS the energy estimation (5.4.3.4 of [5]) is replaced by the following procedure.

Low-level signals are detected by analysing 1 ms subsegments h of the interleaved input signal x, including the previous frame. If all subsegment energies E_h of the subsegments to be merged are below a threshold E_{max}, with $E_{max} = -65$ dB , then the frame is scaled as much as possible and in a manner that there are no residual samples left after the scaling, i.e. L_{res}=0:

A 20 ms frame is shortened by setting s as:

$$s = s_{end} + \mathrm{mod}\left(L_{TS,TSM} - \mathrm{mod}\left(s_{end}, L_{TS,TSM}\right), L_{TS,TSM}\right) - \mathrm{mod}\left(L_{TS,TSM} - L_{res,TSM}, L_{TS,TSM}\right) \quad (6.2\text{-}88)$$

$$s = \begin{cases} s + \left\lceil \frac{-s}{L_{ts,TSM}} \right\rceil L_{ts,TSM}, & if\ s < 0 \\ s - \left\lceil \frac{s-(s_{end}-n_{ch,TSM})}{L_{ts,TSM}} \right\rceil L_{ts,TSM}, & if\ s > s_{end} \\ s, else \end{cases} \quad (6.2\text{-}89)$$

where s_{end} is s_{end} for time shrinking according to Table 2 of [5] with the parameters adapted to the actual number of channels to process.

A 20 ms is extended by setting s as:

$$s = s_{start} + \mathrm{mod}\left(L_{TS,TSM} - L_{res,TSM}, L_{TS,TSM}\right) \quad (6.2\text{-}90)$$

where s_{start} is s_{start} for time stretching according to Table 3 of [5] with the parameters adapted to the actual number of channels to process.

The generation of the output frame is as in subclause 5.4.3.7 of [5] with the adaption to the number of channels to process.

Not that for low-level signal the output frame is generated without similarity estimation or quality estimation.

6.2.7.4 Metadata and processing parameter adaption

6.2.7.4.1 Overview

Some of the IVAS modes utilize interpolators to seamlessly interpolate between metadata or parameters than may be derived from the metadata associated to the previous frame and metadata or parameters associated to the current frame to get a linear combination of those parameters for each time instance of the rendering/reconstruction, a time instance being a time domain sample index or a filter bank time slot. These interpolators are typically linear interpolators and are adapted to the length of the frame to be rendered.

Some of the IVAS modes use metadata that is associated to either a sub frame, i.e., a 5 ms portion of the non-modified transport channel signal or to a time slot, e.g., one slot in the CLDFB filter bank, of the non-modified transport channel and use this metadata to reconstruct the output signal. In case of a time scale modified frame, a mapping is determined to map the subframe or slot of the frame to be rendered to the subframe or slot of the non-modified metadata.

6.2.7.4.2 Metadata adaption

6.2.7.4.2.1 Adaption with un-even spacing

For the metadata mapping an index interpolation function $g_{a,map}(n)$ for the current frame is determined:

$$g_{a,map}(n) = \begin{cases} \frac{n+(L_{seg}-L_{rem})}{2L_{seg}-1}, L_{seg} \le n < L_f \\ n\frac{2L_{seg}-L_{rem}}{L_{seg}(2L_{seg}-1)}, 0 \le n < L_{seg} \end{cases} \quad (6.2\text{-}91)$$

Where: L_{seg} is a half length of the frame without TSM in time slots

$2L_{seg}$ the length of the frame without TSM in time slots

L_f is the length of the frame to be rendered in time slots (n_{TS})

$L_{rem} = L_f - L_{seg}$

For un-even spacing of the metadata adaption where the metadata is stored in a ring buffer of length L_{md}, with a read offset into the buffer n_{offset} the vector of the metadata mapping of the current frame $m_{sf}(n)$ is determined:

$$m_{sf}(n) = \mathrm{mod}(n_{offset} + \max\left(0, \frac{\lfloor (2L_{seg}-1) g_{a,map(n)} \rceil}{L_{sf}}\right), L_{md}) \quad (6.2\text{-}92)$$

Where: L_{sf} is the default length of a subframe in time slots.

$\lfloor \ \rceil$ denotes the rounding to the nearest integer

For un-even spacing if the metadata adaption where metadata is stored in a simple buffer with no offset the mapping vector $m_{sf}(n)$ is determined as:

$$m_{sf}(n) = \max\left(0, \frac{\lfloor (2L_{seg}-1) g_{a,map(n)} \rceil}{L_{sf}}\right) \quad (6.2\text{-}93)$$

6.2.7.4.2.2 Adaption with even spacing

Metadata is mapped with even spacing with MASA format, using the methods described in the following.

A slot to subframe map $M_{slot-sf}(n)$ is determined that describes which original subframe m_{orig} is used for each rendering slot n.

If the number of slots N_{slots} in the adapted frame is a factor of the number of the slots $N_{slots,sf} = 4$ for each subframe in the original frame, the mapping length $L_{map}(m_{orig})$ for each original subframe (i.e., for how many rendering slots a certain original subframe is used) is determined by

$$L_{map}(m_{orig}) = \frac{N_{slots}}{N_{slots,sf}}, \quad if\ \mathrm{mod}(N_{slots}, N_{slots,sf}) == 0$$

where m_{orig} is the subframe index of the original frame and $\mathrm{mod}()$ denotes modulo.

If $\mathrm{mod}(N_{slots}, N_{slots,sf}) \neq 0$, some subframes are used for a different number of rendering slots. In that case, the number of rendering slots for each original subframe is determined by

$$L_{map}(m_{orig}) = \mathrm{floor}\left(\frac{N_{slots}}{N_{slots,sf}}\right) + increment(m_{orig})$$

where $\mathrm{floor}()$ denotes rounding down, and the residual increment is determined by

$$increment(m_{orig}) = \mathrm{floor}(decSum(m_{orig}) + \epsilon)$$

where ϵ is a small value to counter rounding errors and the decimal sum is determined by

$$decSum(m_{orig}) = \sum_{i=0}^{m_{orig}} \mathrm{remainder}\left(\frac{N_{slots}}{N_{slots,sf}}\right)$$

where $\mathrm{remainder}()$ denotes taking the remainder of the division. The decimal sum is further modified for each original subframe by

$$decSum(m_{orig} + 1) = decSum(m_{orig}) - 1, \ \ if\ increment(m_{orig}) > 0$$

The slot to subframe map $M_{slot-sf}(n)$ is then determined for each rendering slot n by

$$M_{slot-sf}(n) = m_{orig}, \quad n \in [L_{sum}(m_{orig}), L_{sum}(m_{orig}) + L_{map}(m_{orig}) - 1]$$

where the sum of the lengths of the previous mapped subframes is determined by

$$L_{sum}(m_{orig}) = \sum_{i=0}^{m_{orig}-1} L_{map}(i)$$

The slot to subframe mapping $M_{slot-sf}(n)$ can then be converted to subframe to subframe mapping as described in clause 6.2.7.4.2.1.

6.2.7.4.3 Processing parameter adaption

6.2.7.4.3.1 Subframes

In the IVAS decoding a complete subframe is always defined as a 5-ms portion of the final output audio signal. Due to the nature of the time scale modification a complete subframe can extend over the borders in case of a render granularity of 1.25 ms. For the rendering and reconstruction of the current frame the frame local subframes for processing are adapted depending on the size of the last frame local subframe of the previous frame and the number of time slots to be rendered in the current frame.

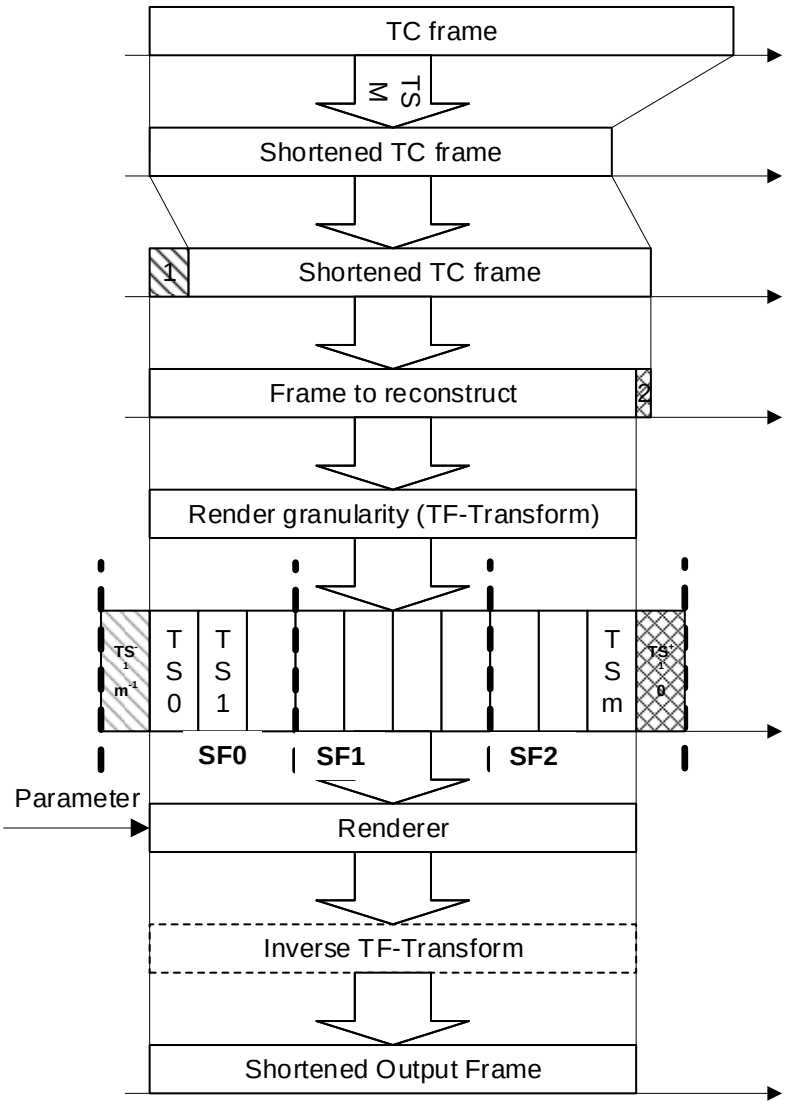


Figure 6.2-21: TSM local subframe adaption

For this the number of time slots $n_{TS,SF}$ in a 5 ms sub frame is determined:

$$n_{TS,SF} = \frac{L_{SF}}{L_{ts}} \tag{6.2-94}$$

The array of number of slots in the a subframe $n_{slots,sf}$ and the number of subframes n_{SF} are determined:

$$n_{slots,0} = \begin{cases} n_{TS,SF}\ , if\ n^{[-1]}_{slots,n^{[-1]}_{SF}-1} = n_{TS,SF} \\ n_{TS,SF} - n^{[-1]}_{slots,n^{[-1]}_{SF}-1},\ otherwise \end{cases} \quad (6.2\text{-}95)$$

$$n_{TS,rem} = n_{TS} - n_{slots,0} \quad (6.2\text{-}96)$$

$$n_{SF} = 1 + \left\lceil \frac{n_{TS,rem}}{n_{TS,FS}} \right\rceil \quad (6.2\text{-}97)$$

$$n_{slots,n_{SF}-1} = \begin{cases} n_{TS,SF}\ , if\ n_{slots,0} = n_{TS,SF} \\ n_{TS,SF} - n_{slots,0},\ otherwise \end{cases} \quad (6.2\text{-}98)$$

$$n_{slots,sf} = n_{TS,FS}, 1 \le sf < n_{SF} - 2 \quad (6.2\text{-}99)$$

Figure 6.2-21 illustrates the local sub frame adaption for a shortened frame with 1.25 ms render time resolution.

6.2.7.4.3.2 Frame interpolators

Certain reconstruction/rendering processes use a full frame interpolator like follows:

$$P_{inter}(n) = g_{a,f}(n)P + \left(1 - g_{a,f}(n)\right)P_{prev} \quad (6.2\text{-}100)$$

Where: n is the index of the current processing unit (time domain sample, CLDFB time slot,…)

P_{inter} is the interpolated processing parameter for the current time index

g_a is the (linear) interpolation function

P, P_{prev} are the frame related processing parameters of the current and the previous frame.

In another way the reconstruction/rendering might be formulated as:

$$\mathbf{y}(n) = (g_{a,f}(n)\mathbf{M} + \left(1 - g_{a,f}(n)\right)\mathbf{M}_{prev})\boldsymbol{x}(n) \quad (6.2\text{-}101)$$

Where: n is the index of the current processing unit (time domain sample, CLDFB time slot,…)

$\boldsymbol{y}(n)$ are the rendered set or is the rendered vector of output samples (time domain, CLDFB domain) of the current time index n

$\boldsymbol{x}(n)$ are the set or is the vector of (time scale modified) intermediate samples (time domain, CLDFB domain) of the current time index n

g_a is the (linear) interpolation function

$\mathbf{M}$, $\mathbf{M}_{prev}$ are the frame related processing parameters of the current and the previous frame.

The interpolation function based on the number of processing units (samples to be rendered, time slots to be rendered) in the current frame to be rendered is expressed as:

$$g_{a,f}(n) = \max(0, \begin{cases} \frac{n+1+(L_{seg}-L_{rem})}{2L_{seg}}, L_{seg} \le n < L_f \\ n+1\frac{2L_{seg}+1-L_{rem}}{2L_{seg}(L_{seg}+1)}, 0 \le n < L_{seg} \end{cases}) \quad (6.2\text{-}102)$$

Where: n is the time index expressed in the interpolator time resolution (time samples, CLDFB time slots,…)

L_{seg} is the portion in which the overlap-and-add is applied by the time scale modification, expressed in the interpolator time resolution

L_f is the length of the frame to be processed, expressed in the interpolator time resolution

$L_{rem} = L_f - L_{seg}$

6.2.7.5 Bitrate switching

6.2.7.5.1 Renderer flushing

When the render granularity on a bitrate switch changes from 5 ms in the previous frame to 1.25 ms in the current frame, to avoid that transport channel samples that can still be rendered in the 1.25 ms are unnecessarily set to zero in the transport channel reconfiguration (see next section), the residual samples of the previous frame that still fit in the 1.25-ms render granularity are padded with zeroes to the length of a 5-ms subframe, rendered with the renderer configuration of the previous frame. The rendered output signal is then shortened to the length that fits into the 1.25 ms render granularity and the number of residual samples and the information about the last subframe are adapted to both the new granularity and the number of 1.25 ms rendered in the flush procedure.

The number of 1.25 ms time slots that can be still rendered is:

$$n_{slots,flush} = \left\lfloor \frac{L_{res}^{[-1]}}{L_{1.25}} \right\rfloor$$

The 5-ms subframe to be flushed (rendered) is the constructed as:

$$\boldsymbol{x}_{rend}(n) = \begin{cases} \boldsymbol{x}_{TC}(n + L_{res}), & 0 \le n < n_{slots,flush} L_{1.25} \\ 0, & n_{slots,flush} L_{1.25} \le n < n_{TS,SF} L_{1.25} \end{cases}$$

The subframe is rendered using the rendering settings of the previous frame to get the rendered signal in the output format y_{rend}.

The valid samples of the flushed are put out:

$$y_{flush}(n) = y_{rend}(n), 0 \le n < n_{slots,flush} L_{1.25}$$

The number of residual samples is corrected:

$$L_{res}^{[-1]} = L_{res}^{[-1]} - n_{slots,flush} L_{1.25}$$

the number of time slots in the last subframe of the previous frame is set to:

$$n_{slots,n_{SF}^{[-1]}-1}^{[-1]} = n_{slots,flush}$$

6.2.7.5.2 Transport channel reconfiguration

If the number of channels buffered in the transport channel changes on a bit rate switch, the transport channel buffers are re-initialized and set to zero. The information of the previous local subframes and the previous number of residual samples is kept.

6.2.7.5.3 Renderer discard samples

If the render granularity changes from 1.25 ms to 5 ms on a bit rate switch, the transport channel buffers are re-initialized and set to zero. To ensure the correct continuation of the global 5 ms subframes, zero samples are padded at the front of the frame to be rendered that are removed after the rendering and the number of time slots in the last sub frame of the previous frame is corrected.

The number of samples to be padded is determined as:

$$L_{discard} = n_{slots,n_{SF}^{[-1]}-1}^{[-1]} L_{1.25}$$

The number of time slots in the last subframe of the previous frame is set to:

$$n_{slots,n_{SF}^{[-1]}-1}^{[-1]} = 1$$

6.3 Stereo audio decoding

6.3.1 Common stereo decoding tools

6.3.1.1 Common Stereo postprocessing

The stereo decoder (CPE tool, in general) contains two postprocessing modules.

The first one takes care of synchronizing upmixed DFT/TD/MDCT stereo output synthesis to match the overall core-decoder delay of 3.25 ms. In case of the DFT stereo, the upmixed DFT stereo left and right channel synthesis are delayed by 0.125 ms (i.e. a difference between core-decoder delay and right overlap part of DFT analysis/synthesis window from Figure 6.3-5). In case of the TD stereo and the MDCT stereo, the upmixed left and right channel synthesis are delayed by 2.0 ms (i.e. a difference between core-decoder delay and core-decoder CLDFB resampling delay of 1.25 ms). Details about differences between stereo modes synthesis handling can be found in clause 6.3.4.2.

The other module is responsible for handling stereo mode switching (when it happens) by means of smoothing past and current downmix and output synthesis signals, extrapolations, and recomputation of certain signals. The process of switching between stereo modes is described in detail in clause 6.3.4.

6.3.1.2 ITD compensation

6.3.1.2.1 Overview

For all stereo modes where the input signals were time-aligned at the encoder before processing either in time-domain, as described in clauses 5.3.2.3.1 for TD-stereo and 5.3.2.4.2 for DFT stereo and MDCT stereo, or frequency-domain, as described in clause 5.3.2.4.5 for DFT-stereo , at the end of the decoder processing a temporal shifting of the lagging channel occurs to reintroduce the initial ITD of the stereo signals. The temporal shifting implementation is done using the Inter-channel Alignment (ICA) described in clause 6.3.1.2.2

6.3.1.2.2 Inter-channel Alignment (ICA) decoder

The ICA decoder operates as an inter-channel re-aligner. After receiving the ICA bitstream, dequantization of the inter-channel temporal shift and gain factor parameters takes place, i.e., prevNCShift, currentNCShift, and the targetGain are generated. Similar to what occurred at the ICA encoder, there is a channel identification and target adjustment stage. The reference and target channels are identified. If the temporal shifts between the current and previous frames differ, the target signal is adjusted by the same structure as the Target Sample Adjuster as described in the encoder. However, the shift of the target signal is in the opposite direction as was performed in the ICA Encoder, i.e., the Inter-Channel Re-Aligner places back the left and right channels to their original temporal difference. The adjusted target channel is scaled by a targetGain to balance the audio levels between the two channels.

Decoder obtains the primary-channel signal and the secondary-channel signal, and the ITD from bitstream. Time-domain upmixing processing on the primary-channel signal and the secondary-channel signal is performed to obtain the left-channel reconstructed signal and the right-channel reconstructed signal. Delay of the left-channel reconstructed signal and the right-channel reconstructed signal is adjusted based on the ITD after an interpolation processing which is performed based on the ITD from current frame and previous frame. The ITD after the interpolation processing is calculated according to a formula

$$A = \alpha \bullet B + (1 - \alpha) \bullet C$$

wherein A is the inter-channel time difference after the interpolation processing in the current frame, B is the inter-channel time difference in the current frame, C is the inter-channel time difference in the previous frame of the current frame, α is a pre-stored interpolation coefficient which can be set to 0.4.

6.3.2 Unified stereo decoding

6.3.2.1 Inter-channel BWE (IC-BWE) decoder

6.3.2.1.1 Inter-Channel BWE Decoder overview

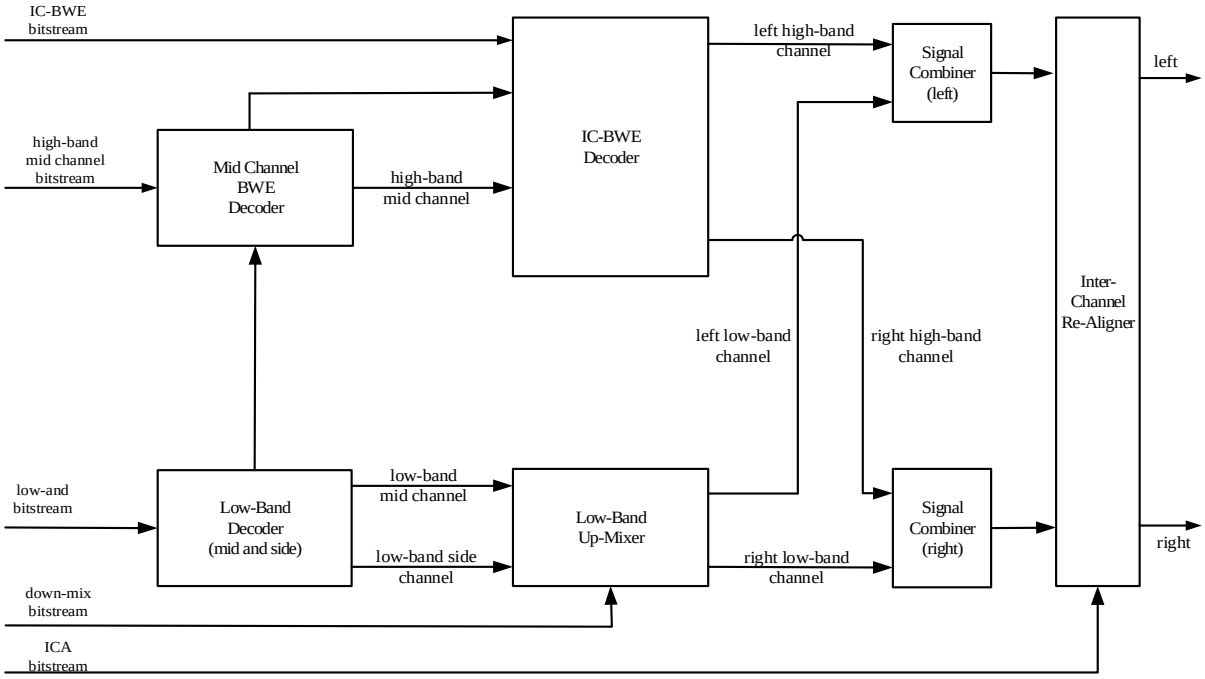


Figure 6.3-1: Block diagram of IC-BWE Decoder in relation to other blocks

As seen in the IC-BWE Decoder System View of Figure 6.3-1, several bitstreams are received by the decoder, the IC-BWE bitstream, the high-band mid-channel bitstream, the low-band bitstream, and the ICA bitstream. The IC-BWE bitstream includes inter-channel bandwidth parameters: adjustment gain parameters (gsIndx), an adjustment spectral shape parameter (spIndx), and refChanIndx_bwe). The Mid-Channel BWE decoder dequantizes the high-band mid-channel bitstream and generates generate a mid-channel time-domain high-band signal by performing bandwidth extension based on the mid-channel bitstream. The IC-BWE Decoder receives the decoded high-band mid-channel signal and also dequantizes the inter-channel-bandwidth parameters and generates a left high-band channel signal and right high-band channel signal. The left channel is generated by combining both left high-band and low-band channel signals. Similarly, the right channel is generated by combining both right high-band and low-band channel signals. Once the target channel is identified, the target channel is generated by modifying the left or right channel signal based on the temporal mismatch value between them. The IC Re-Aligner generates the modified target channel signal by temporally shifting first samples of the target channel signal relative to second samples of the reference channel signal by an amount based on the temporal mismatch value. However, the temporal shifting that takes place is in the opposite direction from the temporal shifting that took place in the ICA Encoder.

6.3.2.1.2 Inter-Channel BWE Decoder Block

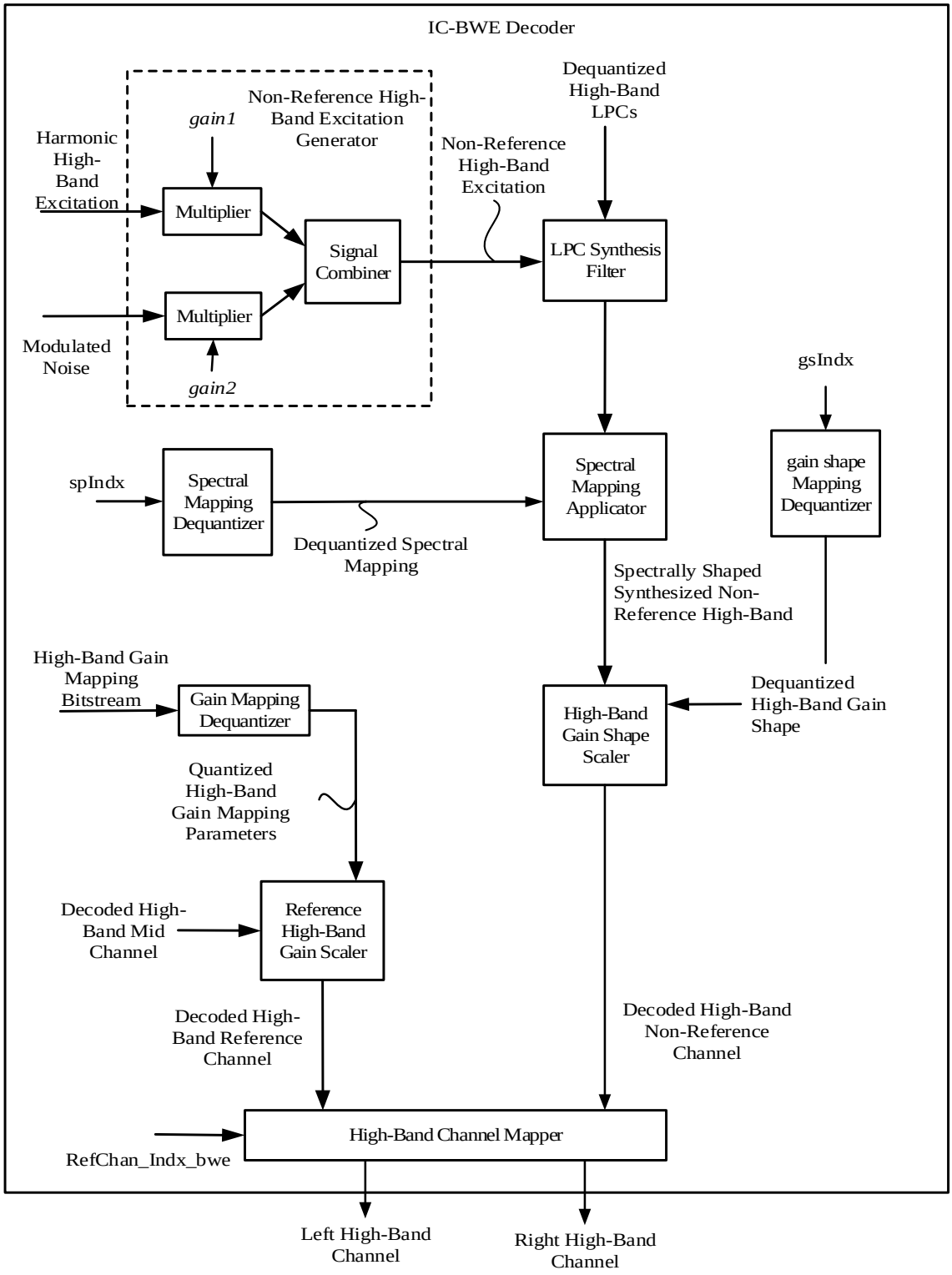


Figure 6.3-2: Block diagram of the IC-BWE Decoder

The IC-BWE decoder includes (a) Non-reference High-Band Excitation Generator; (b) a LPC Synthesis Filter; (c) Spectral Mapping Applicator; (d) Spectral Mapping De-quantizer; (e) High-Band Gain Shape Scaler; (f) Non-Reference High-Band Gain Scaler; (g) Gain Mapping Dequantizer; (h) Reference High-Band Gain Scaler; (i) and a High-Band Channel Mapper.

The Non-Reference High-Band Excitation Generator operates the same way as described in the IC-BWE Encoder.

A harmonic high-band excitation generated from the (decoded) low-band bitstream is provided to the multiplier and scaled by *gain1*. The modulated noise is provided to the other multiplier and scaled by gain2. Both gain1 and gain2 are received from the Encoder. Thus, the non-reference high-band excitation is generated in as generated in the IC-BWE encoder. The non-reference high-band excitation is provided to the LPC Synthesis Filter. The LPC Synthesis Filter generates a synthesized non-reference high-band signal based on the non-reference high-band excitation and the Dequantized high-band LPCs, that were received in the high band mid channel bitstream transmitted from the encode. The LPC Synthesis Filter applies the Dequantized high-band LPCs to the non-reference high-band excitation to generate the synthesized non-reference high-band. The synthesized non-reference high-band is provided to the Spectral Mapping Applicator. After the Spectral Mapping Dequantizer dequantizes the spectral mapping parameter(s), spIndx, the decoded specMapping (Dequantized Spectral Mapping) is provided to the Spectral Mapping Applicator. The Spectral Mapping Applicator operates in the same way as described in the IC-BWE Encoder, and uses the same filter as expressed below:

$$H(z) = \frac{1}{1-u*z^{-1}} \quad (6.3\text{-}1)$$

where *u* represents the Quantized Spectral Mapping parameters. After filtering, the spectrally shaped synthesized non-reference high-band is provided to the high-band gain shape scaler. The High-Band Gain Shape Scaler scales the spectrally shaped synthesized non-reference high-band based on the quantized high-band gain shape parameter, gsIndx. The output of the High-Band Gain Shape Scaler is the decoded high-band non-reference channel signal. The decoded high-band non-reference channel signal is provided to the high-band channel mapper. The high-band gain mapping bitstream from the encoder is provided to the Gain Mapping Dequantizer. The Gain Mapping Dequantizer dequantizes high-band gain mapping bitstream and generates the quantized high-band gain mapping parameters. The quantized high-band gain mapping parameters are provided to the Reference High-Band Gain Scaler, and a decoded high-band mid channel generated from the high-band mid channel bitstream is provided to the Reference High-Band Gain Scale. The Reference High-Band Gain Scaler scales the decoded high-band mid channel signal based on the quantized high-band gain mapping parameters and generates a decoded high-band reference channel signal. The decoded high-band reference channel is provided to the High-Band Channel Mapper. The High-Band Channel Mapper uses the high-band reference channel indicator (RefChan_Indx_bwe) to determine which of the decoded high band reference signal or decoded high band non-reference channel signals to designate as left or right.

6.3.2.2 TD-based stereo decoding

6.3.2.2.1 TD stereo decoder overview

The TD stereo decoder receives the bitstream from which the parameters corresponding to the primary channel and the secondary channel are extracted along with the index corresponding to the mixing ratio β. Note that, for simplicity, the decoded parameters of the TD stereo decoder are denoted in the same way as in the TD stereo encoder described in clause 5.3.2.3. The schematic diagram in Figure 6.3-3 shows the principal blocks of the TD stereo decoder.

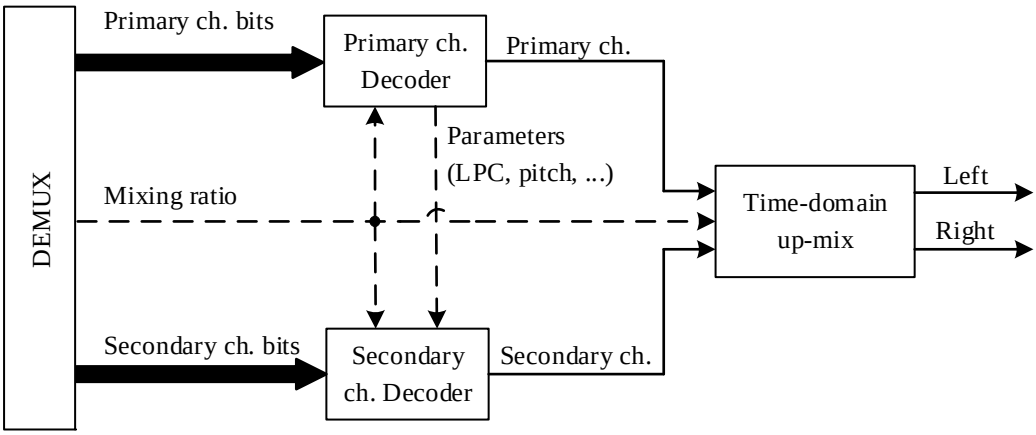


Figure 6.3-3: TD stereo decoder

The decoded mixing ratio β is the key parameter in the TD stereo decoder. The decoded mixing ratio β is used as an indicator of both the primary channel and the secondary channel and it is also used to determine the bitrate allocation among the channels. Therefore, the decoded mixing ratio β is used by the decoder of the primary channel, the decoder of the secondary channel and also by the TD stereo upmix.

In case of LRTD submode the mixing ratio β conveys either the value of 1or the value of 0 indicating the mapping of the primary/secondary channel to the left/right channel of the TD stereo upmix. In that case the instantaneous mixing ratio β_{inst} is decoded to determine the bitrate allocation, as described in clause 5.3.2.3.8, between the primary channel and the secondary channel.

The decoded parameters of the primary channel comprise the parameters of the ACELP coder at the given bitrate. That is the coder type, LP filter coefficients, pitch information, fixed codebook indices and gains. The decoding of the ACELP parameters is described in detail in clause 6.2.2.2. Let the synthesized signal of the primary channel decoder be denoted as $Y'(n)$. Depending on the mixing ratio, or the instantaneous mixing ratio in case of LRTD submode, the bit allocation between the primary and the secondary channel and the information whether the LP filter and the pitch are re-used from the primary channel, the decoder of the secondary channel decides about the ACELP decoder. The decoded coder type in the secondary channel is used to determine whether UC/IC decoder will be used or the GC decoder will be used to decode and synthesize the signal in the secondary channel. The mixing ratio β or, in case of LRTD submode, the instantaneous mixing ratio β_{inst} is used to determine the bitrate of the primary and the secondary channel. The allocated bitrate ultimately leads to the decision whether a 4-subframe or a 2-subframe scheme will be applied at the GC decoder. The schematic diagram in Figure 6.3-4 shows the logic inside the TD stereo decoder of the secondary channel.

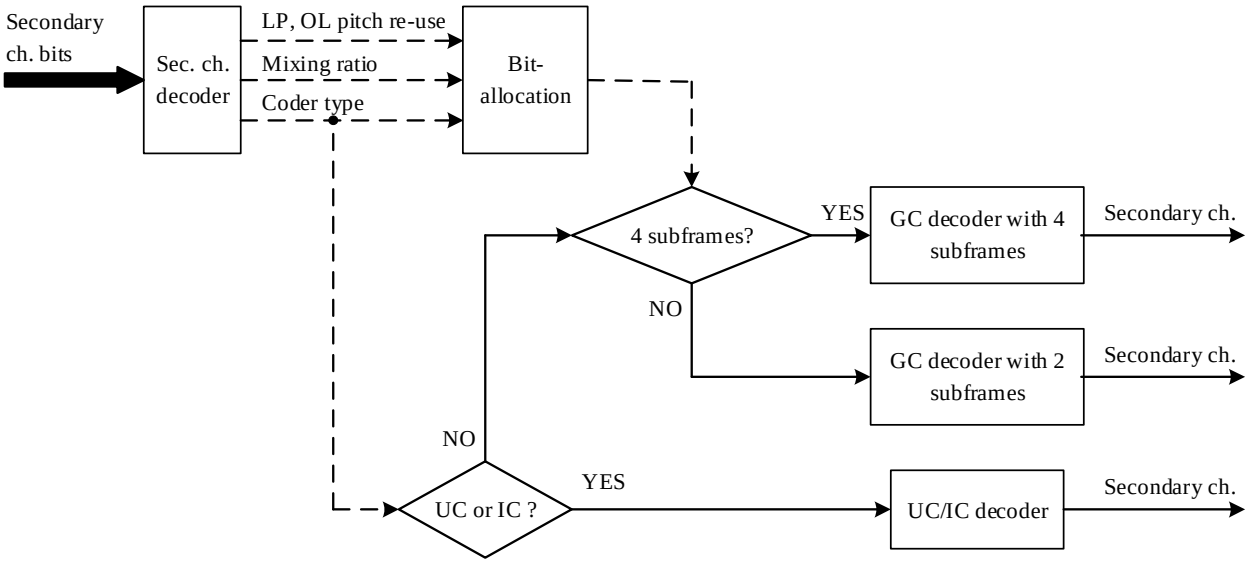


Figure 6.3-4: Subframe selection in the secondary channel of the TD stereo decoder

Both, the GC decoder and the UC/IC decoder in the secondary channel may re-use the LP filter coefficients while only the GC decoder may reuse the pitch information from the primary channel. This is described in detail the following clause. The decoding of the ACELP parameters in the secondary channel and the synthesis of the output signal is described in detail in clause 6.2.2.2.

The time-domain stereo up-mixing operation is the inverse of the time-domain down-mixing operation described in clause 5.3.2.3.3. The up-mixing operation applies the mixing ratio β on the primary channel and the secondary channels to produce the left channel and the right channel of the synthesized stereo output signal. In case of LRTD submode, the mixing ratio β has the value of 1 or 0 indicating the mapping between the primary/secondary channel to the left/right output channel. The time-domain up-mixing operation is defined with the following expressions

$$\begin{aligned} L'(n) &= \frac{\beta \cdot Y'(n) - \beta \cdot X'(n) + X'(n)}{2 \cdot \beta^2 - 2 \cdot \beta + 1} \qquad \text{for } n = 0, \ldots, N-1 \\ R'(n) &= \frac{-\beta \cdot Y'(n) - \beta \cdot X'(n) + Y'(n)}{2 \cdot \beta^2 - 2 \cdot \beta + 1}, \qquad \text{for } n = 0, \ldots, N-1 \end{aligned} \tag{6.3-2}$$

where $L'(n)$ and $R'(n)$ denote the left channel and the right channel of the stereo output, respectively, and $Y'(n)$ and $X'(n)$ the primary and the secondary channel, respectively.

6.3.2.2.2 Reusing close-loop pitch and adaptive cobebook of the primary channel

The decoding of the adaptive codebook vector in the primary channel and the secondary channel of the TD stereo decoder can be described similarly as in clause 6.1.1.2.1.1 of [3]. If the secondary channel reuses the close-loop pitch information from the primary channel, the adaptive codebook parameters are replaced by the closed-loop pitch

information from the primary channel. Let the close-loop pitch information from the primary channel be denoted as T_{CL_Y} and the pitch gain of the secondary channel as $\hat{g}_{p\_X}$. Both, the close-loop pitch of the primary channel and the pitch gain of the secondary channel are transmitted and decoded for each subframe of the current frame. The close-loop pitch of the primary channel and the pitch gain of the secondary channel are used to calculate the adaptive codevector, denoted as $v_X(n)$, according to clause 6.1.1.2.1.1 of [3].

In case of the GC decoder with two subframes (see Figure 6.3-4) the close-loop pitch of the first subframe in the secondary channel, $T_{CL\_X}^{[0]}$, is calculated as the average value of the close-loop pitch of the first and the second subframe of the primary channel. Similarly, the close-loop pitch of the second subframe in the secondary channel, $T_{CL\_X}^{[1]}$, is calculated as the average value of the close-loop pitch of the third and the fourth subframe of the primary channel

That is

$$T_{CL\_X}^{[0]} = 0.5\left(T_{CL\_Y}^{[0]} + T_{CL\_Y}^{[1]}\right)$$
$$T_{CL\_X}^{[1]} = 0.5\left(T_{CL\_Y}^{[2]} + T_{CL\_Y}^{[3]}\right) \quad (6.3\text{-}3)$$

6.3.2.2.3 Reusing LP filter coefficients of the primary channel and dequantization of LSF parameters for the secondary channel

When the coherence between the coefficients of the LP filter of the primary channel and the coefficients of the LP filter of the secondary channel is high enough, as described in clause 5.3.3.3.4.3, the LP filter coefficients of the primary channel may be reused inside the decoder of the secondary channel. In such case one of the following scenarios occurs:

- the quantized LSP vector of the primary channel from the previous frame is used to calculate the LP filter coefficients of the secondary channel in the current frame
- the quantized LSP vector of the primary channel from the previous frame is resampled from the internal sampling frequency of 16 kHz down to 12.8 kHz and then used to calculate the LP filter coefficients of the secondary channel in the current frame
- in case the coder type of the secondary channel is INACTIVE the LP filter coefficients of the primary channel in the current frame are re-used in the secondary channel
- same as in the previous case and when the coherence between the LP coefficients of the primary and the secondary channel is low, then a pre-defined default (average) LSP vector is used to calculate the LP filter coefficients of the secondary channel in the current frame

In scenarios involving LSP vectors the conversion of LSP coefficients to LP filter coefficients is done using the procedure described in clause 5.1.9.7 of [3]. The interpolation of LSP coefficients is performed as described in clause 5.3.3.3.4.5 of [3].

In TD stereo decoding the quantized LSF parameters of the primary channel signal in the current frame are obtained from the bitstream. The IVAS decoder obtaining the quantized LSF parameters includes performing spectrum broadening on the quantized LSF parameters of the primary channel signal; obtaining a prediction residual of LSF parameters of a secondary channel signal from the bitstream; determining a quantized LSF parameter of the secondary channel signal based on the prediction residual of the LSF parameter of the secondary channel signal and the spectrum-broadened LSF parameter of the primary channel signal. The spectrum-broadened LSF parameters of the primary channel signal are obtained by performing pull-to-average processing on the quantized LSF parameters of the primary channel signal, wherein the pull-to-average processing is performed according to the following formula:

$$LSF_{SB}(i) = \beta \cdot LSF_P(i) + (1-\beta) \cdot \overline{LSF_S}(i)$$

wherein LSF_{SB} represents a vector of the spectrum-broadened LSF parameters of the primary channel signal, $LSF_P(i)$ represents a vector of the quantized LSF parameters of the primary channel signal, i represents a vector index, β represents a broadening factor, $0 < \beta < 1$, for example $\beta = 0.91$, $\overline{LSF_S}$ represents a mean vector of an original LSF parameter of the secondary channel signal, $1 \leq i \leq M$, i is an integer, and M represents a linear prediction parameter.

Decoding the spread LSF parameters of the primary channel signal comprises obtaining a quantized LSF parameter of a primary channel signal, obtaining a target adaptive spreading factor of a stereo signal, spreading the quantized LSF parameter of the primary channel signal based on the target adaptive spreading factor. The spread LSF parameter of the

primary channel signal is a quantized LSF parameter of a secondary channel signal in the current frame, or it is used to determine a quantized LSF parameter of a secondary channel signal in the current frame.

6.3.2.2.4 Estimation of spatial parameters of the background noise

In the TD stereo mode the spatial properties of the background noise are estimated during inactive segments signalized with the VAD flag. The key parameter is the inter-channel correlation (IC) which is calculated in the time domain. The inter-channel correlation parameter is calculated as

$$IC_{LR}^{[m]} = \frac{\left|\sum_{n=0}^{N-1} l(n) \cdot r(n)\right|}{\sqrt{\sum_{n=0}^{N-1} l^2(n) \cdot \sum_{n=0}^{N-1} r^2(n)}} \tag{6.3-4}$$

where $l(n)$ and $r(n)$ are the left channel and the right channel of the decoded stereo signal, respectively.

The second key parameter describing the spatial properties of the background noise is the inter-channel level difference (ILD). To calculate the ILD let's express the energy ratio between the left channel and the right channel of the decoded stereo signal as

$$c_{LR} = \frac{\sum_{n=0}^{N-1} l^2(n)}{\sum_{n=0}^{N-1} r^2(n)} \tag{6.3-5}$$

The ILD is then calculated as

$$ILD_{LR}^{[m]} = \frac{c_{LR}-1}{c_{LR}+1} \tag{6.3-6}$$

As both spatial parameters are calculated from a single frame their fluctuation is high. Therefore, the spatial parameters are smoothed by means of a simple IIR filtering. The smoothed inter-channel correlation is calculated as

$$\widetilde{IC}_{LR}^{[m]} = 0.95 \cdot \widetilde{IC}_{LR}^{[m-1]} + 0.05 \cdot IC_{LR}^{[m]} \tag{6.3-7}$$

and the smoothed inter-channel level difference as

$$\widetilde{ILD}_{LR}^{[m]} = 0.9 \cdot \widetilde{ILD}_{LR}^{[m-1]} + 0.1 \cdot ILD_{LR}^{[m]} \tag{6.3-8}$$

During the initialization period, when $f_{CNI} = 0$, the smoothed parameters are set to their instantaneous values. That is

$$\begin{aligned} \widetilde{IC}_{LR}^{[m]} &= IC_{LR}^{[m]}, \quad \text{if } f_{CNI} = 0 \\ \widetilde{ILD}_{LR}^{[m]} &= ILD_{LR}^{[m]}, \quad \text{if } f_{CNI} = 0 \end{aligned} \tag{6.3-9}$$

The initial values for $\widetilde{IC}_{LR}^{[0]}$ and $\widetilde{ILD}_{LR}^{[0]}$ are 0.

The generation of the stereo comfort noise is described in detail in clause 6.3.2.3.8.3. The generation of the stereo comfort noise requires the calculation of the mixing factor γ. In the TD stereo mode the mixing factor γ is calculated in time domain using the estimated spatial parameters. That is

$$\gamma = \sqrt{\frac{\widetilde{IC}_{LR}^{[m]}}{1-\widetilde{IC}_{LR}^{[m]}} + 1 - \left[\widetilde{ILD}_{LR}^{[m]}\right]^2} - \sqrt{\frac{\widetilde{IC}_{LR}^{[m]}}{1-\widetilde{IC}_{LR}^{[m]}}} \tag{6.3-10}$$

6.3.2.3 DFT-based stereo decoding

6.3.2.3.1 General

The DFT stereo decoding is the duality of the DFT-based stereo encoding described in clause 5.3.2.4. The mono downmix is coded and decoded by one the core-coders, before performing the upmix in frequency domain. Some stereo processing steps are still performed in time domain like, like the ITD handling, and the decorrelator.

6.3.2.3.2 STFT analysis

A time-frequency transform like at the encoder side is applied to the decoded downmix $\hat{m}$ yielding time-frequency vectors:

$$\widehat{M_i}[\mathrm{k}] = \sum_{n=0}^{\mathrm{N}-1} \widehat{m}[n + i.M - L1 - zp] w_{dec}[n\%2, n] e^{-j\frac{2\pi kn}{N}} \quad (6.3\text{-}11)$$

where $w_{dec}[n\%2, n]$ are the analysis window functions at the decoder side defined for odd and even i indexes, i being the index of the M size stride (stride parameter), zp corresponds to the zero padding of the analysis windows, $L1$ the outer overlapping size, k is the frequency index, N the DFT size, and j is the imaginary unit. Unlike the encoder side, the stride is of only 10 ms, and two DFTs are performed pr frame of 20 ms. The overlapping in the outer bounds of the two DFTs analyses are of 3.125 ms, while the inner overlapping is of 9.375 ms. Zero padding is added at both ends equally to end up with windows of a total duration of 20 ms as depicted in Figure 6.3-5.

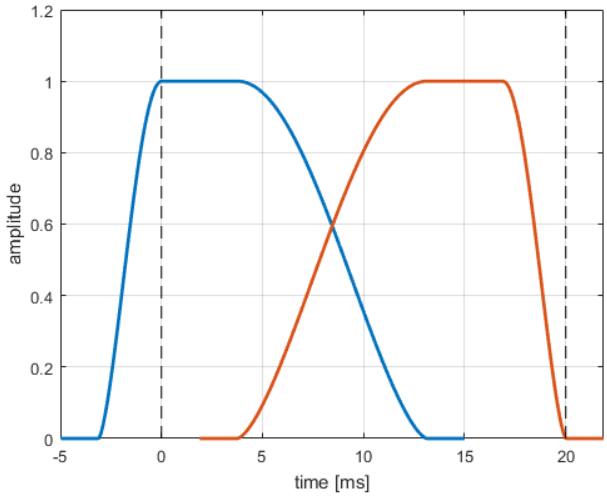


Figure 6.3-5: Analysis and synthesis windows of DFT stereo at decoder side

Same windows are used for the STFT synthesis, since overlapping windows are defined as sine windows.

$$w_{dec}[0, n] = \begin{cases} sin\left(\frac{\pi.(n - zp + 0.5)}{2.L1}\right) for\ zp \le n < zp + L1 \\ 1, for\ zp + L1 \le n < M - L2/2 \\ sin\left(\frac{\pi.(n - zp + 0.5)}{2.L2}\right) for\ M - L2/2 \le n < M + L2/2 \end{cases} \quad (6.3\text{-}12)$$

and zero otherwise. The second window corresponds to inverse of the first one,

$$w_{dec}[1, n] = w_{dec}[0, N - 1 - n] \quad (6.3\text{-}13)$$

where $L1$ is the size in samples of the outer overlapping of 3.125 ms, $L2$ is the size in samples of the inner overlapping of 9.375 ms.

6.3.2.3.3 Stereo parameter decoding

The global IPD index is decoded on 4 bits for the active frames and on 2 bits for the SID frame. The global IPD is then recovered by:

$$\widehat{gIPD} = gIPD_{index}.deltaIPD - \pi \quad (6.3\text{-}14)$$

where $deltaIPD = 2\pi / gIPD_{index\_max}$, where $gIPD_{index\_max} = 2^{gIPD_{bits}}$.

The side and residual prediction gains are coded jointly. First the absolute ILD level is decoded along with sign of the side gain: The ILD level is coded on 4-bit value between 0 and 15, and is used to select the codebook used for coding jointly the absolute value of the side gain and the residual prediction gain:

$$\begin{cases} \widehat{|g|} = gains_{cdbk}[8 * ILD_{level} + gains_{index}][0] \\ \hat{r} = gains_{cdbk}[8 * ILD_{level} + gains_{index}][1] \end{cases} \quad (6.3\text{-}15)$$

where $gains_{index}$ is decoded using the absolute and predictive coding scheme and $gains_{cdbk}[]$ is the table given in Table 5.3-8, all described in clause 5.3.2.4.9.

The side decoded gain is then given by

$$\hat{g} = sign(g).|\hat{g}| \quad (6.3\text{-}16)$$

The decoded parameters are stored in memory to be used for interpolation and to be used in case of packet loss. The interpolation is done per 10 ms subframe in the DFT stereo decoder to align with the core decoding delay.

6.3.2.3.4 Residual signal decoding

If the residual coding is employed, i.e. for bitrate at and higher than 32 kbps, the residual signal is decoded for the low frequencies.

First the global gain ggi is read on 7 bits, If the global gain index is 127, all the residual signal vector is set to zero. Otherwise, the global gain is first decoded as:

$$gg = dequantize\_gain(gg) = 10^{\frac{90}{20 \cdot 127} ggi}, \text{ for } ggi \in \{0, \ldots, 126\} \quad (6.3\text{-}17)$$

The residual vector is split into 8-dimension blocks, and for each of them a coding parameter is read by entropy decoding using a probability model associated to the bitrate.

Entropy decoding for param[k]=0 of a block starts by decoding *nz_count*, the number of nonzero values of ±1, with raw coding using 2 bits. Then, the nonzero mask is decoded, which contains *nz_count* ones and *blk_length*-*nz_count* zeros, with raw coding of the sign bits of the nonzero values.

```
decode_low_entropy_block(block, blk_length)
{
  nz_count = rc_uni_dec_read_bits(2)
  left_1 = nz_count
  left_0 = blk_length – nz_count

  for (i = 0; i < blk_length; i++)
  {
      if (left_1 == 0)
      {
          sym = 0
      }
      else if (left_0 == 0)
      {
          sym = 1
      }
      else
      {
          count0 = left_0 * ECSQ_tab_inverse[left_0 + left_1]
          sym = rc_uni_dec_read_bit_prob_fast(count0, 14)
      }

      if (sym != 0)
      {
          sym *= 1 - 2 * rc_uni_dec_read_bit() /* apply the sign */
          left_1--
      }
      else
      {
          left_0--
      }

      block[i] = sym
  }
}
```

Entropy decoding for $param[k] \geq 1$ of a block starts by computing $shift = \max(0, param[k] - 3)$, which represents the number of least significant bits of the absolute values that are coded approximately uniformly or with raw coding. The most significant bits of each absolute value are decoded using a probability model selected by $param[k]$ together with escape coding. For $shift \leq 4$, decoding of LSBs takes into account that for absolute values the probability of zero (0 maps to one value) is half of the probability of nonzeros (1 maps to two values, ± 1). For larger shifts, the length difference is negligible and raw coding is used using $shift$ bits. For nonzero values, the sign bits are decoded using raw coding.

```
decode_normal_block(block, blk_length, param)
{
  shift = max(0, param - 3)
```

```
  for (i = 0; i < blk_length; i++)
  {
    sym = arith_decode_prob_escape(ECSQ_tab_vals[param - 1], 16)

    if (shift != 0)
    {
      if ((sym > 0) || (shift > 4))
      {
        lsbs = rc_uni_dec_read_bits(shift)
      }
      else /* (sym == 0) && (shift <= 4) */
      {
        lsbs = rc_uni_dec_read_symbol_fast(ECSQ_tab_abs_lsbs[shift], 1 << shift, 14)
      }
      sym = (sym << shift) | lsbs
    }

    if (sym != 0)
    {
      sym *= 1 - 2 * rc_uni_dec_read_bit() /* apply the sign */
    }

    block[i] = sym
  }
}
```

The decoding of the entire quantized q_input vector can be expressed in terms of the previous two functions, which decode low entropy blocks and normal blocks, after decoding their $param[k]$.

```
decode_raw_vector(q_input, N)
{
  block_cnt = (N + 7) / 8

  for (k = 0; k < block_cnt; k++)
  {
    blk_length[k] = min(8, N – 8 * k)
    param[k] = rc_uni_dec_read_symbol_fast(ECSQ_tab_param, 16, 14)

    if (param[k] == 0)
    {
      decode_low_entropy_block(block[k], blk_length[k])
    }
    else
    {
      decode_normal_block(block[k], blk_length[k], param[k])
    }

    for (i = 0; i < blk_length[k]; i++)
    {
      q_input[8 * k + i] = block[k][i]
    }
  }
}
```

The decoded residual signal q_input is then scaled by the decoded global gain gg. The resulting decoded residual signal is then transformed back to the time domain using an inverse MDCT of size 20 ms with an overlapping of 8.75 ms using sine windowing. The time domain decoded residual signal is transformed to the DFT domain using the same STFT analysis used for the decoder downmix. The resulting signal is then $\widehat{Res}_i[k]$ and used in the subsequent stereo upmixing. In case the residual coding is disabled, the spectrum $\widehat{Res}_i[k]$ is set to zero.

6.3.2.3.5 Bass post filter application

6.3.2.3.5.1 Adding the BPF signal in the DFT domain

In DFT-based stereo, the BPF signal is generated the same as in EVS, clause 6.1.4.2 [3]. The subtraction of the BPF signal, however, is not performed in the CLDFB domain, but in the DFT domain. The BPF signal is transformed to the frequency domain the same way as the downmix signal, see clause 6.3.2.3.2, by applying equation (6.3-11) to calculate the spectrum M_{BPF}. This spectrum is then subtracted from the downmix spectrum after applying a weighting:

$$\widehat{M}_i[\mathrm{k}] \coloneqq \widehat{M}_i[\mathrm{k}] - M_{BPF}[k] \cdot w_{BPF}[k], k \in [0, 39].$$

The values of wBPF are given in Table 6.3-1.

Table 6.3-1: DFT-domain Weights for the BPF signal

k	0	1	2	3	4	5	6	7
$w_{BPF}[k]$	1.00004	0.994684	0.978763	0.952708	0.917218	0.873235	0.821906	0.76454
k	8	9	10	11	12	13	14	15
$w_{BPF}[k]$	0.702561	0.637451	0.570704	0.503772	0.438018	0.374674	0.314811	0.259307
k	16	17	18	19	20	21	22	23
$w_{BPF}[k]$	0.208838	0.163865	0.124639	0.091211	0.063451	0.041069	0.023648	0.010670
k	24	25	26	27	28	29	30	31
$w_{BPF}[k]$	0.001553	0.004319	0.007570	0.008799	0.008564	0.007361	0.005613	0.003664
k	32	33	34	35	36	37	38	39
$w_{BPF}[k]$	0.001777	0.000138	0.001142	0.002011	0.002472	0.002565	0.002356	0.001929

6.3.2.3.5.2 Low-cost adaptation of bass post filter for the residual signal

The bass post-filter introduced for EVS in 6.1.4.2 [3] adapts the post-filter strength $\alpha \in [0, 0.5]$ to how well the post-filtered signal correlates with the input signal. A low correlation suggests the filter may have a degrading impact, and as a result the filter output is attenuated. The post-filter strength is also adapted to the LP filter stability, where a low stability leads to an attenuated filter. In EVS operations the post filter strength α and the fundamental pitch period in samples T are both updated each subframe. To prevent the abrupt changes at the subframe boundaries, an adaptation mechanism is introduced for IVAS operations that is used when applying the BPF on the residual signal.

The adaptation mechanism includes a post-filter adaptation module to decide whether to apply the post filter or not based on the energy discontinuities in the signal between consecutive subframes. The input to the adaptation module consists of the reconstructed frequency domain signal $\hat{S}$ generated by the decoder and the post-filter difference signal s_{diff} which describes the difference between the filtered and non-filtered signal in time domain.

Adaptation module first measures the energy of the spectrum $E_{\hat{S}_{cb}}$ in the critical frequency band $[k_{start}, k_{end}]$ for each frame:

$$E_{\hat{S}_{cb}} = \frac{1}{k_{end}-k_{start}+1}\sum_{k_{start}}^{k_{end}} \hat{S}(k)^2 \tag{6.3-18}$$

The frequency bin limits k_{start} and k_{end} are set to match the frequency range of the critical band. If the MDCT frame length is equal to $L_{MDCT} = 160$, the sampling rate is 8000 Hz and the critical frequency range is 1000 Hz – 1600 Hz, hence $k_{start} = 39$ and $k_{end} = 64$. Otherwise, the limits are selected as such $k_{start} = 28$ and $k_{end} = 40$.

The calculated energy $E_{\hat{S}_{cb}}$ is then subject to low pass filtering to mimic the overlap-add in the synthesis of $\hat{S}(k)$ with:

$$\tilde{E}_{\hat{S}_{cb}} = \gamma E_{\hat{S}_{cb}} + (1-\gamma)E_{\hat{S}_{cb}}^{[-1]} \tag{6.3-19}$$

where the value of low pass filter coefficient γ is equal to $\gamma = 0.61$.

The size of discontinuities is measured by averaging the step at the subframe boundaries of the filter difference signal s_{diff}. The filter difference signal is defined as the impact of the filter expressed as a negative difference signal or a correction signal or an error signal, $s_{diff} = \alpha(\hat{s} - s_p)$ where s_p is the post-filter output. The average energy of the step at the subframe boundaries $\tilde{E}_{step}$ is then:

$$\tilde{E}_{step} = \frac{1}{N_{sf}}\sum_{i=1}^{N_{sf}} \left(s_{diff}(n_i) - s_{diff}(n_i - 1)\right)^2 \tag{6.3-20}$$

where i denotes the subframe number, N_{sf} is the number of subframes and equals to $N_{sf} = 5$. The subframe indices of the subframe boundaries $n_1, n_2, \ldots, n_{N_{sf}}$ mark the start of each new subframe. A frame length of $N = 160$ would mean that $n_1 = 0, n_2 = 32, n_3 = 64, n_4 = 96, n_5 = 128$. The first sample $n_1 = 0$ in this case refers to the last sample of the previous frame $s_{diff}^{[-1]}(N-1)$.

The ratio between $\tilde{E}_{step}$ and $\tilde{E}_{\hat{S}_{cb}}$ is then calculated to provide a decision variable, $\tilde{E}_{ratio}$:

$$\tilde{E}_{ratio} = \frac{\tilde{E}_{step}}{\tilde{E}_{\hat{S}_{cb}}} \tag{6.3-21}$$

The range of the decision variable $\tilde{E}_{ratio}$ is limited with an upper limit of $E_{ratio,lim} = 2$ in the following expression:

$$\tilde{E}_{ratio,1} = \begin{cases} \tilde{E}_{ratio}, & \tilde{E}_{ratio} \leq \tilde{E}_{ratio,lim} \\ \tilde{E}_{ratio,lim}, & \tilde{E}_{ratio} > \tilde{E}_{ratio,lim} \end{cases} \quad (6.3\text{-}22)$$

Following that $\tilde{E}_{ratio,1}$ is low pass filtered with a filter coefficient of $\beta = 0.68$:

$$\tilde{E}_{ratio,LP} = \beta \tilde{E}_{ratio,1} + (1-\beta)\tilde{E}_{ratio,LP}^{[-1]} \quad (6.3\text{-}23)$$

Finally, the decision to activate the post-filter is taken by subjecting the resulting $\tilde{E}_{ratio,LP}$ value to the threshold $E_{thr} = 1$:

$$D_{bpf} = \begin{cases} active, & \tilde{E}_{ratio,LP} < E_{thr} \\ inactive, & \tilde{E}_{ratio,LP} \geq E_{thr} \end{cases} \quad (6.3\text{-}24)$$

As a result, if $D_{bpf} = active$, the post-filter is applied to the reconstructed residual signal. When $D_{bpf} = inactive$, the output is the reconstructed residual signal.

6.3.2.3.6 Residual predicted signal generation

6.3.2.3.6.1 Stereo filling

Stereo filling technology is used to estimate the component of the side channel being uncorrelated to the mid channel. This component is also called the stereo residual. The stereo filling is controlled by a control parameter, the residual prediction gain, received from the encoder, provided to a decorrelator for adaptive adjustment of the amount of decorrelation. The parameter is obtained as described in clause 5.3.2.4.8 and relates to the performance of the parametric description of the input stereo audio signal. The residual prediction gain is low when the input audio signal is well represented by the parametric stereo representation, and high when there are uncorrelated components that are not fully captured by the stereo model, i.e., there is a stereo residual component.

Depending on the bitrate, stereo residual coding or various stereo filling modes are utilized, as shown in Table 6.3-2.

Table 6.3-2: Stereo residual modes

	13.2 kbps	16.4 kbps	24.4 kbps	32 kbps
Residual coding	-	-	-	0-1 kHz
Enhanced stereo filling	0-6.4 kHz	0-6.4 kHz	0-8 kHz	-
FD stereo filling	> 6.4 kHz (TCX/HQ)	> 6.4 kHz (TCX/HQ)	> 8 kHz (TCX/HQ)	1-8 kHz, > 8 kHz (TCX/HQ)
TD stereo filling	> 6.4 kHz (ACELP)	> 6.4 kHz (ACELP)	> 8 kHz (ACELP)	> 8 kHz (ACELP)

At 32 kbps, the residual is encoded for the lowest frequency bands up to frequency band $b_{res\_cod\_max} - 1$, see 0.

Stereo filling is applied up to at most 16 kHz (SWB), which corresponds to at most 13 frequency bands, i.e.

$$b_{respred} = \min(N_b, 12). \quad (6.3\text{-}25)$$

where N_b is the number of decoding frequency bands.

Depending on the stereo filling mode, two different methods are applied for low frequency bands $\{b_{res\_cod\_max}, \ldots, b_0\}$ where b_0 corresponds to the frequency band not having any coefficient $k > k_0$ with

$$k_0 = \left\lfloor \frac{N_{FFT}}{2} + 0.5 \right\rfloor, \quad (6.3\text{-}26)$$

where N_{FFT} denotes the number of frequency coefficients and $\lfloor \cdot \rfloor$ denotes the floor function. For the higher bitrate, 32 kbps, low-complex FD stereo filling is applied for these lower frequency bands. For the lower bitrates, up to 24.4 kbps, enhanced stereo filling is applied, see clause 6.3.2.3.6.4.

For the higher frequency bands $\{b_0, \ldots, b_{respred}\}$ the low-complex FD stereo filling is applied for the TCX/HQ core codec mode (see 6.3.2.3.6.2). For high-band ACELP, a separate time-domain stereo filling is applied (see 6.3.2.3.6.5). The transitions between the modes are handled specially, see clause 6.3.2.3.6.2 and 6.3.2.3.6.5.3.

6.3.2.3.6.2 FD stereo filling

The FD stereo filling consists of a low-complex decorrelation of the downmix signal S_{dmx} by frequency-dependent adaptive band-wise mixing of downmix signals of past frames.

A short delay offset index i_{short} is obtained as

$$i_{short} = D_{MAX} - D_{short} + b \bmod 2\ , \tag{6.3-27}$$

where $D_{MAX} = 4$ and $D_{short} = 2$. The modulo operation mod makes the delay index offset i_{short} vary by one per consecutive frequency bands. To get the optimal decorrelation parameter per frequency band, the filter length is adapted per band such that a longer delay is set for the lower frequency bands and a shorter delay for the higher frequency bands. A long delay index offset i_{long} is obtained as

$$i_{long} = \max\left(4, \left\lfloor (\mathrm{D_{MAX}} + 4 - 1)\frac{b}{N_{bands}-1} + 0.5 \right\rfloor\right) - 4 \tag{6.3-28}$$

To make sure i_{short} is indeed corresponds to a shorter or equal delay:

$$i_{short} := \max(i_{short}, i_{long}) \tag{6.3-29}$$

As each frame consists of two subframes, even indices i_{short} and i_{long} are obtained as

$$\begin{aligned} i_{short} &:= 2\left\lfloor \frac{i_{short}}{2} \right\rfloor \\ i_{long} &:= 2\left\lfloor \frac{i_{long}}{2} \right\rfloor \end{aligned} \tag{6.3-30}$$

Corresponding delays d_{short} and d_{long} are obtained as

$$\begin{aligned} d_{short} &= D_{MAX} - i_{short} \\ d_{long} &= D_{MAX} - i_{long} \end{aligned} \tag{6.3-31}$$

Finally, offset indices to use for the stereo filling are obtained as

$$\begin{aligned} i_{short} &:= \left(i_{short} + i_{past} + 1\right) \bmod D_{MAX} \\ i_{long} &:= \left(i_{long} + i_{past} + 1\right) \bmod D_{MAX} \end{aligned} \tag{6.3-32}$$

where i_{past} corresponds to the latest index of an array in which the downmix signals has been stored.

In addition, corresponding gains g_{short} and g_{long} are obtained based on a determined target delay for the stereo filling as described in clause 6.3.2.3.6.3. However, if the delays d_{short} and d_{long} are equal,

$$\begin{aligned} g_{short} &= 1 \\ g_{long} &= 0. \end{aligned} \tag{6.3-33}$$

Further, transients are avoided in the stereo filling, by setting

$$\begin{aligned} g_{short} &= 0 \quad if \quad g_p(i_{short}, b) = -1 \\ g_{long} &= 0 \quad if \quad g_p(i_{long}, b) = -1 \end{aligned} \tag{6.3-34}$$

where $g_p(i, b)$ denotes the residual prediction gain (see $\hat{r}_i[b]$ in clause 6.3.2.3.7) of a previous frame of index i in an array of previous residual prediction gains. $g_p(i, b)$ is set to -1 for earlier frames having $attackPresent$ or $wasTransient$ flags set, otherwise it reflects the residual prediction gain for the corresponding frequency band b.

If both filter gains g_{short} and g_{long} are zero, the stereo filling component of $S_{DMX}(k)$ is set to zero. Otherwise, the decorrelator is adaptively adjusted to generate the stereo filling, estimating the component of the side channel being uncorrelated to the downmix signal, as

$$S_{DMX}(k) = g_2(m, b)\, S_{AVG}(k) \qquad \forall k \in k_b \tag{6.3-35}$$

where k_b denotes the frequency indices of frequency band b, and the weighted average $S_{AVG}(k,n)$ is

$$S_{AVG}(k) = g_{short} S_{DMX\_past}(i_{short}, k) + g_{long} S_{DMX\_past}(i_{long}, k) \quad (6.3\text{-}36)$$

The decorrelation signal strength g_2, controlling the amount of decorrelated signal in the stereo synthesis, is determined by calculation of

$$g_2(m,b) = \min\left(G_{amp}\,\overline{g_p}, (1 - G_{rest})\,\overline{g_p} + G_{rest}\, g_p^*(m,b)\right) \quad (6.3\text{-}37)$$

where $G_{amp} = 0.6$ and $\overline{g_p}$ is obtained as a weighted average

$$\overline{g_p} = g_{short}\, g_p(i_{short}, b) + g_{long}\, g_p(i_{long}, b), \text{ and} \quad (6.3\text{-}38)$$

$$g_p^*(m,b) = f\, g_p(m,b). \quad (6.3\text{-}39)$$

with

$$f = \sqrt{\frac{E_{DMX}+0.001}{E_{DMX\_past}+0.001}} \quad (6.3\text{-}40)$$

corresponding to the ratio between the energy of the current downmix spectrum $S_{DMX}(k,n)$ and averaged previous downmix signals $S_{AVG}(k,n)$ as obtained from equation (6.3-37). The energies are obtained as

$$E_{DMX}(m,b) = \sum_{k=k_b} S_{DMX}^2(k,n), \text{ and} \quad (6.3\text{-}41)$$

$$E_{DMX\_past}(m,b) = \sum_{k=k\_b} S_{AVG}^2(k,n), \quad (6.3\text{-}42)$$

where k_b is the set of frequency coefficients in band b, excluding any coefficient $k > k_0$.

For the higher frequency bands $\{b_0, \ldots, b_{respred}\}$, in TCX/HQ core codec mode, stereo filling components are obtained in accordance with equation (6.3-35). However, for frames where ACELP was used, the gains are set to zero, i.e.

$$\begin{aligned} g_{short} &= 0 \quad if \quad core\_hist\left(\tfrac{d_{short}}{2}\right) = ACELP \\ g_{long} &= 0 \quad if \quad core\_hist\left(\tfrac{d_{short}}{2}\right) = ACELP \end{aligned} \quad (6.3\text{-}43)$$

where $core\_hist$ is an array containing information of what core codec mode has been used.

In transitions from ACELP to TCX/HQ mode, the high band energy $E_{hb}(m)$ is updated as

$$E_{hb}(m) = E_{hb}(m-1) + \frac{2}{N_{FFT}} E_{DMX}(m,b), \quad (6.3\text{-}44)$$

where $E_{DMX}(m,b)$ is also covering all the coefficients up to $N_{FFT}/2$.

In the transition from TCX/HQ core codec mode to ACELP, a fixed short delay index i_{short} is determined as

$$i_{short} = D_{MAX} - D_{sf} \quad (6.3\text{-}45)$$

with $D_{sf} = 2$. The delay offset index i_{short} is then obtained as given in equation (6.3-32). Stereo filling components are obtained as in equation (6.3-36) with $g_{long} = 0$ and with g_{short} determined as

$$\begin{aligned} g_{short} &:= \tfrac{1+g_{short}}{2} \quad if \quad g_{short} \geq 0 \\ g_{short} &:= 0 \quad if \quad g_{short} < 0 \end{aligned} \quad (6.3\text{-}46)$$

6.3.2.3.6.3 Determination of stereo filling target delay

A target delay for the stereo filling is determined as basis for determining the gains g_{short} and g_{long}. The sum and maximum of the residual prediction gains over the N_{bands} frequency bands $b \in [0, \ldots, N_{bands} - 1]$ are determined as

$$max\_pg = \max_b g_p(b) \quad (6.3\text{-}47)$$

$$sum\_pg = \sum_b g_p(b) \quad (6.3\text{-}48)$$

where $g_p(b)$ are the decoded residual prediction gains (see $\hat{r}_i[b]$ in clause 6.3.2.3.7).

Then, if $sum\_pg > 0$, the mean and variation of the residual prediction gain control parameter are estimated. The mean of the residual prediction gain is estimated using a first order autoregressive process, AR(1), in accordance with

$$\overline{g_p} := \alpha_M \, sum\_pg + (1 - \alpha_M) \, \overline{g_p} \tag{6.3-49}$$

$$\widetilde{g_p} := \alpha_V \, var\_pg + (1 - \alpha_V) \, \widetilde{g_p} \tag{6.3-50}$$

where $\alpha_M = 0.1$ and α_V=0.1, and the variation of the residual prediction gain is computed as

$$var\_pg = \left| sum\_pg - \overline{g_p} \right|. \tag{6.3-51}$$

If $\overline{g_p} > 0$, a ratio of the variation and mean of the control parameter is determined as

$$\hat{g}_p = \min\left(1.5 * L_{ratio}, \frac{\widetilde{g_p}}{\overline{g_p}}\right), \tag{6.3-52}$$

where L_{ratio}=0.18.

A long-term variation to mean ratio is computed as

$$\begin{aligned} \hat{g}_{p,LT} &:= \alpha_{up} \, \hat{g}_p + \left(1 - \alpha_{up}\right)\hat{g}_{p,LT} \quad && if \quad \hat{g}_p > \hat{g}_{p,LT} \\ \hat{g}_{p,LT} &:= \alpha_{down} \, \hat{g}_p + (1 - \alpha_{down})\hat{g}_{p,LT} && else \end{aligned} \tag{6.3-53}$$

where α_{up} and α_{down} are determined based on the maximum peak gain of equation (6.3-47) as

$$\begin{cases} \alpha_{up} = 0.03 \\ \alpha_{down} = 0.05 \end{cases} \quad if \quad max\_pg > T_{pg}$$
$$\begin{cases} \alpha_{up} = 0.1 \\ \alpha_{down} = 0.001 \end{cases} \quad otherwise \tag{6.3-54}$$

where the peak gain threshold $T_{pg} = 0.6$. Subsequently, the targeted decorrelation filter length d_{target} is calculated as

$$\begin{aligned} d_{target} &= D_{short} && if \quad \hat{g}_{p,LT} \geq L_{ratio} \\ d_{target} &= \min\left(D_{long}, D_{short} + \left(D_{offset} + D_{long} - D_{short}\right)\left(1 - \frac{\hat{g}_{p,LT}}{L_{ratio}}\right)\right) && else \end{aligned} \tag{6.3-55}$$

where $D_{short} = 2$ and D_{long}=4 are two filter lengths, corresponding to a short and a long decorrelation filter, $D_{offset} = 2$. The short and long stereo filling filter gains are obtained based on the determined targeted decorrelation filter length d_{target} as

$$g_{short} = \frac{D_{long} - d_{target}}{D_{long} - D_{short}} \tag{6.3-56}$$

and

$$g_{long} = \sqrt{1 - g_{short}^2}. \tag{6.3-57}$$

6.3.2.3.6.4 Enhanced Stereo Filling

For replacing residual parts in the stereo upmix of the decoded base downmix channel for bitrates below 32 kbps, a special filling signal is generated for the lowband (i.e. all frequencies up to the core sampling rate, which is either 12.8 or 16 kHz). This so-called Enhanced Stereo Filling (ESF) signal is generated by filtering the core downmix channel (sampled at the core sampling frequency) with an allpass-based decorrelation filter. Since it is applied in time-domain the filtering is a broadband processing of the decoded core downmix signal. The ESF signal is subsequently transformed to DFT domain along with the downmix signal. In the spectral domain the two upmix channels are determined by combining weighted representations of the downmix signal and the ESF signal. The weighting and upmixing is thereby done in a band-wise manner using the energies of the two signals and the transmitted and dequantized residual prediction gain parameter in the respective stereo band.

In order to obtain the desired stereo filling signal a special allpass filter is applied to the downmix channel. This filter is designed to have short, dense impulse responses which is achieved by applying multiple stages of basic allpass filters where each stage consists of cascaded Schroeder allpass filters nested into a third Schroeder filter, i.e.

$$B(z) = H\left(\left(z^{-d_3}S(z)\right)^{-1}\right) \quad (6.3\text{-}58)$$

where

$$S(z) = \frac{g_1+z^{-d_1}}{1-g_1 z^{-d_1}}\frac{g_2+z^{-d_2}}{1-g_2 z^{-d_2}} \quad (6.3\text{-}59)$$

and

$$H(z) = \frac{g_3+z^{-1}}{1-g_3 z^{-1}}. \quad (6.3\text{-}60)$$

The design of such a basic allpass filter unit is illustrated in Figure 6.3-6.

The delays d_i and the gains g_i used in each stage are chosen to be rather small to avoid an overly reverberant filling signal. To further create dense and random-like impulse responses the delays are also selected to be mutually prime for all allpass filters.

With the input signal to the filter always sampled at the core sampling rate, which is either 12.8 kHz or 16 kHz for all DFT Stereo modes, a filter that was found to give suitable results for both possible sampling rates is

$(z) = \prod_{i=1}^{3} B_i(z)$(6.3-61)

where B_i are the cascaded basic allpass filters units with gains and delays as shown in Table 6.3-3.

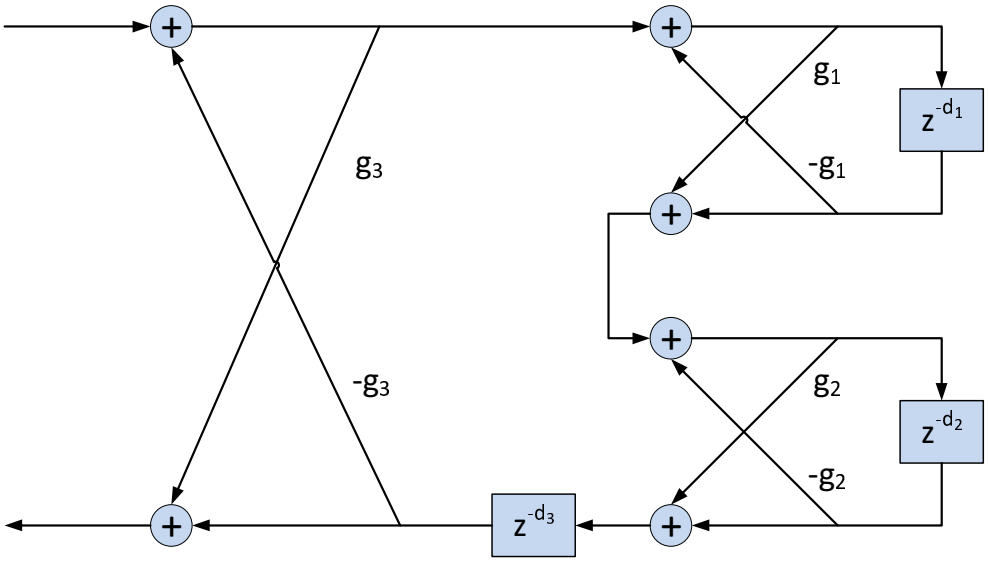


Figure 6.3-6: Basic allpass filter unit

Table 6.3-3: Allpass delays and gains

Filter	g_1	d_1	g_2	d_2	g_3	d_3
$B_1(z)$	0.5	2	-0.2	47	0.5	61
$B_2(z)$	-0.4	29	0.2	41	-0.5	73
$B_3(z)$	0.4	31	-0.3	37	0.5	59

To avoid smeared transients when applying the enhanced stereo filling signal during stereo upmix, zeroes are fed to the allpass filter units in frames where an attack was detected by the transient detector on encoder side (see clause 5.1.8 of [3]) and this control information is available at the decoder. For smooth transitions between regular input and zeroed transient frames, a small linear fade over 32 (at 12.8 kHz) or 40 samples (at 16 kHz) is done in both directions.

After generating the allpass-filtered filling signal in time-domain, the signal is brought to DFT domain via STFT analysis (see 6.3.2.3.2). As part of the transform the signal is also up-sampled from the core sampling rate to the output sampling rate to align it with the transformed downmix signal.

In DFT domain, the ESF signal now substitutes the missing residual signal $\hat{\rho}$ for all bands b that are part of the low-band and is used together with the bandwise residual gains $\hat{r}_b$ and the energy-adjusting factor $g_{norm,b}$ during the stereo upmix (see 6.3.2.3.7).

The energy-adjusting gain $g_{norm,b}$ is calculated from the energies of downmix signal $\widehat{M}_{t,b}$ and residual signal $\hat{\rho}_{t,b}$ as

$$g_{norm,b} = \sqrt{\frac{E_{\widehat{M}_{t,b}}}{E_{\hat{\rho}_{t,b}}}} \tag{6.3-62}$$

where the energies are calculated for each band as

$$HE_{\widehat{M}_{t,b}} = \sum_{k\in I_b} \left|\widehat{M}_{t,k}\right|^2 \tag{6.3-63}$$

and

$$HE_{\hat{\rho}_{t,b}} = \sum_{k\in I_b} \left|\hat{\rho}_{t,k}\right|^2. \tag{6.3-64}$$

The energies are further smoothed using previous energies $E_{\widehat{M}_{t-1,b}}$ and $E_{\hat{\rho}_{t-1,b}}$ and smoothing factor α as

$$E_{\widehat{M}_{t,b}} \leftarrow \alpha\, E_{\widehat{M}_{t-1,b}} + (1-\alpha)\, E_{\widehat{M}_{t,b}} \tag{6.3-65}$$

And

$$E_{\hat{\rho}_{t,b}} \leftarrow \alpha\, E_{\hat{\rho}_{t-1,b}} + (1-\alpha)\, E_{\hat{\rho}_{t,b}} \tag{6.3-66}$$

where α is set to 0 directly after transient frames, i.e. no smoothing, and to 0.2 for all other frames.

To obtain a smoother output and to partially compensate the ambience loss induced by low-rate coding, a compressor is applied to the energy-adjusting gain g_{norm} which compresses the values towards one.

The compression is done via

$$\tilde{g}_{norm,b} = \exp(f(\log(g_{norm,b})) \tag{6.3-67}$$

where $f(t)$ is a non-linear function defined as

$$Hf(t) = t - \int_0^t c(\tau)d\tau \tag{6.3-68}$$

and the function c satisfies

$$H0 \leq c(t) \leq 1. \tag{6.3-69}$$

The value of c around t then specifies how strongly this region is compressed, where the value 0 corresponds to no compression and the value 1 corresponds to total compression. Furthermore, the compression scheme is symmetric if c is even, i.e. $c(t) = c(-t)$.

Concretely, $c(t)$ is defined as

$$c(t) = \begin{cases} 1 & -\alpha < t < \alpha, \\ 0 & else, \end{cases} \tag{6.3-70}$$

which gives rise to

$$f(t) = t - max\{min\{\alpha, t\}, -\alpha\}. \tag{6.3-71}$$

This simplifies the calculation of $\tilde{g}_{norm}$ to

$$\tilde{g}_{norm} = g_{norm,b} min\left\{max\left\{\exp(-\alpha), \frac{1}{g_{norm,b}}\right\}, \exp(\alpha)\right\} \tag{6.3-72}$$

with α chosen as $\log(1.25)$ the final compression is given as

$$\tilde{g}_{norm,b} = g_{norm,b} min\left\{max\left\{0.8, \frac{1}{g_{norm,b}}\right\}, 1.25\right\}. \quad (6.3\text{-}73)$$

In all bands of the low-band where ESF is used, this compressed value $\tilde{g}_{norm,b}$ is then used for energy-normalization of the ESF signal.

6.3.2.3.6.5 TD stereo filling

6.3.2.3.6.5.1 General

For ACELP frames the bandwidth extension as described in clauses 5.2.6.1 and 5.2.6.2 of [3] is not transformed to DFT domain due to delay reasons. Therefore, the upmix to stereo is done in time-domain via Inter-Channel Bandwidth Extension (IC-BWE, clause 5.3.2.2.1). IC-BWE mainly aims at restoring correct panning in the bandwidth extension range. For also restoring ambience by adding a residual prediction signal to the ACELP high-band, two different methods are used depending on whether the current frame is following another ACELP frame before or is a transition frame directly after a TCX or HQ frame.

For both cases, the high-band energy of the current ACELP frame is required. This can be easily obtained by calculating it on the high-band signal in time-domain before stereo upmix.

6.3.2.3.6.5.2 Regular ACELP frame: calculation of average prediction gain and addition in time-domain

For non-transition ACELP frames, the stereo filling is done entirely in time-domain and can be seen as an extension of IC-BWE. This requires saving the high-band signal of the previous frame which will be used as the stereo filling signal. For energy normalization of the filling signal, a single normalization gain for the complete bandwidth extension range has to be calculated from the energies $E_{\widehat{m},HB}$ and $E_{\widehat{m}_{SF}}$ of the HB of input signal $\widehat{m}$ and the TD stereo filling signal $\widehat{m}_{SF}$, as well as the bandwise residual prediction parameters of the current and previous frame $\hat{r}_{t,b}$ and $\hat{r}_{t-1,b}$ for all subbands b inside the bandwidth extension range.

From the HB prediction gains weighted average gains are computed as

$$\bar{r}_t = \frac{\sum_{b\in HB}\hat{r}_{t,b}w_b}{\sum_{b\in HB}w_b} \quad (6.3\text{-}74)$$

and

$$\bar{r}_{t-1} = \frac{\sum_{b\in HB}\hat{r}_{t-1,b}w_b}{\sum_{b\in HB}w_b} \quad (6.3\text{-}75)$$

where w_b are bandwise weighting factors depending on the band resolution (see Table 5.3-6 in clause 5.3.2.4.6). The weighting factors are given as defined in Table 6.3-4 and Table 6.3-5.

Table 6.3-4: Residual gain weighting ERB 4 scale

b	9	10	11	12
w_b	1	1	1	1

Table 6.3-5: Residual gain weighting ERB 8 scale

b	4	5	6
w_b	1	0.5	0.4

Using $\bar{r}_t$ and $\bar{r}_{t-1}$ a normalized gain $\bar{r}_{norm,HB}$ is calculated as

$$\bar{r}_{norm,HB} = \min\left(\bar{r}_{t-1}, 0.4\,\bar{r}_{t-1} + 0.6\,\bar{r}_t\sqrt{\frac{E_{\widehat{m},HB}}{E_{\widehat{m}_{SF}}}}\right). \quad (6.3\text{-}76)$$

Finally, the TD stereo filling signal $\widetilde{m}_{SF}$ is added to the output signal after stereo upmix of the HB:

$$l \leftarrow l + \bar{r}_{norm,HB}\widehat{m}_{SF}, \tag{6.3-77}$$

$$r \leftarrow r - \bar{r}_{norm,HB}\widehat{m}_{SF}. \tag{6.3-78}$$

Since $\bar{g}_{norm,HB}$ was derived from residual prediction gains that were calculated in DFT domain and therefore from a windowed signal, it has faded with the gain from the previous frame in the DFT overlap region:

$$\bar{r}_{norm,HB} = win_{DFT}\bar{r}_{norm,t,HB} + (1 - win_{DFT})\bar{r}_{norm,t-1,HB} \tag{6.3-79}$$

6.3.2.3.6.5.3 TCX/HQ→ACELP transition frame: bandwise stereo filling in frequency-domain

For transition frames where the current ACELP frame was preceded by a TCX or HQ frame, the stereo filling is done in a bandwise manner in frequency domain, instead. In this case, an FD stereo filling signal is available and the normalization and addition of the filling signal can, in general, be done the same as for regular FD stereo filling in the high-band (as described in clause 6.3.2.3.6.2).

However, no bandwise energies are available for current HB signal. Therefore, only one normalization factor is calculated as

$$g_{norm} = \sqrt{\frac{E_{\widehat{m},HB}}{E_{\widehat{m}_{SF},HB}}} \tag{6.3-80}$$

where $E_{\widehat{m},HB}$ is the energy of current HB signal calculated in time-domain and $E_{\widehat{m}_{SF},HB}$ the energy of the stereo filling signal. This single normalization factor is applied for all bands in the high-band. The normalized residual gain thus becomes

$$\hat{r}_{norm,b} = g_{norm}\hat{r}_b. \tag{6.3-81}$$

6.3.2.3.7 Stereo upmix

The upmix is done in frequency domain. using the dequantized stereo parameters are calculated as:

$$\hat{L}_i[k] = \widehat{M}_i[k].\,(1 + \hat{g}_i[b]) + \hat{r}_i[b].\,g_{norm}.\,\hat{\rho}_i[k] + \widehat{Res}_i[k] \tag{6.3-82}$$

and

$$\hat{R}_i[k] = \widehat{M}_i[k].\,(1 - \hat{g}_i[b]) - \hat{r}_i[b].\,g_{norm}.\,\hat{\rho}_i[k] - \widehat{Res}_i[k] \tag{6.3-83}$$

for k ∈ I_b, where $\hat{\rho}_i[b]$ is a substitute when the decoded residual signal $\widehat{Res}_i[k]$ is missing, and g_{norm} is the energy adjusting factor:

$$g_{norm} = \sqrt{\frac{E_{\widehat{M}_i[k]}}{E_{\hat{\rho}_i[k]}}} \tag{6.3-84}$$

The channels are then rotated for reinjecting the global IPD:

$$\begin{cases} \hat{L}_i[k] = \hat{L}_i[k].\,e^{j2\pi.\widehat{gipd}} \\ \hat{R}_i[k] = \hat{R}_i[k] \end{cases} \tag{6.3-85}$$

The left and right channel waveforms are then generated by applying STFT synthesis to the reconstructed left and right spectra.

6.3.2.3.8 Stereo comfort noise injection

6.3.2.3.8.1 General

In the DFT stereo decoder the IC/ICC and ILD parameters are transmitted in the bitstream. The decoded stereo parameters are then used in stereo comfort noise injection instead of the spatial parameters estimated as part of clause 6.3.2.2.4.

6.3.2.3.8.2 Re-using decoded spatial parameters for background noise estimation

Let the decoded ICC and ILD parameters be denoted as

$$ICC_{PS}^{[m]}(b), ILD_{PS}^{[m]}(b), \quad b = 0, \ldots, B_{PS} - 1 \tag{6.3-86}$$

where B_{PS} is the number of frequency bands used by the DFT stereo decoder. The maximum frequency in the DFT stereo decoder corresponds to the last index of the last frequency band. That is

$$k_{max\_\mathrm{PS}} = \max(k(B_{PS} - 1)) \tag{6.3-87}$$

The generation of the stereo comfort noise is described in detail in clause 6.3.2.3.8.3. The generation of the stereo comfort noise requires the calculation of the mixing factor γ. In the DFT stereo mode the mixing factor γ is calculated per frequency band using the decoded stereo parameters. That is

$$\gamma(b) = \sqrt{\frac{ICC_{PS}^{[m]}(b)}{1 - ICC_{PS}^{[m]}(b)} + 1 - \left[ILD_{PS}^{[m]}(b)\right]^2} - \sqrt{\frac{ICC_{PS}^{[m]}(b)}{1 - ICC_{PS}^{[m]}(b)}}, \quad b = 0, \ldots, B_{PS} - 1 \tag{6.3-88}$$

where $ICC_{PS}^{[m]}(b)$ is the decoded inter-channel coherence parameter in the *b*th band and $ILD_{PS}^{[m]}(b)$ is the decoded inter-channel level difference parameter in the *b*th band, both defined in eq. (6.3-86).

6.3.2.3.8.3 Comfort Noise Generation and Injection

The stereo comfort noise is generated and injected in the frequency domain. Let's denote the complex spectrum of the left channel of the decoded stereo signal as $L(k)$ where $k = 0, \ldots, M - 1$ and $M = 640$ is the length of the FFT transform. Analogically, let's denote the complex spectrum of the right channel of the decoded stereo signal as $R(k)$ where $k = 0, \ldots, M - 1$. Thus, the left and the right channel of the decoded stereo signal are both sampled at 32 kHz.

The frequency resolution of the background noise spectrum estimated on the decoded core signal is $P = 25$ Hz (see clause 6.2.2.4). However, the frequency resolution of the decoded stereo signal is 50 Hz. Thus, there is a mismatch in the freqency resolution between the estimated noise spectrum and the spectrum of the decoded stereo signal. The mismatch in spectral resolution is resolved by averaging the level of the background noise in two adjacent spectral bins. This is done as follows.

The spectral envelope of the generated stereo comfort noise is controlled with the spectral envelope of the estimated background noise. The spectral envelope of the estimated backgroud noise is derived from the expanded power spectrum in eq. (6.2-49). The frequency resolution of the expanded power spectrum is compressed by the factor of two. Let's first express the minimum and the maximum level in each two adjacent frequency bins of the expanded power spectrum as

$$\begin{aligned} \tilde{P}_{min}(k) &= \min\left(\tilde{P}(2k), \tilde{P}(2k+1)\right), \quad \text{for } k = 0, \ldots, \frac{N}{2} - 1 \\ \tilde{P}_{max}(k) &= \max\left(\tilde{P}(2k), \tilde{P}(2k+1)\right), \quad \text{for } k = 0, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{6.3-89}$$

The reduction of the frequency resolution is done as follows

$$L_{CN}(k) = \begin{Bmatrix} \tilde{P}_{min}(k), & \text{if } \frac{\tilde{P}_{max}(k)}{\tilde{P}_{min}(k)} > 1.2 \\ 0.5\left(\tilde{P}(2k) + \tilde{P}(2k+1)\right), & \text{otherwise} \end{Bmatrix}, \quad \text{for } k = 0, \ldots, \frac{N}{2} - 1 \tag{6.3-90}$$

Thus, the level of the comfort noise is set to the minimum level in two adjacent frequency bins of the expanded power spectrum if the ratio between the maximum and the minimum value exceed the threshold of 1.2. This prevents excessive noise injection on signals with strong tilt of the estimated background noise. In all other situations, the level of the comfort noise is set to the average level across two adjacent frequency bins.

To prevent abrupt changes in the level of the added comfort noise it is necessary to apply a fade-in fade-out strategy for stereo comfort noise injection. For this purpose a soft VAD parameter is needed. This is achieved by a simple smoothing of the binary VAD flag as follows

$$\tilde{\mathrm{V}}_{\mathrm{fact}}^{[\mathrm{m}]} = \begin{cases} 0.7 \cdot \tilde{\mathrm{V}}_{\mathrm{fact}}^{[\mathrm{m}-1]} + 0.3 \cdot \mathrm{f}_{\mathrm{VAD}}^{[\mathrm{m}]}, & \text{if } f_{\mathrm{VAD}}^{[\mathrm{m}]} > \tilde{\mathrm{V}}_{\mathrm{fact}}^{[\mathrm{m}]} \\ 0.95 \cdot \tilde{\mathrm{V}}_{\mathrm{fact}}^{[\mathrm{m}-1]} + 0.05 \cdot \mathrm{f}_{\mathrm{VAD}}^{[\mathrm{m}]}, & \text{otherwise} \end{cases} \tag{6.3-91}$$

where it can be seen that the soft VAD parameter is limited to the range from 0 to 1. The soft VAD parameter rises more quickly when the VAD flag is changed from 0 to 1 and less quickly when it drops from 1 to 0. Thus, the fade-out period is longer than the fade-in period. During the initialization period, when $f_{CNI} = 0$, the soft VAD parameter is set to 0. That is

$$\widetilde{\mathrm{V}}_{\mathrm{fact}}^{[\mathrm{m}]} = 0, \quad \text{if } f_{\mathrm{CNI}} = 0 \tag{6.3-92}$$

The level of the comfort noise is scaled with a factor which is linearly dependent on the soft VAD parameter. As a result, the energy of the comfort noise follows the soft VAD parameter, thereby applying a fade-in fade-out effect to the injected stereo comfort noise. The re-scaling factor $\mathrm{r_{scale}(k)}$ is initialized to 0 during the initialization period, when $f_{CNI} = 0$, and updated in each frame as follows

$$\mathrm{r_{scale}(k)} = \frac{\mathrm{N}}{2}\sqrt{0.5 \cdot \mathrm{g_{scale}} \cdot \mathrm{L_{CN}(k)}}, \quad \text{for } k = 0, \ldots, N/2 - 1 \tag{6.3-93}$$

where

$$g_{scale} = 0.8 \cdot \tilde{V}_{fact}^{[m]} \tag{6.3-94}$$

Two random signals with Gaussian PDF are generated with

$$\begin{aligned} G_1(k) &\sim \mathcal{N}(0,1) \\ G_2(k) &\sim \mathcal{N}(0,1) \end{aligned} \tag{6.3-95}$$

for $k = 0, \ldots, M - 1$. The random signals are mixed together to create the left channel and the right channel of the stereo comfort noise. The mixing is done either with the mixing factor $\gamma(b)$ defined in eq. (6.3-88) if the spatial parameters were decoded from the bitstream in the DFT stereo mode. Alternatively, the mixing is done using the mixing factor γ defined in eq. (6.3-10) if the spatial parameters were estimated in the TD stereo mode using the procedure described in clause 6.3.2.2.4.

In case the spatial parameters were decoded from the bitstream in the DFT stereo mode the mixing process then takes the following form

$$\begin{aligned} \mathrm{N_L(k)} &= \mathrm{r_{scale}(k)}\left[\left(1 + \mathrm{ILD_{PS}^{[m]}(b_k)}\right)\mathrm{G_1(k)} + \gamma(\mathrm{b_k})\mathrm{G_2(k)}\right], \quad \text{for } k = 0, \ldots, \min\left(\mathrm{k_{max\_PS}}, \mathrm{N}/2 - 1\right) \\ \mathrm{N_R(k)} &= \mathrm{r_{scale}(k)}\left[\left(1 - \mathrm{ILD_{PS}^{[m]}(b_k)}\right)\mathrm{G_1(k)} - \gamma(\mathrm{b_k})\mathrm{G_2(k)}\right], \quad \text{for } k = 0, \ldots, \min\left(\mathrm{k_{max\_PS}}, \mathrm{N}/2 - 1\right) \end{aligned} \tag{6.3-96}$$

where $N_L(k)$ and $N_R(k)$ are the generated comfort noise signals for the left channel and the right channel, respectively and $\gamma(b_k)$ is the mixing factor of b_k-th band containing the *k*th frequency bin. Note, that the same mixing factor is applied to all bins of a given frequency band. Note also, that the comfort noise signals are generated only up to the maximum frequency bin supported by the DFT stereo decoder which is expressed by $\min(k_{\mathrm{max\_PS}}, N/2 - 1)$.

In case the spatial parameters were estimated in the TD stereo mode the mixing process then takes the following form

$$\begin{aligned} \mathrm{N_L(k)} &= \mathrm{r_{scale}(k)}\left[\left(1 + \widetilde{\mathrm{ILD}}_{\mathrm{LR}}^{[\mathrm{m}]}\right)\mathrm{G_1(k)} + \gamma\mathrm{G_2(k)}\right], \quad \text{for } k = 0, \ldots, N/2 - 1 \\ \mathrm{N_R(k)} &= \mathrm{r_{scale}(k)}\left[\left(1 - \widetilde{\mathrm{ILD}}_{\mathrm{LR}}^{[\mathrm{m}]}\right)\mathrm{G_1(k)} - \gamma\mathrm{G_2(k)}\right], \quad \text{for } k = 0, \ldots, N/2 - 1 \end{aligned} \tag{6.3-97}$$

The left channel and right channel of the generated stereo comfort noise signal is then injected in the decoded stereo signal as follows

$$\begin{aligned} \hat{L}_i[k] &= \hat{L}_i[k] + \mathrm{N_L(k)}, \quad \text{for } k = 0, \ldots, N/2 - 1 \\ \hat{R}_i[k] &= \hat{R}_i[k] + \mathrm{N_R(k)}, \quad \text{for } k = 0, \ldots, N/2 - 1 \end{aligned} \tag{6.3-98}$$

6.3.2.3.9 STFT synthesis

From the reconstructed left and right spectra $\hat{L}_i[k]$ and $\hat{R}_i[k]$, stereo time domain signal is synthesized by the inverse DFTs:

$$\hat{l}_i[n] = \sum_{k=0}^{N-1} \hat{L}_i[k] \cdot e^{\frac{2\pi jkn}{N}}, \quad \text{for } 0 \le n < N, \tag{6.3-99}$$

and

$$\hat{r}_i[n] = \sum_{k=0}^{N-1} \hat{R}_i[k] \cdot e^{\frac{2\pi jkn}{N}}, \quad \text{for } 0 \le n < N. \tag{6.3-100}$$

Finally, windowing and overlap-adding allow reconstructing a frame of 2.*M* samples for the two channels, by computing the following equalities at every even DFT index *i*:

$$\hat{l}'[(i/2)\cdot 2.\mathrm{M} + n - zp] = \begin{cases} \hat{l}_{i-2}[2M - zp - L1 + n]\cdot w_{dec}[1, 2M - zp - L1 + n] + \hat{l}_{i-1}[n]\cdot w_{dec}[0,n], & \text{for } zp \le n < zp + L1 \\ \hat{l}_{i-1}[n] \quad , & \text{for } zp + L1 \le n < zp + (2M - L1 - L2)/2 \\ \hat{l}_{i-1}[n]\cdot w_{dec}[0,n] + \hat{l}_i[\alpha + n]\cdot w_{dec}[1, \alpha + n], & \text{for } zp + (2M - L1 - L2)/2 \le n < zp + (2M - L1 + L2)/2 \\ \hat{l}_i[\alpha + n] \quad , & \text{for } zp + (2M - L1 + L2)/2 \le n < zp + 2M - L1 \end{cases} \tag{6.3-101}$$

and

$$\hat{r}'[(i/2)\cdot 2.\mathrm{M} + n - zp] = \begin{cases} \hat{r}_{i-2}[2M - zp - L1 + n]\cdot w_{dec}[1, 2M - zp - L1 + n] + \hat{r}_{i-1}[n]\cdot w_{dec}[0,n], & \text{for } zp \le n < zp + L1 \\ \hat{r}_{i-1}[n] \quad , & \text{for } zp + L1 \le n < zp + (2M - L1 - L2)/2 \\ \hat{r}_{i-1}[n]\cdot w_{dec}[0,n] + \hat{r}_i[\alpha + n]\cdot w_{dec}[1, \alpha + n], & \text{for } zp + (2M - L1 - L2)/2 \le n < zp + (2M - L1 + L2)/2 \\ \hat{r}_i[\alpha + n] \quad , & \text{for } zp + (2M - L1 + L2)/2 \le n < zp + 2M - L1 \end{cases} \tag{6.3-102}$$

where $\alpha = -zp - L1 - (2M - L1 - L2/2)$ is the offset between the beginning points of the 2 windows within a frame of *2M* samples. The synthesis windows are the same as the analysis windows at the decoder side.

6.3.2.3.10 DFT-based stereo PLC

6.3.2.3.10.1 General

In case of a packet loss, the PLC operation is activated for the DFT stereo. The decoded down-mix signal is generated by running the down-mix decoder PLC to obtain a down-mix PLC frame $\hat{s}_M(n)$. A DFT representation of the down-mix PLC frame $\hat{S}_M(k)$ is obtained in the same way as in error-free decoding as described by $\widehat{M_l}$[k] in (6.3-11) in 6.3.2.3.2. The stereo parameters that were decoded in the previous frame are generally reused as substitution parameters together with $\hat{s}_M(n)$ in the same was as in 6.3.2.3.10.2, where the side gain prediction residual concealment frame (if present) is generated as described in 6.3.2.3.10.3. The generated down-mix PLC frame $\hat{S}_M(k)$ is used together with the substituted parameters and the generated side prediction residual $\tilde{S}_R(k)$ to perform a DFT stereo synthesis as described in 6.3.2.3.7 and 6.3.2.3.9.

6.3.2.3.10.2 Side gain recovery after frame loss

During error-free decoding of active frames, the side gain parameter $\hat{g}_S[b]$ for band b is decoded as described in 6.3.2.3.3. To improve the error recovery after a frame loss, a prediction memory corruption flag $g_{S,CORRUPT}$ is maintained. It is initialized to $FALSE$, and upon reception of a bad frame it is set to $TRUE$. Once a non-predictive decoding of the side gain parameter $\hat{g}_S[b]$ (see $\hat{g}$ in (6.3-15) and (6.3-16) in clause 6.3.2.3.3) is received, the flag is restored to $FALSE$. If $g_{S,CORRUPT} = FALSE$ during error free decoding, a representation of the location of the source can be found by low-pass filtering the mean of $\hat{g}_S[b]$ according to

$$\begin{cases} \bar{g}_{S,LP} = 0.425\bar{g}_S + 0.575\bar{g}_{S,LP}^{[-1]}, & |\bar{g}_S| \ge 0.6 \\ \bar{g}_{S,LP} = 0, & |\bar{g}_S| < 0.6 \\ \bar{g}_S = \frac{1}{N_{bands}}\sum_{b=0}^{N_{bands}-1}\hat{g}_S[b] \end{cases} \tag{6.3-103}$$

where N_{bands} is the number of DFT stereo bands currently in use for $\hat{g}_S[b]$. The side gain parameter $\hat{g}_S[b]$ largely controls the level difference, and average side gain can be interpreted as a location of the source. If $g_{S,CORRUPT} = TRUE$, the value from the previous frame is used.

$$\bar{g}_{S,LP} = \bar{g}_{S,LP}^{[-1]} \tag{6.3-104}$$

When a bad frame is indicated, the side gain parameter from the previous frame is used $\hat{g}_S[b] = \hat{g}_S^{[-1]}[b]$. If a good frame is received, and after decoding of $\hat{g}_S[b]$ and updating of the $\bar{g}_{S,LP}$ it is found that the following two conditions are true

$$\begin{cases} g_{S,CORRUPT} = TRUE \\ \bar{g}_{S,LP} > 0.6 \ \vee \ \bar{g}_{S,LP} < -0.6 \end{cases} \quad (6.3\text{-}105)$$

where ∨ denotes inclusive OR, the decoded side gain of the previous frame is used, $\hat{g}_S[b] := \hat{g}_S^{[-1]}[b]$, replacing the decoded side gain parameters $\hat{g}_S[b]$. The condition $|\bar{g}_{S,LP}| > 0.6$ indicates that the reconstructed sound source is stable and concentrated to either the left or the right channel in the recently decoded frames. This is a situation which is sensitive to error-propagation and reusing the previously decoded parameters improves the performance in the state of corrupted memory.

6.3.2.3.10.3 Side prediction residual PLC

In case of a packet loss for a DFT stereo frame where a decoded side prediction residual is present, the PLC operation is activated to produce a concealment frame of the side prediction residual. This is achieved by combining the Phase ECU with the predicted stereo residual obtained by the stereo filling algorithm as described in 6.3.2.3.6.2. First, the DFT domain down-mix PLC frame $\hat{S}_M(k)$ is run through the frequency domain decorrelator (see decorrelation of S_{dmx} in 6.3.2.3.6.2) to obtain $\hat{S}_D(k)$. The magnitude of the previously decoded side prediction residual $\hat{S}_R^{[-1]}(k)$ is combined with the phase from $\hat{S}_D(k)$ to retain the correlation property with respect to the down-mix PLC frame $\hat{S}_M(k)$. This could be done by matching the magnitude of $\hat{S}_D(k)$ with the magnitude of $\hat{S}_R^{[-1]}(k)$. A low-complex adjustment is made by matching the order of the absolute values of the real and imaginary part and the signs for each bin of $\hat{S}_R^{[-1]}(k)$ with each bin of $\hat{S}_D(k)$, as expressed in (6.3-108). Following this principle, the phase matched $\tilde{S}_R(k)$ is calculated according to

$$\begin{cases} \tilde{S}_R(k) = a + jb \\ a = sign\left(Re\left(\hat{S}_D(k)\right)\right)c \\ b = sign\left(Im\left(\hat{S}_D(k)\right)\right)d \end{cases} \quad (6.3\text{-}106)$$

where c, d is

$$\begin{cases} c = \left|Re\left(\hat{S}_R^{[-1]}(k)\right)\right| \\ d = \left|Im\left(\hat{S}_R^{[-1]}(k)\right)\right| \end{cases} \quad (6.3\text{-}107)$$

in the case where the order of the absolute values for the real and imaginary components are the same, i.e.

$$\left|Re\left(\hat{S}_D(k)\right)\right| \geq \left|Im\left(\hat{S}_D(k)\right)\right| \wedge \left|Re\left(\hat{S}_R^{[-1]}(k)\right)\right| \geq \left|Im\left(\hat{S}_R^{[-1]}(k)\right)\right| \quad (6.3\text{-}108)$$

and otherwise

$$\begin{cases} c = \left|Im\left(\hat{S}_R^{[-1]}(k)\right)\right| \\ d = \left|Re\left(\hat{S}_R^{[-1]}(k)\right)\right| \end{cases} \quad (6.3\text{-}109)$$

An example of the low complex phase matching is illustrated in Figure 6.3-7, where the operation moves the phase within the correct $\pi/4$ section of the unit circle.

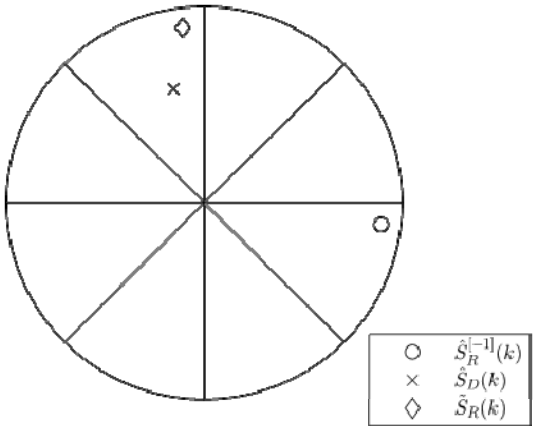


Figure 6.3-7: Low complex phase matching within $\pi/4$ of target

The Phase ECU algorithm as described in [4] 5.4.3.5.2 and 5.4.3.5.3 is applied on $\hat{S}_R^{[-1]}(k)$, where the N_{peaks} peaks $k_p(i), i = 0, \dots, N_{peaks} - 1$ are identified and then refined to $k'_p(i), i = 0, \dots, N_{peaks} - 1$ on a fractional frequency scale. As described in 6.3.2.3.2, the second analysis window of the subframe of the DFT stereo is a time-reversed version of the first analysis window. Since the last residual subframe is generated from the second subframe, the first ECU subframe is time-reversed to create the matching window shape. The phase adjustment for the time-reversed ECU frame is calculated according to

$$\begin{cases} \Delta\phi_i = -2\phi_i - 2\pi k'_p(i)\left(N_{step21} + N_{lost} \cdot L_{DFT}\right)/L_{DFT} \\ \phi_i = \arctan\left(\hat{S}_R^{[-1]}\left(k'_p(i)\right)\right) - f_{frac}(1/3.0329 + \pi) \\ f_{frac} = k'_p(i) - k_p(i) \end{cases} \quad (6.3\text{-}110)$$

where N_{step21} is the number of samples between the start of the second DFT stereo subframe of the previous frame to the start of the first subframe of the current frame, N_{lost} is the number of consecutively lost frames and L_{DFT} is the length of the DFT analysis frame. For the first lost frame $N_{lost} = 1$. The second subframe is not time reversed and the phase adjustment $\Delta\phi_i$ is computed similar to [4] 5.4.3.5.2, except the frame length and frame alignment gives different constants.

$$\Delta\phi_i = 2\pi k'_p(i) N_{lost} \quad (6.3\text{-}111)$$

As in [4] 5.4.3.5.3, the peaks of $\hat{S}_R^{[-1]}(k)$ are adjusted by applying the phase adjustment $\Delta\phi_i$ to the peak bins and their neighbourhood bins according to

$$\begin{cases} \hat{S}_{R,adj}(k) = \left(e^{j\Delta\phi_i}\hat{S}_R^{[-1]}(k)\right)^*, & 1st\ subframe \\ \hat{S}_{R,adj}(k) = e^{j\Delta\phi_i}\hat{S}_R^{[-1]}(k), & 2nd\ subframe \end{cases} \quad (6.3\text{-}112)$$

for $k \in K_{peak}$, where $(\cdot)^*$ denotes complex conjugate and results in a time reversal for the 1st subframe and K_{peak} denotes the set of bins that are part of the peaks and their neighbouring bins in the current frame that are within the limits of the spectrum.

$$K_{peak} = \bigcup_{i=0}^{N_{peaks}-1}\{k_p(i) - 1, k_p(i), k_p(i) + 1\} \quad (6.3\text{-}113)$$

The Phase ECU algorithm in [4] 5.4.3.5.2 and 5.4.3.5.3 is complemented with a separate source for the noise component of the spectrum. The side prediction residual concealment spectrum $\hat{S}_R(k)$ is formed by combining the phase adjusted peaks and their neighbouring bins with the energy adjusted decorrelated down-mix signal $\tilde{S}_R(k)$, i.e.

$$\hat{S}_R(k) = \begin{cases} \hat{S}_{R,adj}(k), k \in K_{peak} \\ \tilde{S}_R(k), k \notin K_{peak} \end{cases} \quad (6.3\text{-}114)$$

The non-peak bins may be seen as the noise component of the spectrum. If no peaks are found, $\hat{S}_R(k)$ will comprise only $\tilde{S}_R(k)$ the energy adjusted noise component. The concealment spectrum $\hat{S}_R(k)$ for the decoded residual signal will be transformed to time domain and included in the reconstructed stereo signal as described in 6.3.2.3.7.

6.3.3 MDCT-based stereo decoding

6.3.3.1 General Overview

The decoding process for the MDCT-based stereo comprises of the decoding of all the side parameters, namely, the parameters of the tools like TNS, SNS, TCX LTP, the IGF parameters and the stereo parameters. Then the bitrate ratio decoded from the bitstream determines the number of bits to be read for each channel and the decoding of the spectral data follows using the range decoder. After noise filling, IGF is applied, the inverse stereo processing takes place and resulting intermediate signal is de-normalized using the global ILD value parsed by the bitstream. Finally, spectral noise shaping is applied to de-whiten the signals, TNS filtering is also applied, if it is active, either before or after the de-whitening process and the inverse MDCT transform is carried out to obtain the original signals of the two channels in time-domain.

6.3.3.2 Decoding of parameters

6.3.3.2.1 TCX block configuration

As in EVS, the coding modes TCX20 or TCX10 are signalled within the bitstream for each channel. Overlap code for the current frame is formed from the short/long transform decision bit and from the binary code that is read from the bitstream for the overlap width as defined in clause 5.3.2.3 of [3].The TCX block is then configured as described in clause 6.2.4.2 of [3].

Additionally, the previous frame overlap coding mode is also read from the bitstream to prevent using a wrong length for the inverse transform if previous frame is lost.

Finally, the subframe-wise kernel type is also read from the bitstream to determine whether kernel switching, described in clause 5.3.3.3.4.2 was enabled at the encoder and which inverse transform is to be applied for the given subframe.

If the coding mode is TCX10, *i* bits are read to directly obtain the value of kernel_type[0] for the first subframe, with

i = (last_core ≠ ACELP_CORE ? 2 : 1)

and one additional bit is read to determine, depending on the least significant bit of kernel_type[0], the value of kernel_type[1] for the second subframe. Otherwise, only kernel_type[0] is used and *i* bits are read to obtain its value.

6.3.3.2.2 Core parameters decoding

For each channel the following parameters are decoded as defined in EVS [3]:

- Global gain (clause 6.2.2.2.4)
- Noise fill parameter (clause 6.2.2.2.5)
- LTP data (clause 6.2.2.2.6)
- TNS parameters (clause 6.2.2.2.7)
- IGF parameter decoding (clause 6.2.2.2.9)

6.3.3.2.3 SNS parameters decoding

6.3.3.2.3.1 Scale parameter decoding

6.3.3.2.3.1.1 General

Depending on the codec bitrate and the core-coder sampling rate, the SNS scale parameters were either encoded using a multi-stage stochastic VQ (MSVQ) or a 2-stage split/AVQ quantizer. Refer to clause 5.3.3.3.6.4, especially table Table

5.3-14 for the concrete assignment of VQ technologies to operating points. For each operating point, the corresponding dequantizer is selected according to the same table.

Decoding the SNS scale parameters includes obtaining codevectors using the transmitted VQ indices and restoring the SNS scale parameters back to left/right representation if joint encoding was used. The information about whether joint encoding was used or if the SNS scale parameters were encoded separately for each channel is read from the bitstream in form of a single signalling bit. In operating points where the MSVQ is used for quantizing the SNS scale parameters and both channels use short transform lengths, a bit for each subframe is read. The bit to determine between joint or separate encoding mode is only written by the encoder if both channels use the same transform block length as joint coding is not possible if transform lengths differ between the channels. Therefore, the decoder does not read the respective bit from the bitstream in that case and defaults to assume the separate encoding mode has been used. This makes it necessary to read and decode the transform length for both channels from the bitstream prior to decoding the SNS scale parameters.

6.3.3.2.3.1.2 2-Stage Split/AVQ dequantizer

The dequantizer for the split/AVQ case comprises a first stage split vector dequantizer that obtains an intermediate quantized SNS scale parameter vector, a second stage algebraic vector dequantizer that obtains an SNS scale parameter residual vector and a combiner stage that combines the intermediate quantized vector and the residual vector to obtain the reconstructed SNS scale parameter vector.

The dequantization process depends on the employed encoding scheme (joint or separate encoding, low-bitrate mode and inter-subframe coding for short blocks). Operation of the separate encoding mode without low-bitrate mode for long blocks in both channels will be described first followed by a description of the differences in the other encoding modes.

In separate encoding mode for long blocks in both channels, three indices are read from the bitstream per channel:

A 10-bit index for the 1st stage split VQ dequantizer $ind_{SNS,1}$

A base codebook index for the AVQ dequantizer

A Voronoi extension index for the AVQ dequantizer

The two indices for each split of the 1st stage are calculated as

$$ind_0 = ind_{SNS,1} \bmod 2^5 \tag{6.3-115}$$

and

$$ind_1 = \left\lfloor \frac{ind_{SNS,1}}{2^5} \right\rfloor. \tag{6.3-116}$$

With these, the first stage vector dequantizer obtains the dequantized vectors for each split and puts them together into the 1st stage intermediate quantized vector $\widehat{SNS}_{VQ,1}$ as given in Equation (5.3-262) The second stage vector dequantizer obtains the 2nd stage quantized residual vector $\widehat{SNS}_{VQ,2}$ as described in clause 6.1.1.2.1.6.1 of [3]. The final quantized SNS scale parameter vector is then calculated by the combiner stage by adding the two dequantized vectors after weighting the residual vector by the factor of 0.4 according to Equation (5.3-264). This is done separately for both channels.

In joint encoding mode, the SNS scale parameters were quantized in a mid/side representation so that the first set of the jointly encoded scale parameters comprises mid-scale parameters and the second set of the jointly encoded scale parameters comprises side scale parameters. The mid representation is first dequantized as described above. For decoding the side representation, first the *zero_side* flag is read from the bitstream as a single bit. If the value of this flag is 1, the side SNS scale parameter vector is set to zero and SNS decoding stops. Otherwise, a base codebook index and a Voronoi extension index for the AVQ are read from the bitstream and AVQ decoding is done as described in clause 6.1.1.2.1.6.1 of [3], to obtain the side SNS scale parameter vector, i.e.

$$\widehat{SNS}_{side}(i) = \begin{cases} \widehat{SNS}_{VQ,2}(i), if\ zero\_side = 0 \\ 0, if\ zero\_side = 1 \end{cases}. \tag{6.3-117}$$

If at least one of the channels uses short blocks, no joint encoding is possible. Instead, low-bitrate mode encoding can be indicated in the bitstream if the bitrate is 48 kbps. This is signalled using a single bit which must be read from the bitstream before decoding the SNS scale parameters. If low-bitrate mode is signalled, the second stage decoding is skipped and only the first stage split VQ decoding is run to calculate the quantized output vector. At other bitrates and if the low-bitrate signalling bit is set to zero, the SNS scale parameters for channels with long blocks are decoded as described above for the separate encoding mode with long blocks in both channels. For channels that use short blocks, two sets of SNS scale parameters need to be decoded. The first set for the first subframe is always decoded as described above for separate encoding mode with long blocks in both channels. If no low-bitrate mode is used, inter-subframe coding of the SNS scale parameters is signalled in the bitstream with one bit. This bit is read from the bitstream and if its value is one, no first stage is decoded for the second subframe's set of SNS scale parameters. Instead, only the AVQ indices are read from the bitstream and decoded as described in clause 6.1.1.2.1.6.1 of [3]. The output quantized SNS scale parameter vector is then calculated as the sum of the AVQ decoded output vector and the first subframe's decoded SNS parameter vector.

6.3.3.2.3.1.3 MSVQ

In operating points that use the MSVQ to quantize the SNS scale parameter, decoding is simply done by reading the respective codebook index from the bitstream and choosing the respective code vector from the respective codebook. The codebook to use for the lookup is chosen depending on the kind of representation in which the SNS scale parameters were quantized and the used transform block length according to Table 5.3-16. Decoding is done subframe-wise. If the transform lengths are the same in both channels, a bit is read from the bitstream to determine between left/right or mid/side encoding mode. If the bit's value is one, joint encoding mode is assumed, otherwise separate encoding mode is assumed. In case the transform lengths differ between the channels, no bit is read and separate encoding mode is assumed per default.

The MSVQ decoder output is given by

$$\widehat{SNS}(i) = \sum_{k=0}^{N} V_k\left(I_{MSVQ}(k), i\right) \quad (6.3\text{-}118)$$

where $V_k(i,j)$ is the i-th coefficient of the j-th vector in the selected codebook of stage k. V_k is selected according Table 5.3-16. If the joint encoding mode is signalled, the SNS scale parameters were transmitted in a mid/side representation together with a *zero_side* flag represented by a single bit. This flag is read from the bitstream prior to decoding the side SNS scale parameters. If the flag's value is 1, no MSVQ decoding is performed for the side SNS scale parameter vector and all of its parameters are set to zero instead.

6.3.3.2.3.1.4 Restoring left/right representation of jointly coded SNS scale parameters

If the joint encoding mode was signalled, the SNS scale parameter vectors for the respective (sub)frame were transmitted in mid/side representation and need to be converted back to a left/right representation. To achieve this, the scale parameter encoder combines the two vectors of jointly encoded scale parameters, comprising mid or side scale parameters, respectively, to obtain the left and right scale parameters by using two different combination rules. The first combination rule includes adding the mid and side scale parameters while the second combination rule includes subtracting the side scale parameters from the mid scale parameters. Specifically, the combination is done as described in clause 5.3.3.3.6.4.5.

6.3.3.2.4 Stereo parameters decoding

6.3.3.2.4.1 ITD decoding

For the mid-high bitrates namely 48 and 64 kbps, where the time-domain audio signals are time-aligned depending on the inter-channel time difference (ITD) as described in clause 5.3.3.2, the ITD value is read from the bitstream and de-quantized. Then the ITD is re-introduced between the two stereo channels as described in 6.3.1.2.

6.3.3.2.4.2 Stereo mode of the core bands

The stereo encoding mode is decoded from the bitstream and it is determined whether the encoded audio signal is encoded using full mid-side (MS_FULL), full dual-mono (DUAL_MONO) or band-wise encoding mode (BW_MS).

If the stereo mode is:

- DUAL_MONO then the MS mask is set to 0 for all core spectral bands $i = 0 \ldots n_{Bands,core} - 1$;

- MS_FULL then the MS mask is set to 1 for all core spectral bands $i = 0 \ldots n_{Bands,core} - 1$;
- BW_MS then the MS mask value of the m_i is read from the bitstream for each spectral band $i = 0 \ldots n_{Bands,core} - 1$;

6.3.3.2.4.3 Global ILD decoding

The global ILD is read from the bitstream depending on the core coding mode and on whether the core coding mode is in sync in both channels. More specifically, if the core coding mode is the same for the channels and equal to TCX20 then the one global ILD value per frame is read by decoding b_{ILD} bits. If the core coding mode is TCX10 for both channels, then the global ILD value is read for each subframe. If the core coding mode is different between channel, then only one global ILD value is read, and for the channel with TCX10 the same global ILD is used for both subframes.

6.3.3.2.4.4 Stereo mode of the IGF bands

Subsequently, for the bitrates that intelligent gap filling is used, the stereo mode for the IGF bands is read from the bitstream. Similarly, as in described in clause 6.3.3.2.4.2 the MS mask is set to 1 if the stereo mode is MS_FULL or is set to 0 if stereo mode is DUAL_MONO or is parsed from the bitstream for each IGF spectral band $i = n_{Bands,core} \ldots n_{Bands,IGF} - 1$.

6.3.3.2.4.5 Split ratio and bitrate distribution

Finally, the split ratio $\hat{r}_{split}$ for the distribution of the remaining bits between the two channels is parsed from the bitstream. The procedure as described in clause 5.3.3.7 is applied to determine the number of bits to be read for each channel to decode the spectral coefficients using the range coder.

6.3.3.3 Decoding process

6.3.3.3.1 Spectral data decoding and noise filling

The low-level arithmetic decoding routines are replaced in IVAS by the optimized arithmetic decoding routines (range coder), which is described in clause 6.2.2.3.3 of the present document. The remaining processing for each channel is the same as in EVS and clause 6.2.2.3 of [3]. More specifically:

- Global gain decoding as described in clause 6.2.2.3.3
- Residual bits decoding and global gain adjustment as described in clause 6.2.2.3.4
- Noise filling as described in clause 6.2.2.3.6

6.3.3.3.2 Application of stereo IGF

6.3.3.3.2.1 General

After the decoding and de-quantizing of the spectral coefficients of the first and second channel is performed, the parametric data for IGF decoding (6.3.4.2.2.) and the two-channel representation, i.e. the stereo mode and the M/S mask, of the IGF bands stereo IGF is applied to obtain the full-bandwidth spectra of the two channels.

For this the two-channel identification, i.e. the core stereo mode and M/S mask are used to regenerate the IGF bands.

As described at the respective encoding clause 5.3.3.6.1 stereo IGF is applied only for the case where stereo encoding is allowed in general i.e. if the coding mode is the same for both channels and therefore, the spectral resolution is the same and if either the core stereo encoding mode or the IGF stereo encoding more is not DUAL_MONO. Otherwise, the IGF is applied to each channel separately as is done for EVS and described in clause 6.2.2.3.8 of [3].

6.3.3.3.2.2 Stereo IGF apply

The application is done for the first and second channel as described in clause 6.2.2.3.8 of [3], where equation (1786) of [3] is replaced by the following steps to get the signals $X_{IGF,1}, X_{IGF,2}$.

A MDCT bin wise M/S mask m_b is generated from the core and IGF M/S masks according to clauses 5.3.3.4.5 and 5.3.3.4.6.

The mapping function m is defined in clause 5.2.2.3.3.4.7 to indicate the core-coder portion used in regenerating the IGF portion.

If $currWLevel(k)$ is 1 for any tile in either the first or the second channel, whitening according to clause 6.2.2.3.8.2 of [3] is applied to the TCX spectrum of both channels to get the whitened source spectra $X_{w,n}, n = 1,2$.

For each tile $t = 0, \ldots, nT - 1$ and using the tile border function $e(k)$ from equation (1016) of [3]:

If $currWLevel_1(k)$ is 2 for the tile of the first channel, a sequence of pseudo random numbers is generated according to 6.2.2.3.8.2 of [3], otherwise:

If $currWLevel_1(k)$ is 1 the whitened sources spectra are chosen as source spectra for reconstructing the first channel: $X_{s,n} = X_{w,n}, n = 1,2$.

If $currWLevel_1(k)$ is 0 the original TCX spectra are chosen as source spectra for reconstructing the first channel: $X_{s,n} = X_n, n = 1,2$.

The IGF independent noise filling according to clause 6.2.2.3.8.1 of [3] is applied to both source spectra.

The signal $X_{IGF,1}$ for the first channel is generated based on the mapping function m and the bin wise M/S mask m_b, chosing between the L/R and M/S representation of the source spectra that matches the required M/S or L/R representation of the IGF bin and generating the matching representation if necessary:

$$X_{IGF,1}(i) = \begin{cases} \tilde{X}_{s,1}(m(i)), X_1(i) = 0 \\ X_1(i), otherwise \end{cases}, \forall i \in [e(k), e(k+1)[\quad (6.3\text{-}119)$$

$$\tilde{X}_{s,1}(m(i)) = \begin{cases} X_{s,1}(m(i)), & m_b(i) = m_b(m(i)) \\ \sqrt{2}(X_{s,1}(m(i)) + X_{s,2}(m(i))), & otherwise \end{cases}$$

If $currWLevel_2(k)$ is 2 for the tile of the second channel, a sequence of pseudo random numbers is generated according to 6.2.2.3.8.2 of [3], otherwise:

If $currWLevel_2(k)$ is 1 the whitened sources spectra are chosen as source spectra for reconstructing the second channel: $X_{s,n} = X_{w,n}, n = 1,2$.

If $currWLevel_2(k)$ is 0 the original TCX spectra are chosen as source spectra for reconstructing the second channel: $X_{s,n} = X_n, n = 1,2$.

If $currWLevel_2(k) \neq currWLevel_1(k)$ the IGF independent noise filling according to clause 6.2.2.3.8.1 of [3] is applied to both source spectra.

The signal $X_{IGF,2}$ for the second channel is generated based on the mapping function m and the bin wise M/S mask m_b, chosing between the L/R and M/S representation of the source spectra that matches the required M/S or L/R representation of the IGF bin and generating the matching representation if necessary:

$$X_{IGF,2}(i) = \begin{cases} \tilde{X}_{s,2}(m(i)), X_2(i) = 0 \\ X_2(i), otherwise \end{cases}, \forall i \in [e(k), e(k+1)[\quad (6.3\text{-}120)$$

$$\tilde{X}_{s,2}(m(i)) = \begin{cases} X_{s,2}(m(i)), & m_b(i) = m_b(m(i)) \\ \sqrt{2}(X_{s,1}(m(i)) - X_{s,2}(m(i))), & otherwise \end{cases}$$

6.3.3.4 Stereo decoding process

6.3.3.4.1 Inverse mid-side transform

As described in clauses 6.3.3.2.4.2 and 6.3.3.2.4.4 the stereo encoding mode is retrieved from the bitstream for both the so called-core bands, i.e. up to the cross-over band to IGF and the IGF bands if IGF is active. Depending on the stereo encoding mode:

- full mid-side: the output first channel of the intermediate audio output is retrieved from the inverse mid operation of the encoded first and second channel and the second channel from the inverse side operation of the encoded first and second channel.

$$\overline{\mathrm{X}}_{1,\mathrm{q}}[\mathrm{k}] = \frac{\sqrt{(2)}}{2}\left(\widetilde{\mathrm{X}}_{1,\mathrm{q}}[\mathrm{k}] + \widetilde{\mathrm{X}}_{2,\mathrm{q}}[k]\right), \quad for\ k = 0, \ldots, \mathrm{L}_{\mathrm{frame}} - 1 \tag{6.3-121}$$

$$\overline{\mathrm{X}}_{2,\mathrm{q}}[\mathrm{k}] = \frac{\sqrt{(2)}}{2}\left(\tilde{X}_{1,q}[\mathrm{k}] - \tilde{X}_{2,q}[k]\right), \quad for\ k = 0, \ldots, L_{frame} - 1 \tag{6.3-122}$$

where: $\tilde{X}_{1,q}, \tilde{X}_{2,q}$ are the M/S encoded signals of the first and second channels;

$\bar{X}_{1,q}, \bar{X}_{2,q}$ represent the intermediate outputs, meaning they are still normalized with respect to their original energy values;

L_{frame} is the length of the frame up to the core bandwidth

- full dual-mono: the first and second channel of the encoded signals remain unaltered for the intermediate output.

- band-wise mid-side, then the stereo encoding mode for each spectral band of the total spectral bands is determined. If the stereo mode for the given spectral band:

  - is dual-mono than the spectral band of the first channel of the intermediate output is the same as the input first channel and the second channel of the intermediate output is the same as the second channels of the encoded signal.

  - is mid-side than the respective spectral band of the first intermediate channel is obtained from the inverse mid operation of the respective spectral bands of the first encoded channel and the second audio channel and the respective spectral band of the intermediate second channel is obtained from the inverse side operation of the first encoded channel and the second encoded channel.

The inverse operation to obtain the intermediate output stereo signal is depicted in Equations (6.3-123) and (6.3-124) below.

$$\overline{\mathrm{X}}_{1,q}\left[b_{offset}(i) + k\right] = \begin{cases} \frac{\sqrt{2}}{2}\left(\overline{\mathrm{X}}_{1,q}\left[b_{offset}(i) + k\right] + \overline{\mathrm{X}}_{2,q}\left[b_{offset}(i) + k\right]\right), & \text{if } m_i = 1 \\ \tilde{X}_{1,q}\left[b_{offset}(i) + k\right], & \text{otherwise} \end{cases} \tag{6.3-123}$$

$$\overline{\mathrm{X}}_{2,q}\left[b_{offset}(i) + k\right] = \begin{cases} \frac{\sqrt{2}}{2}\left(\overline{\mathrm{X}}_{1,q}\left[b_{offset}(i) + k\right] - \overline{\mathrm{X}}_{2,q}\left[b_{offset}(i) + k\right]\right), & \text{if } m_i = 1 \\ \overline{\mathrm{X}}_{2,q}\left[b_{offset}(i) + k\right], & otherwise \end{cases} \tag{6.3-124}$$

where: $i = 0,1, \cdots, n_{bands,core} - 1$;

$k = 0 \ldots N_{bins}(i) - 1$;

$\tilde{X}^{i}_{1,q}\left[b_{offset}(i) + k\right], \tilde{X}^{i}_{2,q}\left[b_{offset}(i) + k\right]$ is the k-th bin of the i-th spectral band of the retrieved encoded signals of the first and second channels;

$\overline{\mathrm{X}}^{i}_{1,q}\left[b_{offset}(i) + k\right], \overline{\mathrm{X}}^{i}_{2,q}\left[b_{offset}(i) + k\right]$ is the k-th bin of the i-th spectral band of the intermediate first and second channel output;

$b_{offset}(i)$ is the spectral offset of the i -th band defined in Table 5.3-17

m_i is the decoded M/S mask decision of the the *i*-th spectral band;

$N_{bins}(i)$is the width of the *i*-th spectral band as in number of spectral bins defined in Table 5.3-17.

For the bitrates that IGF is active, the same inverse mid-side processing is applied for the IGF bands depending on the respective stereo mode.

6.3.3.4.2 De-normalization to global ILD

Depending on the de-normalization value, either the first or second channel is de-normalized to obtain the first channel and the second channel of the decoded audio.

The de-normalization value g_{ILD} is calculated as in Equation (5.3-276), where in this case $\widehat{ILD}$ is the quantized ILD value decoded from the bitstream. Then the de-normalization processing applies as:

$$\mathrm{X}_{1,\mathrm{q}}^{\mathrm{MDCT}}[\mathrm{k}] = 1/g_{\mathrm{ILD}} \cdot \bar{\mathrm{X}}_{1,\mathrm{q}}^{\mathrm{MDCT}}[\mathrm{k}], \quad \text{for } \mathrm{k} = 0 \ldots \mathrm{L}_{\mathrm{frameTCX}} - 1, \quad \text{if } \mathrm{g}_{\mathrm{ILD}} < 1 \qquad (6.3\text{-}125)$$

or

$$X_{2,\mathrm{q}}^{MDCT}[k] = g_{ILD} \cdot \bar{X}_{2,\mathrm{q}}^{MDCT}[k], \quad \text{for } k = 0 \ldots L_{frameTCX} - 1, \quad \text{if } \mathrm{g}_{\mathrm{ILD}} > 1 \qquad (6.3\text{-}126)$$

6.3.3.5 Post-stereo decoding process

6.3.3.5.1 General

After the decoded stereo signal is obtained, the first and second channel are further processed to conduct a decoder side temporal noise shaping if it was signaled as active in the decoded parameters and a spectral noise shaping to obtain the perceptually un-whitened original audio spectra of the first and second channel. Finally, the inverse MDCT transform is performed to obtain the final time-domain first and second channel decoded audio signal.

6.3.3.5.2 Spectral noise shaping

6.3.3.5.2.1 General overview

The spectral shaping applied on the signal in the encoder using SNS needs to be reverted during decoding. Applying the inverse shaping restores the spectral curve of the original signal and shapes the quantization noise introduced into the signal to be minimally perceived. The SNS decoder comprises a scale parameter decoder that decodes the encoded SNS scale parameters which were either encoded jointly or separately and a spectral signal processor that processes the decoded MDCT spectrum by scaling the MDCT coefficients using scale factors obtained from the decoded SNS scale parameters. The encoded SNS scale parameters are decoded using one of two vector dequantizers that provide the decoded SNS scale parameters from the quantization indices read from the bitstream. After decoding the scale parameters, the scale parameter decoder interpolates them to obtain the scale factors to use for scaling the MDCT spectrum by applying each scale factor on all spectral samples in each frequency band.

6.3.3.5.2.2 Spectral Shaping

After the SNS scale parameters have been reconstructed, scale factors are interpolated from them and the inverse shaping with respect to the decoder is applied on the MDCT spectrum.

6.3.3.5.2.2.1 Interpolation of SNS scale factors

The scale parameter decoder performs the interpolation of the SNS parameters in the same way as done in the encoder, i.e. according to Equation (5.3-269) While being interpolated, the scale parameters are still in a log domain of base 2. After the interpolation is performed, the parameters are converted back to linear domain to obtain the set of scale factors used for shaping the MDCT spectrum, according to

$$g_{SNS}(b) = 2^{scf(b)}, b = 0, \ldots, 63. \qquad (6.3\text{-}127)$$

6.3.3.5.2.2.2 Scaling the MDCT spectrum

The interpolated scale factors are applied to the MDCT spectrum by the spectral signal processor in the same bands as used in the SNS encoder (see clause 5.3.3.3.6.2). This results in a scaled decoded MDCT spectrum with restored spectral shape as in the original encoded signal and perceptually shaped quantization noise. Scaling of the spectrum is done as described in clause 5.3.3.3.6.5.3.

6.3.3.5.3 Temporal noise shaping

If from the decoded TNS data TNS is active, and it is signalled from the bitstream that TNS is applied on the whitened spectrum than the TNS filtering is applied prior to the SNS.

Furthermore, if it is signalled from the bitstream, TNS filtering is applied on the intermediate signal after the SNS.

The TNS filtering is the same as for EVS and is described in detail in clause 6.2.2.3.10 of [3].

If the configuration, determined as described in clause 6.2.4.2 of [3], indicates that some sub-frames are coded using TCX5 then sub-frame containing MDCT bins of 2 TCX5 sub-frames is de-interleaved prior to the TNS filtering either before or after the de-whitening process with SNS.

6.3.3.6 Frequency-to-time domain transformations

6.3.3.6.1 General

As a final decoding step, the time-to-frequency transformation is applied on each channel to obtain the final time-domain signals of the left and right channels respectively. In clause 6.2.4 of [3], the details of the processing of the inverse MDCT, overlap-add and windowing are described. Different special windows transitions are used instead of those described in 6.2.4.4 of [3], and are described in 6.3.3.6.3

6.3.3.6.2 Kernel switching

Let kernelType be an abbreviation for kernel_type[n], the transform kernel type signaled for the subframe at index n, and let the constants MDCT_IV, MDST_IV, MDCT_II, and MDST_II be defined as inclause 5.3.3.3.4.2.

If kernelType equals MDCT_IV, an inverse MDCT (IMDCT) is applied according to 6.2.4 of [3], as described above.

If kernelType equals MDST_IV, an inverse MDST (IMDST) is applied similarly to an IMDCT, with two exceptions:

- The TDAC equation is given by equation (907) of [3], and the unfolding signs are adapted accordingly.
- The inverse DCT_{IV} kernel, defined by equation (1874) of [3], is replaced by an inverse DST_{IV} kernel: $\tilde{x}^q(n) = \sum_{k=0}^{L-1} z^q(k) \sin\left[(n+\frac{1}{2})(k+\frac{1}{2})\frac{\pi}{L}\right], n = 0,1,\dots,L-1$

If kernelType equals MDCT_II, an inverse MDCT-II is applied in the same way as an IMDCT, with two exceptions:

- The TDAC equation is, instead of by equation (907) of [3], specified as in subclause 5.3.3.3.4.2.
- The inverse DCT_{IV} kernel, defined by equation (1874) of [3], is replaced by an inverse DCT_{II} kernel: $\tilde{x}^q(n) = \sum_{k=0}^{L-1} z^q(k) \cos\left[(n+\frac{1}{2})(k+0)\frac{\pi}{L}\right], n = 0,1,\dots,L-1$

If kernelType equals MDST_II, an inverse MDST-II is applied in the same way as the IMDCT, with two exceptions:

- The TDAC equation is, instead of by equation (907) of [3], specified as in subclause 5.3.3.3.4.2.
- The inverse DCT_{IV} kernel, defined by equation (1874) of [3], is replaced by an inverse DST_{II} kernel: $\tilde{x}^q(n) = \sum_{k=0}^{L-1} z^q(k) \sin\left[(n+\frac{1}{2})(k+1)\frac{\pi}{L}\right], n = 0,1,\dots,L-1$

In other words, the above spectrum-time converter switches, based on the transmitted kernelType control information, between inverse transform kernels of a first group of transforms having different TDA symmetries at sides of a kernel (IMDCT-IV and IMDST-IV) and a second group of transforms having the same TDA symmetries at both kernel sides (IMDCT-II and IMDST-II), analogous to the encoder-side time-spectrum conversion described in clause 5.3.3.3.4.2.

6.3.3.6.3 Transition between long and short blocks with double overlap

As in [3], the overlap mode FULL is used for transitions between long ALDO windows and short symmetric windows (HALF, MINIMAL) for short block configurations (TCX10, TCX5).

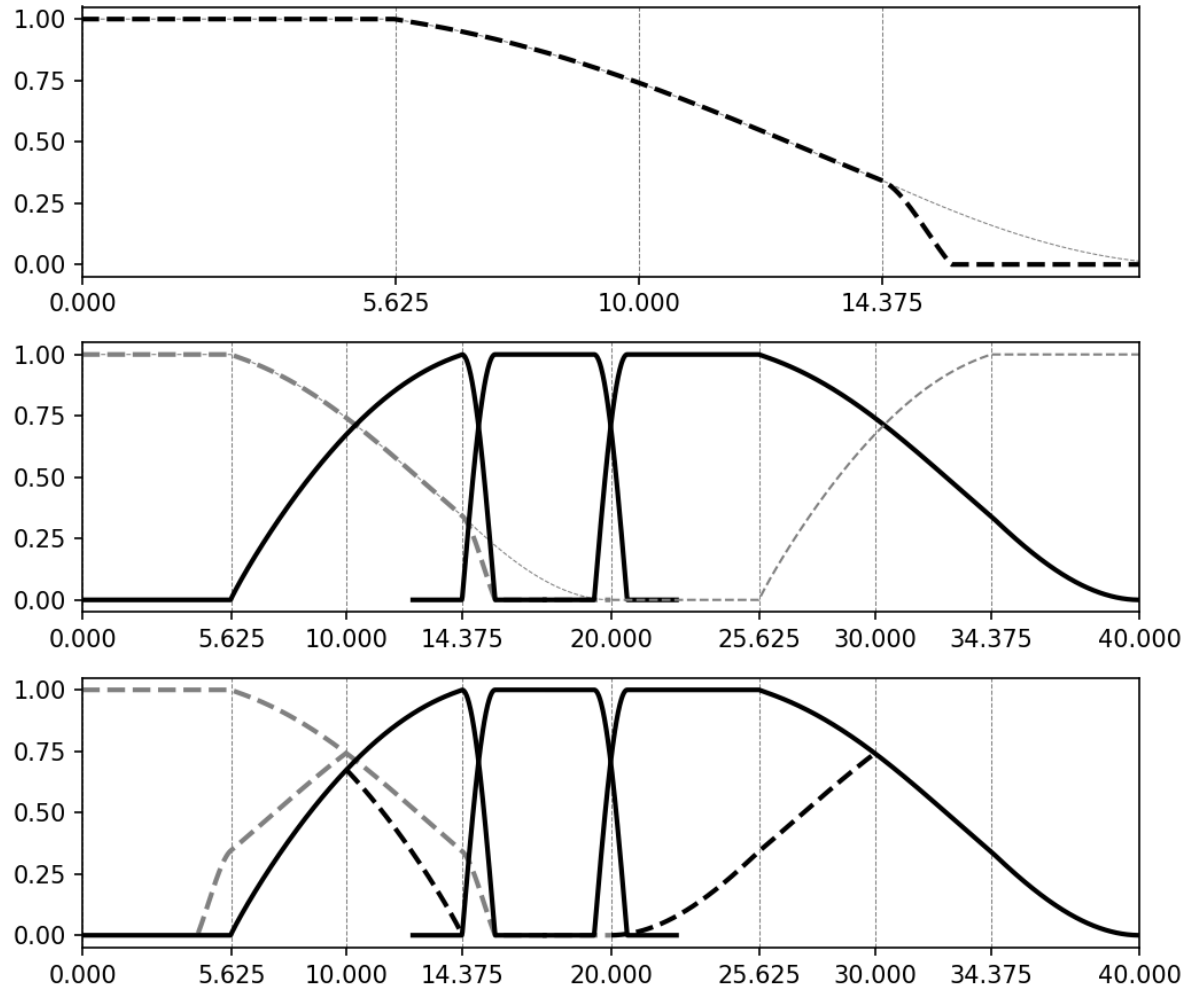


Figure 6.3-8: Window transition from ALDO to short blocks in the decoder

The right overlap of the transition window (FULL) has a length of 8.75 ms (full overlap) + 1.25 ms (minimal overlap) = 10 ms. For deriving the right overlap of the transition window (FULL), the right slope of the ALDO synthesis window (long slope, as shown with light grey dashed curve in the top plot of Figure 6.3-8) is shortened to 10 ms by zeroing out the last 4.375 ms. The last 1.25 ms of the remaining non-zero slope are multiplied with the 1.25 ms MINIMAL overlap slope. The resulting 10 ms slope is depicted with the black curve (5.625 ms-15.625 ms) in the top plot of Figure 6.3-8.

The window sequence FULL, TCX5, MINIMAL, TCX5, MINIMAL, TCX10, FULL corresponding to the overlap code 00-10 or 10-10 is shown in the middle plot of Figure 6.3-8. Details on the window sequences and the overlap codes are available in 5.3.2.3 of [3]. The 10 ms right overlap of the transition window (FULL) is shown as thick dashed grey line from 5.625 ms to 15.625 ms in the middle plot of Figure 6.3-8.

After applying the DCT to every short block in the current frame, the unfolding, windowing and overlap-adding is performed for the overlap portions between the short blocks. This is followed by unfolding and windowing the left overlap portion of the first short block, as shown by non-zero black full and dashed curves from 5.625 ms to 14.375 ms in the bottom plot of Figure 6.3-8. The unfolding and windowing of the 10 ms right overlap (FULL) part of the block transformed with the DCT in the previous frame is shown with grey dashed lines in the bottom plot of Figure 6.3-8. As the last step overlap-adding of the previous and current frame is done, the current frame consisting of the 3 short blocks whose windowing functions are shown with the full black line in the middle and the bottom plot of Figure 6.3-8.

The multi-overlap region from 14.375 ms to 15.625 ms is formed by the three windows as shown in the middle and bottom plots of Figure 6.3-8. For correctly handling the multi-overlap region, it is required to follow here described order of the unfolding, windowing and overlap-adding.

The 10 ms right overlap of the transition window (FULL) is also used for the sequences FULL, TCX10, HALF, TCX5, MINIMAL, TCX5, HALF and FULL, TCX10, MINIMAL, TCX5, MINIMAL, TCX5, MINIMAL.

Milliseconds are shown on the x axis and scalar values of the windows on the y axis in Figure 6.3-8.

6.3.3.7 PLC in MDCT-based stereo

6.3.3.7.1 Frequency-domain concealment

6.3.3.7.1.1 General

A general description of TCX frequency-domain concealment methods in EVS is given in clauses 5.4.2.3, 5.4.2.4 and 5.4.2.5 of [4]. For the IVAS MDCT stereo mode (see clause 6.3.3) some stereo-related adaptations are necessary.

6.3.3.7.1.2 Correlation-based noise substitution

For signal parts with strong correlation between the channels it is important to take that correlation into account also for concealment.

For frequency-domain concealment newly generated white noise is used to replace the noise filling on the copied version of the last good frame which is used as substitute for the following lost frame(s). However, if for a stereo signal this is applied to each channel independently, the noise will be uncorrelated which could lead to unpleasant fluctuations in the stereo image if the stereo signal is correlated. Therefore, the stereo signal should be given to the noise filling in a decorrelated representation.

For MDCT stereo, the stereo signal is always transmitted as an uncorrelated signal. Either directly, if the original signal already has very little correlation, or after a (full or bandwise) transform to a decorrelated M/S representation (see clause 5.3.3.4.7). To be able to do independent noise filling on the stereo channels, the signal therefore has to be stored before the inverse M/S decoder stage (see clause 6.3.3.4.1) during good frames. In lost frames, the noise filling is then applied to each channel of this copied signal in the same manner as in EVS FD-PLC. After the noise substitution, the signal is processed by the MDCT stereo decoder in the same way as the last good frame to obtain the output L/R representation again. The signal flow during both good and bad frames – for cases where the last good frame is (at least partial) M/S – is shown in Figure 6.3-9.

Additionally, a second inverse MDCT stereo processing has to be done earlier before the PLC mode decision described in clause 5.4.2.1 of [4]. This decision is always done in MDCT domain but can also decide on using time-domain PLC. Since there is no M/S representation available after the inverse MDCT transform to time-domain, the signal fed to the PLC mode decision should also be in an L/R representation. This means that before putting each channel into the PLC mode decision, a (temporary) L/R signal is generated by running the MDCT stereo processing on the (unprocessed) copy of the last good frame signal.

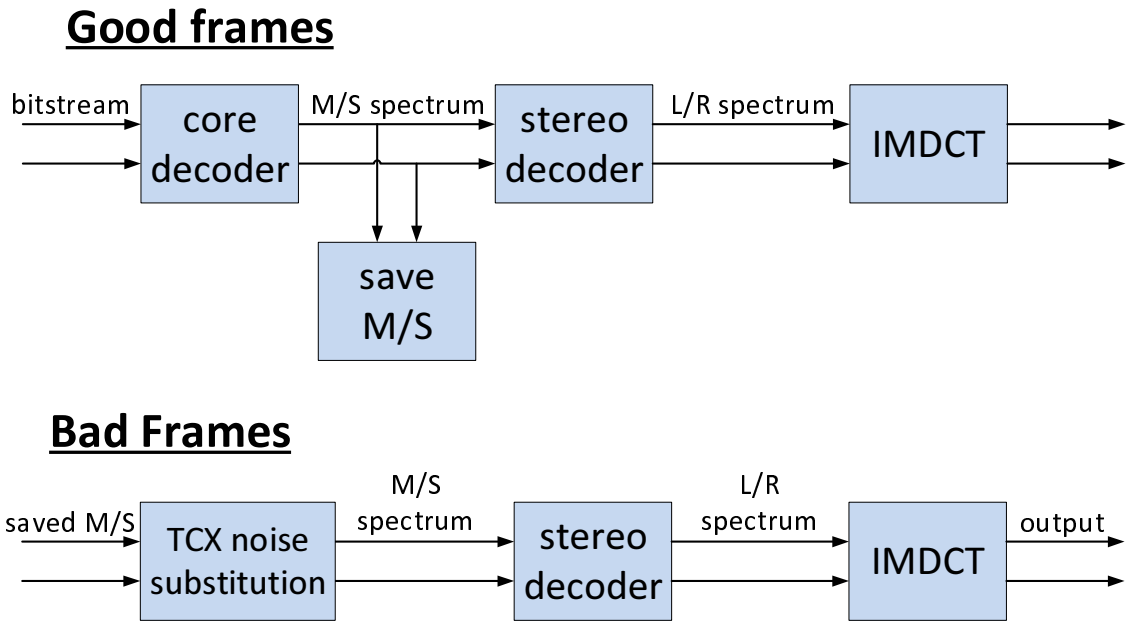


Figure 6.3-9: MDCT Stereo PLC signal flow (if last good frame was M/S)

6.3.3.7.1.3 Fading to background noise for long periods of frameloss

6.3.3.7.1.3.1 General

If a higher number of consecutive frames are lost and must be concealed, the methods for TCX frameloss concealment can behave sub-optimally. E.g. for frequency-domain PLC, frame repetition and sign scrambling can lead to annoying tonal metallic artifacts when the same frame's spectrum is repeated over and over for a longer time. Instead of repeating the signal indefinitely, the concealment signal is faded out towards a synthetic comfort noise signal with the spectral shape of the background which presents a more pleasant way of concealing such long periods of frame errors. After a maximum concealment time of 2 seconds (100 frames), the background noise signal is again faded to complete silence. This is done to clearly indicate to the user that the connection might be lost (or at least that it is severely disturbed by constant package loss).

The general process is illustrated in Figure 6.3-10. The Concealment Signal is generated by the Frequency-domain concealment algorithm (including potential tonal MDCT concealment). The Comfort Noise is generated using the background noise information calculated by the minimum statistics noise tracking as a mixture of gaussian white noise signals and it is scaled so that its energy matches the tracked background noise signal energy. Both signals are generated and faded between in the whitened MDCT domain. The fading factor *dampingFacCum* controls the crossfade between the two signals and is calculated as described in [4] clause 5.4.6.1.4, equation (223) (Note that in MDCT-Stereo mode, *stabFac* is always 1). Afterwards, the combined signal is spectrally shaped using crossfaded SNS scale factors before the resulting signal is transformed back to time domain. Finally, fading to the background level is done in time domain, as described in [4], clause 5.3.4.2.1.

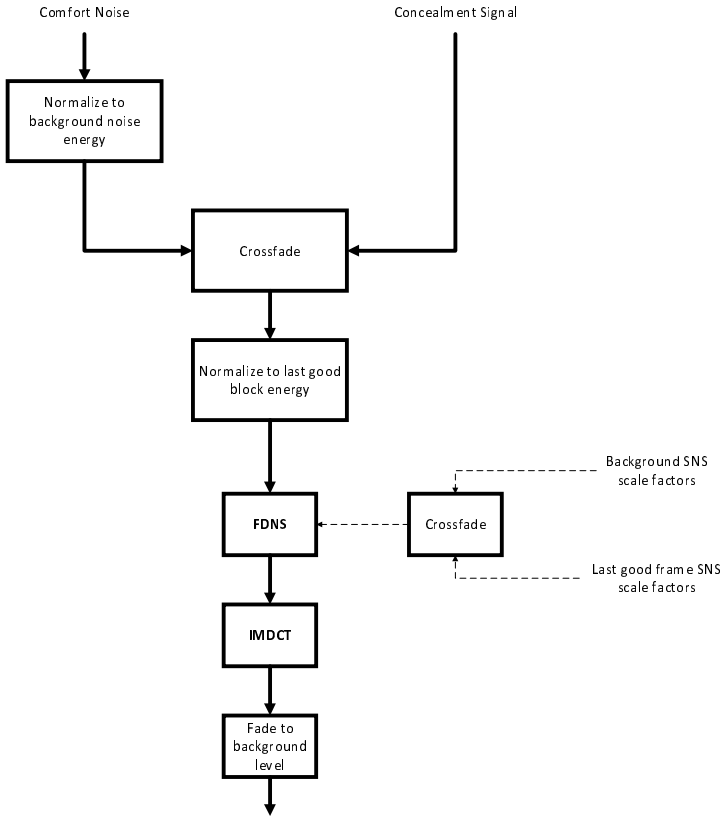


Figure 6.3-10: Fading to background noise in MDCT Stereo PLC

6.3.3.7.1.3.2 Background noise tracking in non-concealed frames

For synthesizing a background-like comfort noise, the spectral shape of the background noise needs to be estimated and tracked in the decoder during non-concealed frames. In EVS, only the overall level of the background noise is tracked using a simplified minimum statistics approach, as described in clause 5.4.6.1.1 of [3]. As not only the level, but the spectral shape of the background noise is needed, background noise tracking in MDCT-Stereo employs the complete minimum statistics noise estimation. The background noise tracking is not applied at the end of the decoding chain in

time domain, but before the inverse MDCT transform. This way, the minimum statistics algorithm is run on the reconstructed MDCT spectrum and no additional FFT of the decoded time-domain signal is needed.

The noise estimation is run only if at least one of the two channels uses long transform blocks and its corresponding VAD flag decoded from the bitstream indicates that the channel was classified as inactive by the encoder. For each channel that fulfills this condition, a power spectrum is calculated using the same approach as given by Equation (6.3-192). Here, the power spectrum is not calculated for a number of bands, but for the whole spectral range. Then, the noise estimation is run for both channels, as described in EVS [3], clause 5.6.3.2, but using the estimated power spectrum as input. If only one power spectrum was calculated, the other channel for which no power spectrum was generated re-uses the spectrum of the other channel. The rationale for this approach is that the background noise is in general diffuse and for a stereo signal should be present in both channels. This allows to update the noise estimate more often and always synchronously in both channels. The whole process is illustrated in Figure 6.3-11.

The broadband background noise target level g^{cng} is computed from the estimated power spectrum of the background noise according to:

$$\hat{g}^{cng} = \frac{1}{160}\sqrt{\sum_{k=j_{min}^{[SID]}(0)-1}^{j_{max}^{[SID]}(L_{SID}-1)} N_{FD-CNG}^{[CNG]}(k)} \qquad (6.3\text{-}128)$$

, for the definition of the involved variables see [3], clause 6.7.3.3.

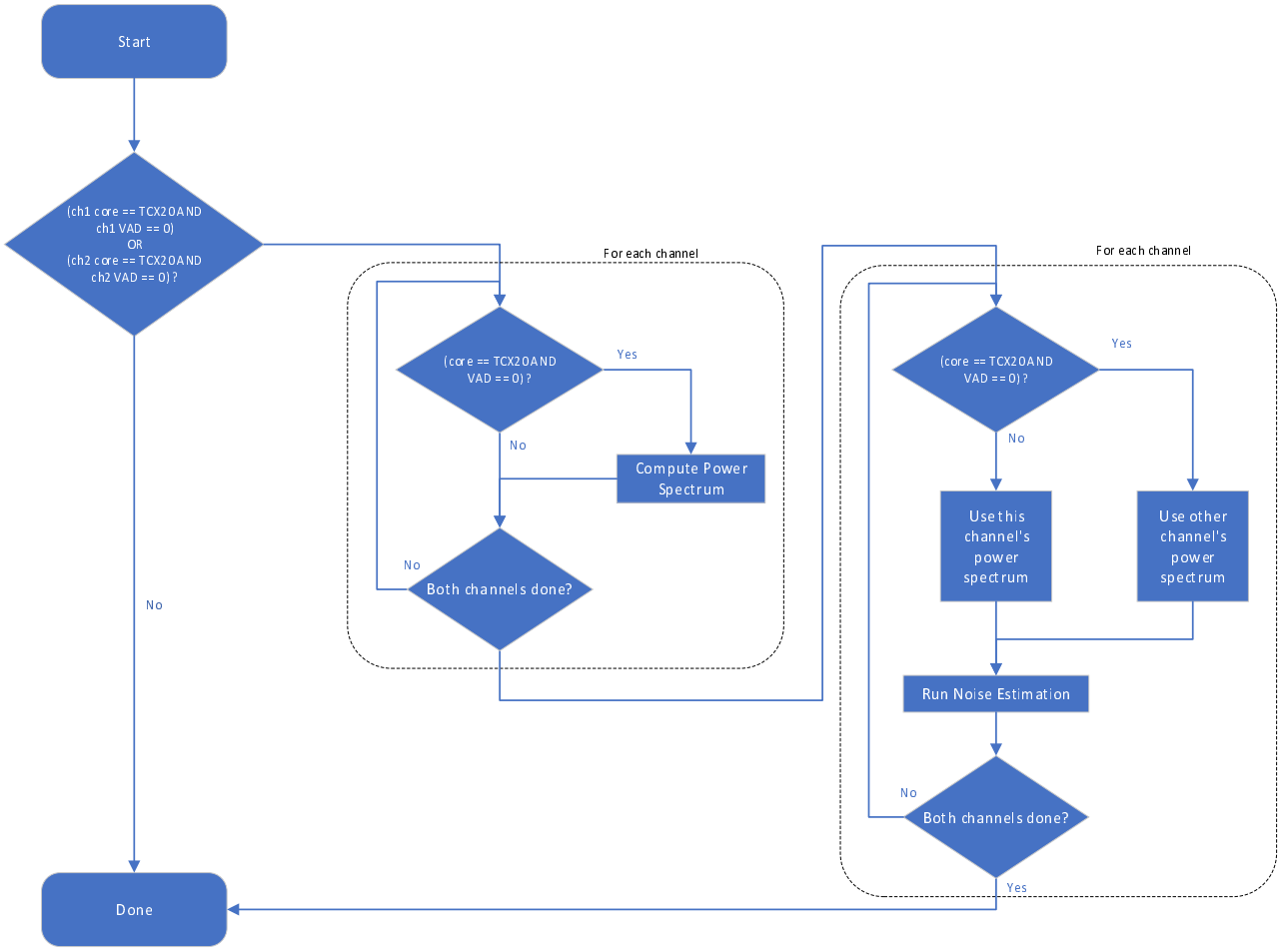


Figure 6.3-11: Background noise estimation in MDCT Stereo

6.3.3.7.1.3.3 Generating the concealment noise

In each lost frame if Frequency-domain concealment is chosen, the comfort noise for concealing longer periods of lost frames is generated using a similar approach as in CNG for MDCT-Stereo (clause 6.3.5.2.3) by mixing together noise from different uncorrelated noise sources to achieve a similar correlation between the channels of the generated comfort noise as was measured between the channels in the last decoded good frame.

As the crossfade between the generated comfort noise and the concealment signal is applied in the whitened MDCT domain, the comfort noise signal must be generated in the whitened domain as well. From the background noise tracking as described in clause 6.3.3.7.1.3.2, a spectral shape of the background noise is available in the form of a power spectrum. In the first lost frame of each frameloss period, SNS scale parameters are calculated using this power spectrum as described in clause 5.3.3.3.6.3. Then, the noise shape vector is whitened by applying the processing described in clause 5.3.3.3.6.5 and the whitened noise shape vector is stored for later use in the comfort noise generation.

Comfort noise generation is done as described in clause 6.7.3.3.2 of [3], but as the noise is generated in a real MDCT-spectrum and not in a complex FFT spectrum, no noise values are generated for the imaginary part of the spectrum. Also, the exact noise values for each MDCT coefficient are generated by weighted mixing of two noise sources as described in clause 6.3.5.2.3, with *coh* being the correlation between the two channels calculated for the last good frame according to Equation (6.3-137).

The value of the common noise source seed is saved prior to generating the noise signal and the seed is reset to this value after generating the noise signal for the first channel. For a channel that uses short transform blocks, the noise seed is also reset after generating the noise signal for the first subframe. If the transform block lengths differ between channels, the random generator in the short-block channel is run twice for each MDCT coefficient and the second value is discarded. This way, the seed has the same value as the other long-block channel at the start of the next frame.

The comfort noise for concealment is generated up to MDCT coefficient j_{max}:

$$j_{max} = \begin{cases} \min(50 f_{internal}, 50 f_{output}), \; if \; IGF \; is \; inactive \\ \min(50 f_{IGF}, 50 f_{internal}), \; if \; IGF \; is \; active \end{cases} \quad (6.3\text{-}129)$$

Here, $f_{internal}$ denotes the internal sampling rate, f_{output} the sampling rate of the decoder output and f_{IGF} the frequency above which IGF is applied. If tonal concealment is active, it can be that tonal components are detected above the IGF start frequency. In this case, j_{max} is set to the highest spectral bin of the highest tonal component found by the peak detection algorithm described in clause 5.4.2.4.2 of [3]. The energy of the generated comfort noise spectrum N is calculated as

$$E_{noise} = 0.001 + \sum_{j=1}^{j_{max}} N(j). \quad (6.3\text{-}130)$$

6.3.3.7.1.4 Fading the whitened concealment signal to noise

The normalized comfort noise signal and the concealment signal are then added as described in 5.4.6.1.3.2.1 of [3]. If tonal concealment is applied, the spectrum coefficients with detected tonal components in the last good frame are set to zero to be filled later by the tonal concealment algorithm. The energy of the noise signal is thus corrected

$$E_{noise} := E_{noise} - E_{tonal} \quad (6.3\text{-}131)$$

$$E_{tonal} = \begin{cases} \sum_{j \in I_{Tone}} N(j), \; I_{Tone} \in I_{Tones}, \; if \; tonal \; concealment \; active \\ 0, \; otherwise \end{cases} \quad (6.3\text{-}132)$$

with I_{Tones} defined as in [4], clause 5.4.2.4.2.2.

The generated comfort noise is normalized to have the same energy as was tracked for the background noise. Both signals are weighted in a crossfade-fashion, so that the combined spectrum is

$$C_{exc}^{[m]} = (1 - dampingFac) \sqrt{\frac{\hat{g}^{cng}}{E_{noise}}} N + dampingFac \; \hat{C}_{exc}^{[m]}. \quad (6.3\text{-}133)$$

Here, *dampingFac*, $C_{exc}^{[m]}$ and $\hat{C}_{exc}^{[m]}$ are defined as in [4], clause 5.4.6.1.3.2.1, Equation (213). E_{noise} is the energy of the generated comfort noise, calculated as given in [4], clause 5.4.6.1.3.2.1, Equation (211). $\hat{g}^{cng}$ is defined in equation (6.2-128).

The combined signal is again normalized to have the same energy as the last good MDCT spectrum:

$$C_{exc}^{[m]}(k) := \sqrt{\frac{E_C - E_{tonal}}{\sum_{j=0}^{j_{max}} C_{exc}^{[m]}(j)}} C_{exc}^{[m]}(k) \quad (6.3\text{-}134)$$

Here, E_C is defined as in [4], clause 5.4.6.1.3.2.1, Equation (212).

6.3.3.7.1.5 Fading the FDNS parameters to noise

In parallel with fading the whitened spectrum to the generated comfort noise, the interpolated SNS scale factors are faded from the last good frames scale factors, scf_{last}, to the scale factors calculated from the estimated background noise power spectrum, scf_{bg}. The fading starts in the first frame where *dampingFacCum* is smaller than one and is applied as a linear combination between the two sets of scale factors:

$$scf_{fadeed}(i) = fade \cdot scf_{last} + (1 - fade) \cdot scf_{bg} \quad (6.3\text{-}135)$$

with *fade* being set to 1 in the first frame where *dampingFacCum* is smaller than one and being updated in each frame according to:

$$fade := 0.95 \cdot fade. \quad (6.3\text{-}136)$$

6.3.3.7.2 Time-domain concealment

6.3.3.7.2.1 General

TCX time-domain concealment in EVS is described in clause 5.4.2.2 of [4]. In MDCT Stereo – similar to frequency-domain concealment (see clause 6.3.3.7.1) – the correlation between the channels has to be taken into account for stereo signals, particularly for the noise contribution of the generated PLC frame. Otherwise, if the same or completely independent, i.e. uncorrelated, noise is used in the two channels, it might result in a collapsing or flaring stereo image. Therefore, the noise needs to be adjusted according to signal correlation.

To this end, the correlation between the channels is calculated and saved in each good frame according to

$$coh_{lr} = \frac{|\langle G_{lr} \rangle|}{\sqrt{\langle G_{ll} \rangle \langle G_{rr} \rangle}} = \frac{|\sum_i l_i r_i|}{\sqrt{\sum_i l_i l_i \sum_i r_i r_i}}, \quad (6.3\text{-}137)$$

In lost frames, this coherence value is used to adjust the random part of the TD-PLC signal generation described in clause 5.4.2.2.2 of [4]. The adjustment parameter α is calculated as

$$\alpha = \sqrt{\frac{1-coh_{lr}}{1+coh_{lr}}}. \quad (6.3\text{-}138)$$

From α and two independently generated noise vectors $noise_1$ and $noise_2$ correctly adjusted noise vectors for the left and right channels $noise_l$ and $noise_r$ are computed as

$$noise_l = noise_1 + \alpha \cdot noise_2, \quad (6.3\text{-}139)$$

$$noise_r = noise_1 - \alpha \cdot noise_2. \quad (6.3\text{-}140)$$

The adjusted noise vectors are taken as the random part of the excitation of the left and right PLC signals. From here, the further processing for TD-PLC signal generation remains unchanged compared to mono output.

6.3.3.7.2.2 Fading to background noise for long periods of frameloss

For longer frameloss periods, also in Time-domain concealment mode, the signal is faded towards a synthesized background noise. The procedure is generally applied as described in [4], clause 5.4.6.1.2, with the following

The background noise tracking as described in clause 6.3.3.7.1.1 is run in non-concealed frames and the broadband background noise level is calculated as given in Equation (6.3-128).

- The gain compensation as described in EVS-PLC [4], clause 5.3.4.2.1, equation (109) is calculated using the LPC coefficients calculated from the background noise estimate (see clause 6.7.3.1.3 of [3]) and scaled by the factor ¼.
- If the background noise estimation has been run before, the current LPC coefficients are converted to LSF domain and faded towards the background noise LSFs. The faded LSFs are calculated as described in [4], clause 5.3.4.2.2, Equation (110), with ftarget set to the background noise LSFs.

6.3.3.7.3 Fading to zero output

Regardless of the concealment mode (Frequency domain or time domain concealment), if a frameloss period exceeds a duration of two seconds (corresponding to 100 lost frames), the output is faded towards complete zero over the next 20 frames. This is achieved by multiplying the target background noise level (see [4], clause 5.3.4.2.1) with the factor

$$g_{zero} = \begin{cases} 1, if\ n_{lost} < 100 \\ 1 - \frac{n_{lost}-100}{20}, if\ 100 < n_{lost} \leq 120, \\ 0, if\ n_{lost} > 120 \end{cases}$$

where n_{lost} is the number of lost frames in the current frameloss period.

6.3.4 Switching between stereo modes

6.3.4.1 Overview

The stereo decoder (in general the CPE decoding tool) consists of the TD stereo, DFT stereo and MDCT stereo coding modes as shown in a high-level block diagram in Figure 6.3-12.

The stereo decoder receives a bitstream transmitted from the encoder, decodes the parameters, performs a core-decoding followed by an up-mixing of the decoded transport channels, and finally produces a stereo signal including left channel $l(n)$ and right channel $r(n)$.

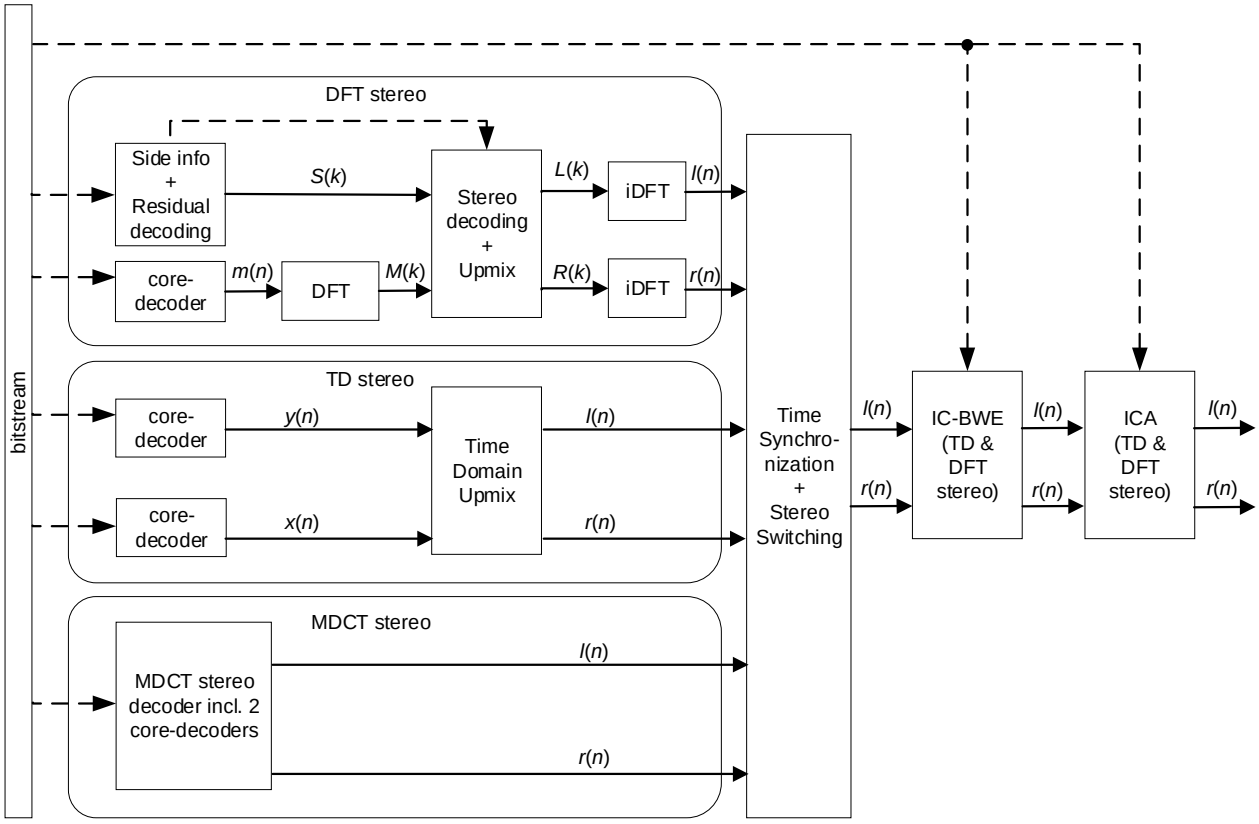


Figure 6.3-12: High-level block diagram illustrating stereo decoding modes

6.3.4.2 General

Giving the three stereo modes, there are substantial differences between the different stereo mode decoders.

The core-decoding, performed at the internal sampling rate, is basically the same regardless of the actual stereo mode; however, core-decoding is done once (mid-channel $m(n)$) for a DFT stereo frame and twice for a TD stereo frame (primary $y(n)$ and secondary $x(n)$ channels) or for a MDCT stereo frame (left $l(n)$ and right $r(n)$ channels). An issue

is to maintain (update) memories of the secondary channel $x(n)$ of a TD stereo frame when switching from a DFT stereo frame to a TD stereo frame, resp. to maintain (update) memories of the $r(n)$ channel of a MDCT stereo frame when switching from a DFT stereo frame to a MDCT stereo frame.

Moreover, further decoding operations after core-decoding strongly depend on the actual stereo mode which consequently further complicates a seamless switching between the stereo modes. The most fundamental differences between different stereo decoders are the following:

DFT stereo decoder (described in clause 6.3.2.3):

- Resampling of the decoded core synthesis from the internal sampling rate to the output stereo signal sampling rate is done in the DFT domain with a DFT analysis and synthesis overlap window having an outer overlapping part length of 3.125 ms (see Figure 6.3.1);

- The low-band (LB) bass post-filtering (BPF) adjustment in ACELP frames is done in the DFT domain;

- The core switching (ACELP core ↔ TCX/HQ core) is done in the DFT domain with an available delay of 3.125 ms;

- Synchronization between the LB synthesis and the HB synthesis (in ACELP frames) requires no additional delay;

- Stereo up-mixing is done in the DFT domain with an available delay of 3.125 ms;

- Time synchronization to match an overall CPE decoder delay (which is 3.25 ms) is applied with a length of 0.125 ms.

TD stereo decoder (described in clause 6.3.2.2):

- Resampling of the decoded core synthesis from the internal sampling rate to the output stereo signal sampling rate is done using the CLDFB filters with a delay of 1.25 ms;

- The LB BPF adjustment in ACELP frames is done in the CLDFB domain;

- The core switching (ACELP core ↔ TCX/HQ core) is done in the time domain with an available delay of 1.25 ms;

- Synchronization between the LB synthesis and the HB synthesis (in ACELP frames) introduces an additional delay;

- Stereo up-mixing is done in the TD domain with a zero delay;

- Time synchronization to match an overall CPE decoder delay is applied with a length of 2.0 ms.

MDCT stereo decoder (described in clause 6.3.3):

- Only a TCX based core-decoder is employed, so only a 1.25 ms delay adjustment is used to synchronize core synthesis signals between different cores;

- The LB BPF (in ACELP frames) is skipped;

- The core switching (ACELP core ↔ TCX/HQ core) is done in the time domain only in the first MDCT stereo frame after the TD or DFT stereo frame with an available delay of 1.25 ms;

- Synchronization between the LB synthesis and the HB synthesis is irrelevant;

- Stereo up-mixing is skipped;

- Time synchronization to match an overall CPE decoder delay is applied with a length of 2.0 ms.

The different operations during decoding, mainly the DFT vs. TD domain processing, and the different delay schemes between the DFT stereo mode and the TD stereo mode are taken into consideration in the procedure below described for switching between the DFT and TD stereo modes.

The following Table 6.3-6 lists in a sequential order the processing operations in the stereo decoder for each frame depending on the current DFT, TD or MDCT stereo mode (see also Figure 6.3-12).

Table 6.3-6: Processing operations in IVAS stereo decoder

DFT stereo mode	TD stereo mode	MDCT stereo mode
Read stereo mode & audio bandwidth information		
Memory allocation		
Stereo mode switching updates		
Stereo decoder configuration		
Core-decoder configuration		
	TD stereo decoder config.	
Core-decoding		
Core switching in DFT domain	Core switching in TD domain	
	Update of DFT stereo mode overlap memories Update MDCT stereo TCX overlap buffer	Reset / update of DFT stereo overlap memories
DFT analysis		
DFT stereo decoding incl. residual decoding		
Up-mixing in DFT domain	Up-mixing in TD domain	
DFT synthesis		
Synthesis synchronization		
IC-BWE, addition of HB synthesis		
ICA decoder		
Common stereo updates		

The switching between DFT, TD and MDCT stereo modes in the stereo decoder involves the use of the stereo mode switching mechanism to maintain continuity of the following several decoder signals and memories 1) to 6) to enable adequate processing of these signals and use of said memories in the stereo decoder:

1) Down-mixed signals and memories of core post-filters at the internal sampling rate, used at core-decoding;

- DFT stereo decoder: downmixed (mid-)channel $m(n)$;

- TD stereo decoder: primary channel $y(n)$ and secondary channel $x(n)$;

- MDCT stereo decoder: left channel $l(n)$ and right channel $r(n)$ (not down-mixed).

2) TCX-LTP post-filter memories. The TCX-LTP post-filter is used to interpolate between past synthesis samples using polyphase FIR interpolation filters (see clause 6.9.2 of [3]);

3) DFT OLA analysis memories at the internal sampling rate and at the output stereo signal sampling rate as used in the OLA part of the windowing in the previous and current frames before the DFT operation;

4) DFT OLA synthesis memories as used in the OLA part of the windowing in the previous and current frames after the IDFT operations at the output stereo signal sampling rate;

5) Output stereo signal, including channels $l(n)$ and $r(n)$; and

6) HB signal memories (see clause 6.1.5 of [3]), left and right channels, used in BWEs and IC-BWE.

While it is relatively straightforward to maintain the continuity for one channel (mid-channel $m(n)$ in the DFT stereo mode or the primary channel $y(n)$ in the TD stereo mode or $l(n)$ channel in the MDCT stereo mode, respectively) in item 1) above, it is challenging for the secondary channel $x(n)$ in item 1) above and also for signals/memories in items 2) to 6) due to several aspects, for example completely missing past signal and memories of the secondary channel, a different down-mixing, a different default delay between DFT stereo mode and TD / MDCT stereo mode, etc. Also, a shorter decoder delay (3.25 ms) when compared to the encoder delay (8.75 ms) further complicates a seamless decoding process.

In the following text, only operations responsible for the stereo mode switching are described in detail.

Reading stereo mode and audio bandwidth information

The stereo decoder (the CPE decoder in general) starts with reading the stereo mode information and audio bandwidth information from the transmitted bitstream. Based on the currently read stereo mode, the related decoding operations

are performed for each particular stereo mode (see Table 6.3-6) while memories and buffers of the other stereo modes are maintained.

Memory allocation

See details later in clause 6.3.4.3.

Stereo mode switching updates

Similarly as at the stereo encoder, the decoder stereo mode switching mechanism handles memories in case of switching from one of the DFT, TD, and MDCT stereo modes to another stereo mode. This keeps updated long-term parameters and updates or resets past buffer memories.

Upon receiving a first DFT stereo frame following a TD stereo frame or MDCT stereo frame, the stereo mode switching mechanism performs an operation of resetting the DFT stereo data structure. Upon receiving a first TD stereo frame following a DFT or MDCT stereo frame, the stereo mode switching mechanism performs an operation of resetting the TD stereo data structure. Finally, upon receiving a first MDCT stereo frame following a DFT or TD stereo frame, the stereo mode switching mechanism performs an operation of resetting the MDCT stereo data structure. Again, upon switching from one of the DFT and TD stereo modes to the other stereo mode, the stereo mode switching mechanism performs an operation of transferring some stereo-related parameters between data structures as described in the stereo encoder part in clause 5.3.4.4.2.

Updates/resets related to the secondary channel of core-decoding are described further in clause 6.3.4.4.2.

Update of DFT stereo mode overlap memories

The stereo mode switching mechanism maintains or updates the DFT OLA memories in each TD or MDCT stereo frame (See “Update of DFT stereo mode overlap memories”, “Update MDCT stereo TCX overlap buffer” and “Reset / update of DFT stereo overlap memories” of Table 6.3-6). In this manner, updated DFT OLA memories are available for a next DFT stereo frame. The actual maintaining/updating mechanism and related memory buffers are described later in clause 6.3.4.4.1.

Synthesis synchronization

To perform a stereo synthesis time synchronization and stereo switching operation (see both in Figure 6.3-12 and Table 6.3-6), the stereo mode switching mechanism comprises a time synchronizer and stereo switch to receive the channels up-mixed stereo channels $l(n)$ and $r(n)$ from the DFT stereo decoder, the TD stereo decoder or the MDCT stereo decoder and to synchronize these up-mixed output stereo channels. The time synchronizer and stereo switch delays the up-mixed output stereo channels $l(n)$ and $r(n)$ to match the core-codec overall delay value and handles transitions between the DFT stereo output channels, the TD stereo output channels and the MDCT stereo output channels.

By default, in the DFT stereo mode, the time synchronizer and stereo switch introduces a delay of 3.125 ms in the DFT stereo decoder. In order to match the core-codec overall delay of 32 ms (frame length of 20 ms, encoder delay of 8.75 ms, decoder delay of 3.25 ms), a delay synchronization of 0.125 ms is applied by the time synchronizer and stereo switch. In case of the TD or MDCT stereo mode, the time synchronizer and stereo switch applies a delay consisting of the 1.25 ms resampling delay and the 2 ms delay used for synchronization between the LB and HB synthesis and to match the overall core-codec delay of 32 ms.

Still referring to Figure 6.3-12 and Table 6.3-6, after time synchronization and stereo switching are performed, the HB synthesis (from BWE or IC-BWE) is added to the core synthesis and ICA decoding is performed before the final stereo synthesis of the left and right channels $l(n)$ and $r(n)$ is outputted from the stereo decoder. These ICA and IC-BWE operations are skipped in the MDCT stereo mode.

6.3.4.3 Memory handling

Similarly as at the stereo encoder memory handling (see clause 5.3.4.3), the decoder stereo mode switching mechanism dynamically allocates/deallocates inter-frame data structures (static memory) depending on the current stereo mode. The decoder stereo mode mechanism thus keeps the static memory impact of the CPE decoding tool as low as possible by maintaining only those parts of the static memory that are used in the current frame. The use of main static memory data structures at the decoder is similar to that at the encoder which is summarized in Table 5.3-20.

In addition, a LRTD stereo sub-mode flag is read to distinguish between the regular TD stereo sub-mode and the LRTD stereo sub-mode. Based on the sub-mode flag, the stereo mode switching mechanism allocates/deallocates related data structures within the TD stereo mode as similarly shown in Table 5.3-20.

6.3.4.4 Switching between DFT and TD stereo modes

6.3.4.4.1 Switching from TD stereo to DFT stereo

The mechanism of switching from the TD stereo mode to the DFT stereo mode in the stereo decoder is challenging by the fact that the decoding steps between these two stereo modes are fundamentally different including a transition from two core-decoders in the last TD stereo frame to one core-decoder in the first DFT stereo frame.

Figure 6.3-13 is a flow chart illustrating processing operations in the stereo decoder upon switching from the TD stereo mode to the DFT stereo mode. Specifically, Figure 6.3-13 shows two frames of the decoded stereo signal at different processing operations with related time instances when switching from a TD stereo frame to a DFT stereo frame.

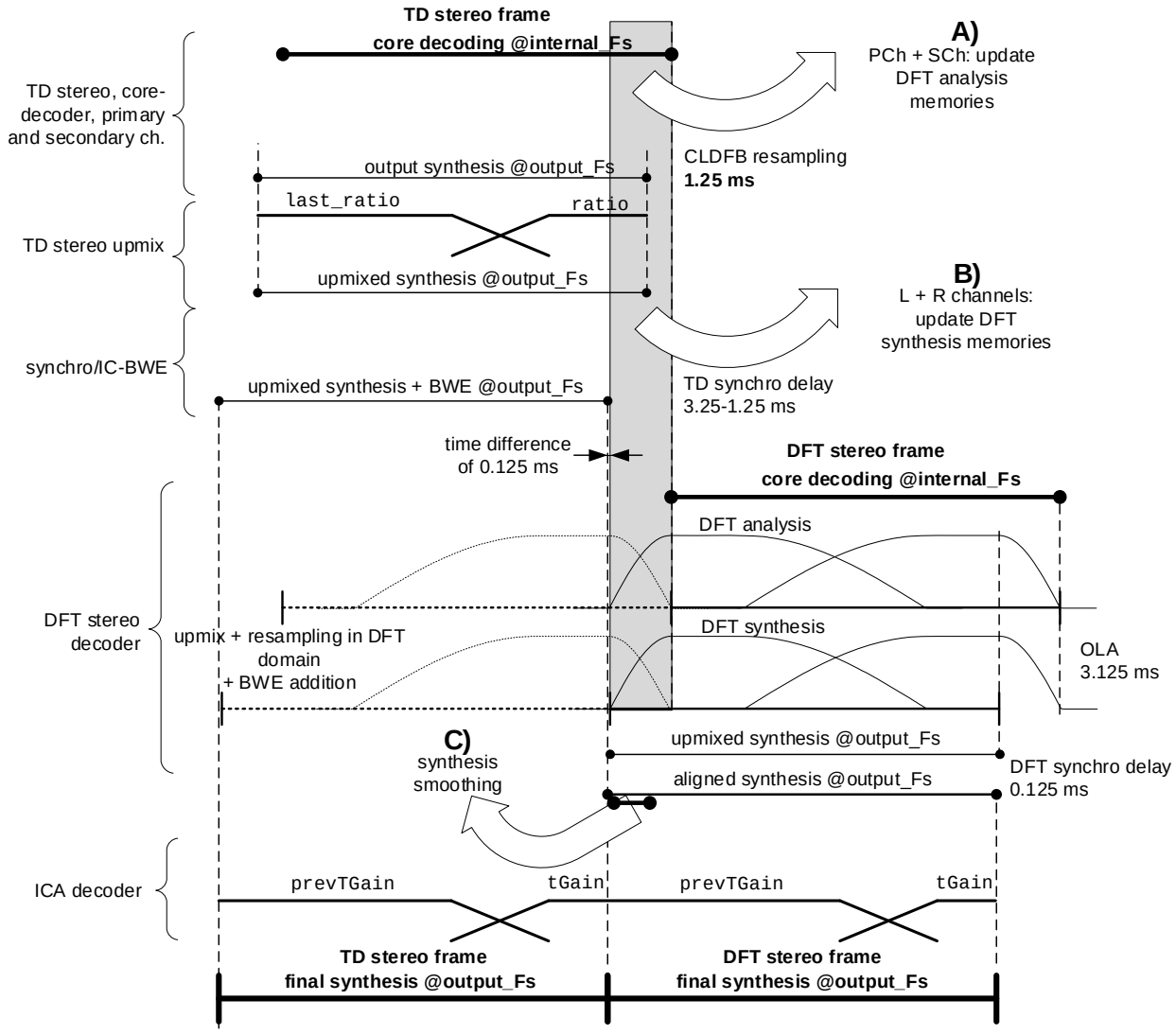


Figure 6.3-13: Flow chart illustrating operations upon switching from TD stereo to DFT stereo

First, the core-decoders of the TD stereo decoder are used for both the primary and secondary channels while each of them output the corresponding decoded core synthesis at the internal sampling rate. In the TD stereo frame, the decoded core synthesis from the two core-decoders is used to update the DFT stereo OLA memory buffers (one memory buffer per channel, i.e. two OLA memory buffers in total; See above described DFT OLA analysis and synthesis memories). These OLA memory buffers are updated in every TD stereo frame to be up-to-date in case the next frame is a DFT stereo frame.

The instance A) of Figure 6.3-13 refers to, upon receiving a first DFT stereo frame following a TD stereo frame, an operation of updating the DFT stereo analysis memories (these are used in the OLA part of the windowing in the previous and current frame before the DFT calculating operation at the internal sampling rate, $input\_mem\_LB[]$, using

the stereo mode switching mechanism. For that purpose, a number L_{ovl} of last samples of the TD stereo synthesis at the internal sampling rate of the primary channel and the secondary channel in the TD stereo frame are used by the stereo mode switching mechanism to update the DFT stereo analysis memories of the DFT stereo mid-channel $m(n)$ and the side channel $s(n)$, respectively. The length of the overlap segment, L_{ovl}, corresponds to the 3.125 ms long overlap part of the DFT analysis window, e.g. $L_{ovl} = 40$ samples at a 12.8 kHz internal sampling rate.

Similarly, the stereo mode switching mechanism updates the DFT stereo BPF analysis memory (which is used in the OLA part of the windowing in the previous and current frame before the DFT operation) of the mid-channel $m(n)$ at the internal sampling rate, $input\_mem\_BPF[]$, using L_{ovl} last samples of the BPF error signal (see clause 6.1.4.2 in [3]) of the TD stereo primary channel. Moreover, the DFT stereo Full Band (FB) analysis memory (this memory is used in the OLA part of the windowing in the previous and current frame before the DFT operation) of the mid-channel $m(n)$ at the output stereo signal sampling rate, $input\_mem[]$, is updated using the 3.125 ms last samples of the TD stereo primary channel HB synthesis (ACELP core) or the primary channel TCX synthesis, respectively. The DFT stereo BPF and FB analysis memories are not employed for the side channel $s(n)$, so that these memories are not updated using the secondary channel core synthesis.

Next, in the TD stereo frame, the decoded ACELP core synthesis (primary and secondary channels) at the internal sampling rate is resampled using CLDFB-domain filtering which introduces a delay of 1.25 ms. In case of the TCX/HQ core frame, a compensation delay of 1.25 ms is used to synchronize the core synthesis between different cores. Then the TCX-LTP post-filter is applied to both primary and secondary core channels.

At the next operation, the primary $y(n)$ and secondary $x(n)$ channels of the TD stereo synthesis at the output sampling rate from the TD stereo frame are subject to TD stereo up-mixing (combination of the primary $y(n)$ and secondary $x(n)$ channels using the TD stereo mixing ratio β in TD up-mixer (see clause 6.3.2.2) resulting in up-mixed stereo left and right channels $l(n)$ and $r(n)$ in the time-domain. Since the up-mixing operation is performed in the time-domain, it introduces no up-mixing delay.

Then, the left $l(n)$ and right $r(n)$ up-mixed channels of the TD stereo frame from the up-mixer of the TD stereo decoder are used to update the DFT stereo synthesis memories (these are used in the OLA part of the windowing in the previous and current frame after the IDFT calculating operation. Again, this update is done in every TD stereo frame by the stereo mode switching mechanism in case the next frame is a DFT stereo frame. Instance B) of Figure 6.3-13 depicts that the number of available last samples of the TD stereo left $l(n)$ and right $r(n)$ channels synthesis is insufficient to be used for a straightforward update of the DFT stereo synthesis memories. The 3.125 ms long DFT stereo synthesis memories are thus reconstructed in two segments using approximations. The first segment corresponds to the (3.125 – 1.25) ms long signal that is available (that is the up-mixed synthesis at the output stereo signal sampling rate) while the second segment corresponds to the remaining 1.25 ms long signal that is not available due to the core-decoder resampling delay.

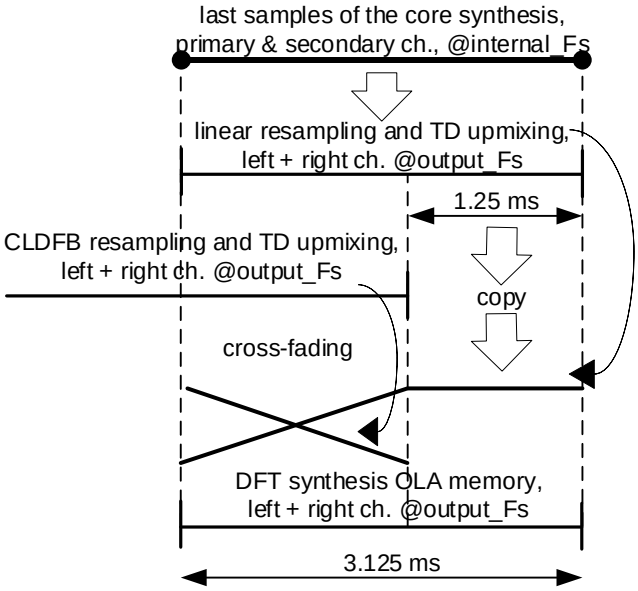


Figure 6.3-14: Flow chart illustrating an instance B) of Figure 6.3-13

Specifically, the DFT stereo synthesis memories are updated by the stereo mode switching mechanism using the following sub-operations as illustrated in Figure 6.3-14. Figure 6.3-14 is a flow chart illustrating the instance B) of Figure 6.3-13, comprising updating DFT stereo synthesis memories in a TD stereo frame on the decoder side:

(a) The two channels $l(n)$ and $r(n)$ of the DFT stereo analysis memories at the internal sampling rate, $input\_mem\_LB[]$, as reconstructed earlier during the decoding (they are identical to the core synthesis at the internal sampling rate), are subject to further processing depending on the actual decoding core:

- ACELP core: the last L_{ovl} samples of the LB core synthesis of the primary and secondary channels at the internal sampling rate are resampled to the output stereo signal sampling rate using a simple linear interpolation with zero delay.

- TCX/HQ core: the last L_{ovl} samples of the LB core synthesis of the primary and secondary channels at the internal sampling rate are similarly resampled to the output stereo signal sampling rate using a simple linear interpolation with zero delay. However, then, the TCX synchronization memory (the last 1.25 ms segment of the TCX synthesis from the previous frame) is used to update the last 1.25 ms of the resampled core synthesis.

(b) The linearly resampled LB signals corresponding to the 3.125 ms long part of the primary and secondary channels of the TD stereo frame are up-mixed to form left and right channels, using the common TD stereo up-mixing routine while the TD stereo mixing ratio β from the current frame is used. The resulting signal is further called “reconstructed synthesis”.

(c) The reconstruction of the first (3.125 – 1.25 ms) long part of the DFT stereo synthesis memories depends on the actual decoding core:

- ACELP core: A cross-fading between the CLDFB-based resampled and TD up-mixed synthesis at the output stereo signal sampling rate and the reconstructed synthesis (from the previous sub-operation (b)) is performed for both the left and right channels during the first (3.125 – 1.25) ms long part of the channels of the TD stereo frame.

- TCX/HQ core: The first (3.125 – 1.25) ms long part of the DFT stereo synthesis memories is updated using the up-mixed synthesis.

(d) The 1.25 ms long last part of the DFT stereo synthesis memories is filled up with the last portion of the reconstructed synthesis.

(e) The DFT synthesis window (see clause 6.3.2.3.2) is applied to the DFT OLA synthesis memories only in the first DFT stereo frame (if the switching from TD to DFT stereo mode happens). It is noted that the last 1.25 ms part of the DFT OLA synthesis memories is of a limited importance as the DFT synthesis window shape converges to zero and it thus masks the approximated samples of the reconstructed synthesis resulting from resampling based on simple linear interpolation.

Finally, the up-mixed reconstructed synthesis of the TD stereo frame is aligned, i.e. delayed by 2 ms in the time synchronizer and stereo switching module in order to match the codec overall delay. Specifically:

- In case there is a switching from a TD stereo frame to a DFT stereo frame, other DFT stereo memories (other than overlap memories), i.e. DFT stereo decoder past frame parameters and buffers, are reset by the stereo mode switching mechanism.

- Then, the DFT stereo decoding, up-mixing and DFT synthesis are performed and the stereo output synthesis (channels $l(n)$ and $r(n)$) is aligned, i.e. delayed by 0.125 ms in the time synchronizer and stereo switch in order to match the core-codec overall delay.

Figure 6.3-15 is a flow chart illustrating an instance C) of Figure 6.3-13, comprising smoothing the output stereo synthesis in the first DFT stereo frame following stereo mode switching, on the decoder side.

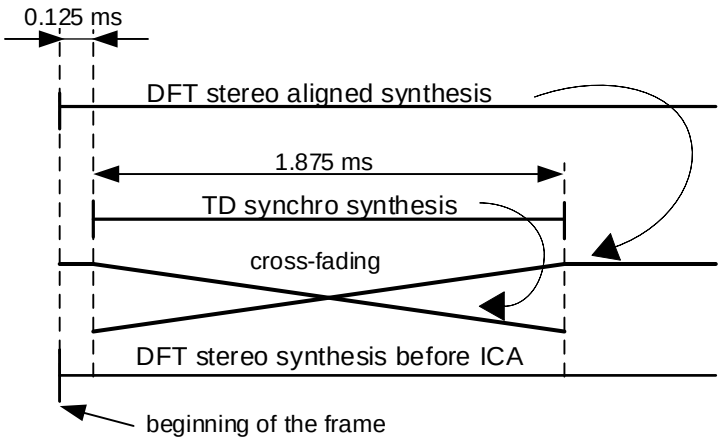


Figure 6.3-15: Flow chart illustrating an instance C) of Figure 6.3-13

Referring to Figure 6.3-15, once the DFT stereo synthesis is aligned and synchronized to the core-codec overall delay in the first DFT stereo frame, the stereo mode switching mechanism performs a cross-fading operation between the TD stereo aligned and synchronized synthesis and the DFT stereo aligned and synchronized synthesis to smooth the switching transition. The cross-fading is performed on a 1.875 ms long segment starting after a 0.125 ms delay at the beginning of both output channels $l(n)$ and $r(n)$ (all signals are at the output sampling rate). This instance corresponds to instance C) in Figure 6.3-13.

Decoding then continues regardless of the current stereo mode with the IC-BWE decoder, the ICA decoder and common stereo decoder updates (see Table 6.3-6).

6.3.4.4.2 Switching from DFT stereo to TD stereo

The fundamentally different decoding operations between the DFT stereo mode and the TD stereo mode and the presence of two core-decoders in the TD stereo decoder makes switching from the DFT stereo mode to the TD stereo mode in the stereo decoder challenging. Figure 6.3-16 is a flow chart illustrating processing operations in the stereo decoder upon switching from the DFT stereo mode to the TD stereo mode. Specifically, Figure 6.3-16 shows two frames of decoded stereo signal at different processing operations with related time instances upon switching from a DFT stereo frame to a TD stereo frame.

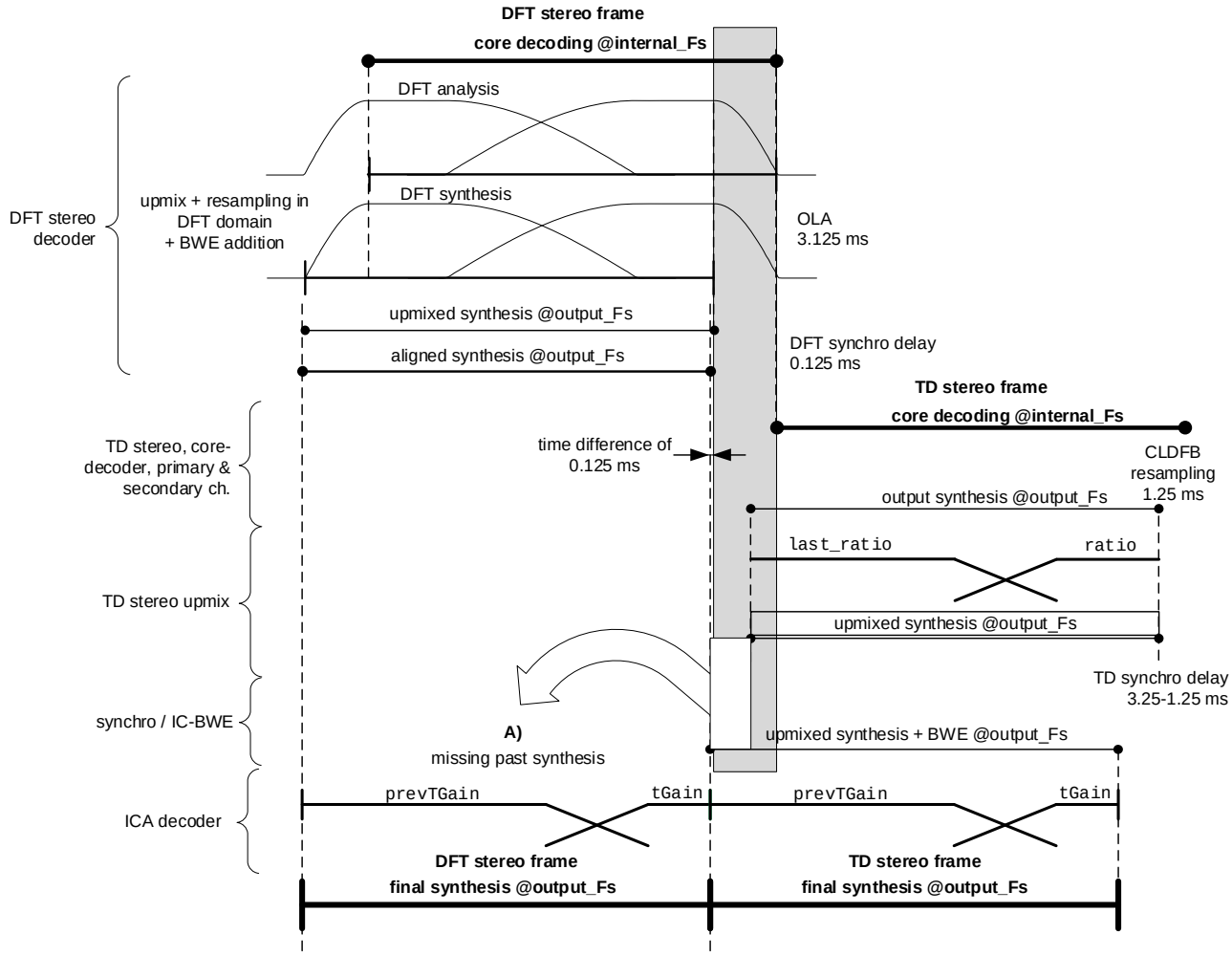


Figure 6.3-16: Flow chart illustrating operations upon switching from DFT stereo to TD stereo

The core-decoding uses the same processing regardless of the actual stereo mode with two exceptions.

First exception: In DFT stereo frames, resampling from the internal sampling rate to the output stereo signal sampling rate is performed in the DFT domain but the CLDFB resampling is run in parallel in order to maintain/update CLDFB analysis and synthesis memories in case the next frame is a TD stereo frame.

Second exception: The BPF is applied in the DFT domain in DFT stereo frames while the BPF analysis and computation of error signal is done in the time-domain regardless of the stereo mode.

Otherwise, all internal states and memories of the core-decoder are simply continuous and well maintained when switching from the DFT mid-channel $m(n)$ to the TD stereo primary channel $y(n)$.

In the DFT stereo frame, decoding then continues with core-decoding of mid-channel $m(n)$, calculation of the DFT transform of the mid-channel $m(n)$ in the time domain to obtain mid-channel $M(k)$ in the DFT domain, and stereo decoding and up-mixing of channels $M(k)$ and $S(k)$ into channels $L(k)$ and $R(k)$ in the DFT domain including decoding of the residual signal. The DFT domain analysis and synthesis introduces an OLA delay of 3.125 ms. The synthesis transitions are then handled in the time synchronizer and stereo switching module (see it in Figure 6.3-12).

Upon switching from the DFT stereo frame to the TD stereo frame, the fact that there is only one core-decoder in the DFT stereo decoder makes core-decoding of the TD stereo secondary channel $x(n)$ complicated because the internal states and memories of the second core-decoder of the TD stereo decoder are not continuously maintained (on the contrary, the internal states and memories of the first (primary channel)) core-decoder are continuously maintained using the internal states and memories of the core-decoder of the DFT stereo decoder). The memories of the second core-decoder are thus usually reset in the stereo mode switching updates by the stereo mode switching mechanism. There are however few exceptions where the secondary channel memory is populated with the memory of certain primary channel buffers, for example previous excitation, previous LSF parameters and previous LSP parameters. In any case, the synthesis at the beginning of the first TD stereo secondary channel frame after switching from the DFT

stereo frame to the TD stereo frame consequently suffers from an imperfect reconstruction. Accordingly, while the synthesis from the first core-decoder is well and smoothly decoded during stereo mode switching, the limited-quality synthesis from the second core-decoder introduces discontinuities during the stereo up-mixing and final synthesis. These discontinuities are suppressed by employing the DFT stereo OLA memories during the first TD stereo output synthesis reconstruction as described next.

The stereo mode switching mechanism suppresses possible discontinuities and differences between the DFT stereo and the TD stereo up-mixed channels by a simple equalization of the signal energy. If the ICA gain factor, g_{ICA}, is lower than 1.0, the left channel $l(n)$ after the up-mixing and before the time synchronization is altered in the first TD stereo frame after stereo mode switching using the following relation:

$$l'(n) = \alpha \cdot l(n) \text{ for } n = 0, \dots, L_{eq} - 1 \quad (6.3\text{-}141)$$

where L_{eq} is the length of the signals to equalize which corresponds in the stereo decoder to a 8.75 ms long segment. It corresponds for example to $L_{eq} = 140$ samples at a 16 kHz output stereo signal sampling rate. Then, the value of the gain factor α is obtained using the following relation:

$$\alpha = g_{ICA} + n \cdot \frac{1 - g_{ICA}}{L_{eq}} \text{ for } n = 0, \dots, L_{eq} - 1. \quad (6.3\text{-}142)$$

Further, referring to Figure 6.3-16, the instance A) relates to a missing part of the TD stereo up-mixed synchronized synthesis of the TD stereo frame corresponding to a previous DFT stereo up-mixed synchronization synthesis memory from DFT stereo frame. This memory of length of (3.25 – 1.25) ms is not available when switching from the DFT stereo frame to the TD stereo frame except for its first 0.125 ms long segment.

Figure 6.3-17 is a flow chart illustrating the instance A) of Figure 6.3-16, comprising updating the TD stereo up-mixed synchronization synthesis memory in a first TD stereo frame following switching from the DFT stereo mode to the TD stereo mode, on the decoder side.

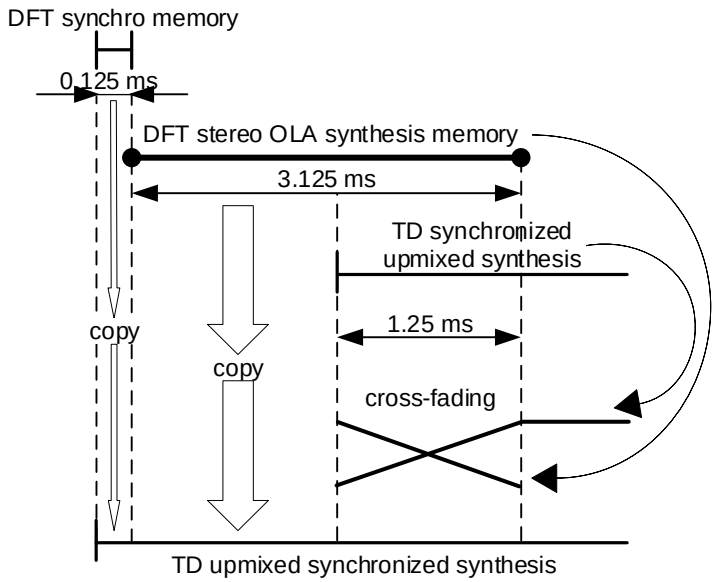


Figure 6.3-17: Flow chart illustrating an instance A) of Figure 6.3-16

Referring to both Figure 6.3-16 and Figure 6.3-17, the stereo mode switching mechanism reconstructs the 3.25 ms of the TD stereo up-mixed synchronized synthesis using the following operations a) to e) for both the left $l(n)$ and right $r(n)$ channels:

a) The DFT stereo OLA synthesis memories are redressed (i.e. the inverse synthesis window is applied to the OLA synthesis memories).

b) The first 0.125 ms part of the TD stereo up-mixed synchronized synthesis is identical to the previous DFT stereo up-mixed synchronization synthesis memory (last 0.125 ms long segment of the previous frame DFT stereo up-mixed synchronization synthesis memory) and is thus reused to form this first part of the TD stereo up-mixed synchronized synthesis.

c) The second part of the TD stereo up-mixed synchronized synthesis having a length of (3.125 – 1.25) ms is approximated with the redressed DFT stereo OLA synthesis memories.

d) The part of the TD stereo up-mixed synchronized synthesis with a length of 2 ms from the previous two steps b) and c) is then populated to the output stereo synthesis in the first TD stereo frame.

e) A smoothing of the transition between the previous DFT stereo OLA synthesis memory and the TD synchronized up-mixed synthesis of the current TD stereo frame is performed at the beginning of the TD stereo synchronized up-mixed synthesis. The transition segment is 1.25 ms long and is obtained using a cross-fading between the redressed DFT stereo OLA synthesis memory and the TD stereo synchronized up-mixed synthesis.

6.3.4.5 Switching between DFT and MDCT stereo modes

6.3.4.5.1 Switching from DFT stereo to MDCT stereo

A mechanism similar to the decoder-side switching from the DFT stereo mode to the TD stereo mode is used in this scenario, where the primary $y(n)$ and secondary $x(n)$ channels of the TD stereo mode are replaced by the left $l(n)$ and right $r(n)$ channels of the MDCT stereo mode.

Since the signal stored in the DFT synthesis memory (see clause 6.3.2.3.9) has the DFT window applied to it, an undressing of the window is done in any MDCT stereo frame following a previous DFT stereo frame. This is done by multiplying the memory with the inverse DFT synthesis window.

Additionally, all required core-coder state variables and buffers are copied over to the second channel in the transition frame.

At the end of the transition frame, a linear cross-fade between the DFT synthesis memory and the MDCT stereo output is performed for both channels.

6.3.4.5.2 Switching from MDCT stereo to DFT stereo

A mechanism similar to the decoder-side switching from the TD stereo mode to the DFT stereo mode is used in this scenario, where the primary $y(n)$ and secondary $x(n)$ channels of the TD stereo mode are replaced by the left $l(n)$ and right $r(n)$ channels of the MDCT stereo mode.

When switching to DFT stereo after an MDCT stereo frame, a passive downmix is applied to the core-coder output memories for both full-band and low-band core-coder output signals. These downmixed memories are used in the core decoding of the single DFT stereo core channel. Furthermore, they are used for a linear cross-fade between the memories and the core-decoder outputs (LB and FB) of the single decoded downmix channel before the DFT stereo upmix. The fade durations are chosen as 2.5 ms for the FB signals and 7.5 ms for the LB signals.

Additionally, the DFT analysis memories (see clause 6.3.2.3.2) for both full-band and low-band are filled by applying a passive downmix to the corresponding MDCT stereo output memories which are being updated during MDCT stereo frames. For the low-band memory this may also require a resampling in case the core sampling rate differs between the current DFT stereo frame and the previous MDCT stereo frame.

6.3.4.6 Switching between TD and MDCT stereo modes

6.3.4.6.1 Switching from TD stereo to MDCT stereo

Switching from the TD stereo mode to the MDCT stereo mode is relatively straightforward because both these stereo modes handle two transport channels and employ two core-decoder instances.

As an opposite-phase down-mixing scheme was employed in the TD stereo encoder (see clause 5.3.4.6.1), the stereo mode switching mechanism similarly alters the TD stereo channel up-mixing at the decoder side to maintain the correct phase of the left and right channels of the stereo sound signal in the last TD stereo frame before the first MDCT stereo frame. Specifically, the stereo mode switching mechanism sets the TS stereo mixing ratio $\beta = 1.0$ and implements an opposite-phase up-mixing (inverse to opposite-phase down-mixing employed in the TD stereo encoder) of the TD stereo primary channel $y(n)$ and TD stereo secondary channel $x(n)$ to calculate the MDCT stereo past left channel $l^{[-1]}(n)$ and the MDCT stereo past right channel $r^{[-1]}(n)$. Consequently, the TD stereo primary channel $y(n)$ is

identical to the MDCT stereo past left channel $l^{[-1]}(n)$ and the TD stereo secondary channel $x(n)$ signal is identical to the MDCT stereo past right channel $r^{[-1]}(n)$.

6.3.4.6.2 Switching from MDCT stereo to TD stereo

Similarly to the switching from the TD stereo mode to the MDCT stereo mode, two transport channels are available and two core-decoder instances are employed in this scenario. In order to maintain the correct phase of the left and right channels of the stereo signal, the TD stereo mixing ratio is set to $\beta = 1.0$ and the opposite-phase up-mixing scheme is used again by the stereo mode switching mechanism in the first TD stereo frame after the last MDCT stereo frame.

6.3.5 DTX operation

6.3.5.1 DTX in Unified stereo

6.3.5.1.1 General

If a Unified stereo CNG frame is received, the decoder generally operates as in DFT-based stereo mode as described in 6.3.2.3. The main difference compared to active frame DFT-based stereo decoding is that the stereo CNG decoder extracts the spectral shape represented by the decoded CNG frame and uses this for stereo CNG generation, as explained in clause 6.3.5.1.2. In addition, there are two differences compared to the EVS CNG decoding, matching the differences as described in clause 5.3.5.1.2.

- Core codec reset in active frame onset
- Energy scaling of CNG

The DFT-based stereo parameters $ITD(m)$, $gIPD(m)$, $g_S[m,b]$ are decoded from the bitstream as described in 0 with the band resolution set by the encoder. In addition, the stereo coherence is decoded as described in 6.3.5.1.4 and a slightly modified DFT-based stereo synthesis is done as described in 6.3.5.1.5.

6.3.5.1.2 Stereo CNG spectral shape extraction

The EVS CNG decoder operates in two modes, LP-CNG and FD-CNG. In Unified stereo CNG, the CNG is generated in DFT domain based on the spectral shape of the decoded EVS CNG frame. For the LP-CNG, the filter LP synthesis filter $\hat{A}(z)$ described in clause 6.7.2.1.4 of [3] is convolved with the denominator of the de-emphasis filter $[1 - \beta_d]$ as described in clause 6.4 of [3], to produce a deemphasized synthesis filter

$$\hat{A}_{demph}(n) = \hat{A}(n) * [1 - \beta_d] \tag{6.3-143}$$

where '$*$' denotes convolution. $\hat{A}_{demph}(n)$ is then transformed to DFT domain where the synthesis shape $1/\hat{A}_{demph}(n)$ is formed according to

$$\begin{aligned} &N_{LP-CNG}(k) = \frac{g_{scale}}{|N_{LPinv}(k)|} \\ &N_{LPinv}(k) = \sum_{n=0}^{L_{FFT}-1} \hat{A}_{demph}(n) e^{-j2\pi kn/N} \\ &g_{scale} = 2\sqrt{\frac{E_{CN}}{0.5L_{FFT}}} \\ &k = 0,1,\ldots,L_{FFT}-1 \end{aligned} \tag{6.3-144}$$

where E_{CN} is a low-pass filtered energy of the low-band excitation signal defined in clause 6.8.4 of [3] and L_{FFT} is the FFT length. For a core coding of $F_s = 12.8$ kHz, $L_{FFT} = 256$ and for $F_s = 16$ kHz, $L_{FFT} = 320$. For the high band (up to 14 kHz for $F_s = 12.8$ kHz and up to 16 kHz for $F_s = 16$ kHz) a corresponding shaping spectrum $N_{LP-CNG}^{SHB}(k)$ is obtained as the inverse magnitude DFT spectrum of the LP-filter for SHB-CNG obtained as defined in clause 6.7.2.1.7 of [3].

For TD-based stereo, the CNG is still utilizing the DFT-based stereo mode. For a smooth transition from active TD-based stereo coding to CNG, background noise parameters estimated for the active coding are adapted and combined with background noise parameters from the SID frame. The correlation between the decoded left and right stereo signals of the TD-based stereo mode is estimated according to

$$C_{LR\_TD} = \frac{dot_{LR}}{\sqrt{enr_L\, enr_R}} \tag{6.3-145}$$

where $dot_{LR} = \sum_n L'(n)\, R'(n)$, and $enr_L = \sum_n L'(n)^2$ and $enr_R = \sum_n R'(n)^2$ are energies of the output signals from the previous active frame. A low pass filtering of the inter-channel correlation is performed over frames according to

$$\hat{C}_{LR\_TD}[m] = \alpha_{corr} C_{LR\_TD}[m] + (1 - \alpha_{corr})\hat{C}_{LR\_TD}[m-1] \tag{6.3-146}$$

with $\alpha_{corr} = 0.8$.

For the first SID after active coding, coherence values for the frequency bands b are obtained differently based on whether active frame coding is done in TD-based stereo mode or DFT-based stereo mode, according to

$$\hat{C}_{band}^{CNG}[m,b] = \begin{cases} \hat{C}_{LR\_TD}[m]\ \hat{C}_{LR\_TD}[m], & if\ firstSidTD = 1 \\ \hat{C}_{band}[m,b], & if\ firstSidTD = 0 \end{cases} \tag{6.3-147}$$

where $\hat{C}_{band}[m,b]$ is defined by equation (6.3-175) and $firstSidTD$ is a flag indicating that the last active frame was running the TD-based stereo mode.

For subsequent inactive the frames the coherence is updated as follows

$$\hat{C}_{band}^{CNG}[m,b] = \begin{cases} \hat{C}_{LR\_TD}\ \hat{C}_{LR\_TD}, & if\ i_{tdCntr} > 8\ \wedge i_{dftCntr} < 50 \wedge i_{sidCntr} < 6 \\ (1-\alpha_{coh})\hat{C}_{band}[m,b] + \alpha_{coh}\hat{C}_{band}^{CNG}[m-1,b], & otherwise \end{cases} \tag{6.3-148}$$

where i_{tdCntr} is the number of TD frames for which $\hat{C}_{LR\_TD}[m]$ has been estimated, $i_{dftCntr}$ is the number of DFT-based stereo frames, $i_{sidCntr}$ is the number of SID frames received by the decoder, and α_{coh} is set to 0.8. $\hat{C}_{LR\_TD}$ is the latest value of $\hat{C}_{LR\_TD}[m]$ estimated during TD-based stereo coding.

For the transition from active TD stereo coding to CNG a crossfade is performed between two noise spectra, one being the background noise parameters representing frames of the active TD stereo coding mode, estimated at the decoder, and the other one based on parameters provided in the SID.

For LP CNG the crossfade is based on background noise parameters estimated in active TD frames, here denoted an $N_{CNA}^{lastActive}(k)$ for the latest active frame. A crossfade length is determined in the first SID after TD coding according to

$$L_{xfade} = \begin{cases} -M_{xfade} r_1 + M_{xfade}, & if\ r_1 < 1 \\ -M_{xfade}\left(\frac{1}{r_1}\right) + M_{xfade}, & otherwise \end{cases} \tag{6.3-149}$$

where M_{xfade} is the maximum crossfade or transition length allowed and r_1 is the energy ratio of the two background estimates determined by

$$r_1 = \sqrt{\frac{\sum_{k=k_0}^{k=k_{N-1}} N_{CNA}^{lastActive}(k)}{\sum_{k=k_0}^{k=k_{N-1}} N_{LP-CNG}(k)}} \tag{6.3-150}$$

Over a transition period L_{xfade}, comfort noise parameters $\widehat{N}_{LP-CNG}(k)$ are generated as the weighted average of the two noise spectra over a transition period according to

$$\widehat{N}_{LP-CNG}(k) = \begin{cases} \left(1 - \frac{i_{inactive}}{L_{xfade}}\right)\left(\alpha_{avg} N_{CNA}^{lastActive}(k)\right) + \frac{i_{inactive}}{L_{xfade}} N_{LP-CNG}(k), & i_{inactive} < L_{xfade} \\ N_{LP-CNG}(k), otherwise \end{cases} \tag{6.3-151}$$

where $i_{inactive}$ is the number of inactive frames and α_{avg} is a compensation factor used to scale the background noise parameters $N_{CNA}^{lastActive}(k)$ of TD stereo active coding mode primary (downmix) signal such that its energy corresponds to the energy of a corresponding downmix signal of the CNG coding mode (which is based on the DFT-based stereo coding mode). The scaling parameter α_{avg} is computed by summing scaling factors $\alpha[b]$ over all bands and dividing by the number of bands, where

$$\alpha[b] = \frac{1}{2}\sqrt{\frac{1 + c + 2\sqrt{c \cdot C_{LR\_TD}}}{c \cdot \beta^2 + (1-\beta)^2 s_{right}^2 + 2\beta(1-\beta) s_{right}\sqrt{c \cdot C_{LR\_TD}}}} \tag{6.3-152}$$

with β being the TD-based stereo mixing ratio controlling the downmixing, see clause 6.3.2.2, and s_{right} a target gain applied to the right channel during upmixing, see ICA encoder target gain in clause 5.3.4.2. c is given by

$$c = \frac{\left(1 + \hat{g}_{S,LP}\right)^2 + \gamma^2}{\left(1 - \hat{g}_{S,LP}\right)^2 + \gamma^2} \tag{6.3-153}$$

where $\hat{g}_{S,LP}$ is the side gain parameter as described in clause 6.3.5.1.3.

For low band LP CNG, a random noise generator is used to generate noise for the real and imaginary parts. The random noise is scaled with $\widehat{N}_{LP-CNG}(k)\frac{N_{output}}{2}$, where N_{output} is the length of the synthesis frame.

For high band LP CNG, denoted as $N_{LP-CNG}^{SHB}(k)$, a scale factor for the high band CNG is computed as

$$scale_{target} = \frac{\bar{\bar{E}}_{hs,i}}{\sum_k N_{LP-CNG}^{SHB}(k)^2} \tag{6.3-154}$$

where the synthesis gain $\bar{\bar{E}}_{hs,i}$ is defined by equation (1971) in [3]. A random noise generator is used to generate noise for the real and imaginary parts. The random noise is scaled with a flipped spectrum of $\widehat{N}_{LP-CNG}(k)$ and $scale_{target}\frac{N_{output}}{2}$.

Two uncorrelated noise spectra $N_{CNG-0}(k)$ and $N_{CNG-1}(k)$ in the case of LP CNG are accordingly generated as

$$N_{CNG-0}(k) = \begin{cases} randNoise_0(k)\ \frac{\widehat{N}_{LP-CNG}(k)\ N_{output}}{2}, \text{for lowband} \\ randNoise_0(k)\ \frac{flipped(\widehat{N}_{LP-CNG}(k))\ N_{output}}{2} scale_{target}, \text{ for highband} \end{cases} \tag{6.3-155}$$

$$N_{CNG-1}(k) = \begin{cases} randNoise_1(k)\ \frac{\widehat{N}_{LP-CNG}(k)\ N_{output}}{2}, \text{for lowband} \\ randNoise_1(k)\ \frac{flipped(\widehat{N}_{LP-CNG}(k))\ N_{output}}{2} scale_{target}, \text{ for highband} \end{cases} \tag{6.3-156}$$

where $randNoise_0(k)$ and $randNoise_1(k)$ are two random gaussian noises generated with different seeds.

For FD CNG the background noise estimate at the decoder is updated using in a similar way as in clause 6.7.3.2.3.2 in [3] by combining SID and shaping parameters estimated at the decoder, with the modification provided in equation (6.3-157). The modifications provide a smooth transition when switching from TD active mode to FD CNG by applying a crossfade over a fixed transition length M_{xfade} between the adapted background noise parameters $\widehat{N}_1$ based on the background noise parameters estimated during active TD-based stereo and the background noise parameters N_2 received in the SID frame, as described below. A full resolution CNG spectrum is obtained as

$$\bar{N}_{FD-CNG}^{full}(k) = r_2(b)\widehat{N}_1(k) \tag{6.3-157}$$

where b is a frequency sub-band index and $r_2(b)$ is determined by

$$r_2(b) = \begin{cases} \min\left(1 + \frac{1}{M_{xfade}}(r_0(b) - 1)i_{inactive}, r_0(b)\right), & if\ i_{inactive} < M_{xfade} \\ r_0(b), & otherwise \end{cases}. \tag{6.3-158}$$

$r_0(b)$ is the ratio of two noise spectra determined by

$$r_0(b) = \frac{N_2(b)}{\widehat{N}_1(b)} = \frac{N_2(b)}{\alpha_{avg}N_1(b)} = \frac{N_{FD-CNG}^{[SID]}(b)}{\alpha_{avg}N_{FD-CNG}^{[shaping,LR]}(b)} \tag{6.3-159}$$

with $N_{FD-CNG}^{[SID]}(b)$, $N_{FD-CNG}^{[shaping,LR]}(b)$ and $N_{FD-CNG}^{[shaping,FR]}(k)$ as defined in clause 6.7.3.2.3.2 in [3].

$\widehat{N}_1(k)$ is equivalent to

$$\widehat{N}_1(k) = \alpha_{avg}N_1(k) = \alpha_{avg}N_{FD-CNG}^{[shaping,FR]}(k) \tag{6.3-160}$$

For the first part of the CNG spectrum, the frequency resolution of the noise shaping function $N_{FD-CNG}^{full}(k)$ is twice the DFT-based stereo resolution since DFT-based stereo uses two subframes for each frame. The CNG spectrum $N_{FD-CNG}(k)$ is formed by averaging the bins two by two according to

$$\widehat{N}_{FD-CNG}(k) = \frac{\bar{N}^{full}_{FD-CNG}(2k) + \bar{N}^{full}_{FD-CNG}(2k+1)}{2}$$
$$k = 0,1, \dots, \frac{L_{FFT}}{2} - 1 \quad (6.3\text{-}161)$$

For higher frequency coefficients, $\widehat{N}_{FD-CNG}(k) = \bar{N}^{full}_{FD-CNG}(k)$. Two uncorrelated noise spectra $N_{CNG-0}(k)$ and $N_{CNG-1}(k)$ are generated as in clause 6.7.3.3.2 of [3] based on pseudo-random Gaussian noise with different random seeds, scaled with $\sqrt{\widehat{N}_{FD-CNG}(k)}\frac{N_{output}}{2}$, where N_{output} is the length of the synthesis frame. Accordingly

$$N_{CNG-0}(k) = randNoise_0(k)\sqrt{\widehat{N}_{FD-CNG}(k)}\frac{N_{output}}{2} \quad (6.3\text{-}162)$$

$$N_{CNG-1}(k) = randNoise_1(k)\sqrt{\widehat{N}_{FD-CNG}(k)}\frac{N_{output}}{2} \quad (6.3\text{-}163)$$

where $randNoise_0(k)$ and $randNoise_1(k)$ are two random gaussian noises generated with different seeds.

6.3.5.1.3 Stereo CNG side gain, ITD and IPD decoding

The side gain $\hat{g}_S[m,b]$ is decoded the same way as in active frames, but with the band resolution as described in Table 5.3-22. The decoded side gain parameters are low-pass filtered during CNG frames according to

$$\hat{g}_{S,LP}[m,b] = 0.2\hat{g}_S[m,b] + 0.8\hat{g}_{S,LP}[m-1,b] \quad (6.3\text{-}164)$$

except for the first CNG frame after active coding where $\hat{g}_{S,LP}[m,b]$ is directly set to $\hat{g}_S[m,b]$.

The ITD parameter is decoded similar to the active frames but without the option of Huffman coding and included an extra step due to the reduced resolution.

$$ITD_{SID}(m) = 2I_{ITD} + 256 \cdot sign \quad (6.3\text{-}165)$$

where I_{ITD} is the received ITD index and $sign \in \{-1,1\}$ is the decoded sign bit.

Whether residual encoding is enabled or not during the active encoding mode indicates whether the foreground and background signals are efficiently separated for active frames, and if it is enabled, $ITD_{SID}(m)$ is used directly for synthesis in the CNG encoding mode, i.e. $ITD_{syn}(m) = ITD_{SID}(m)$. However, for CNG stereo synthesis where the active frames are encoded at a bitrate $\leq 24.4\ kbps$, where residual coding is not utilized (indicating the foreground and background signals are not efficiently separated for active frames), the ITD is adjusted by a gradual fading from the ITD of the previous frame towards the received ITD target, $ITD_{target} = ITD_{SID}(m)$, as

$$ITD_{syn}(m) = ITD_{syn}(m-1) + ITD_{step} \quad if \quad itd\_xfade\_counter < L_{xfade} \quad (6.3\text{-}166)$$

where $itd\_xfade\_counter$ is a counter of number of frames for which the fade has been performed and $L_{xfade} = 100$ is the total length of the fade, unless interrupted by active frames. One exception is for SID frames following active segments of at most N_(xfade_reset)=2 active frames, which instead are handled according to equation (6.3-168). $ITD_{syn}(m-1)$ denotes the ITD of the previous frame, being the latest ITD value of the fading and starting from the ITD of the last active frame prior the CNG period. The size of the steps taken towards the target ITD is set in the beginning of the CNG period, and updated whenever a new ITD_{target} is received, according to

$$ITD_{step} = \frac{ITD_{target} - ITD_{syn}(m-1)}{L_{xfade} - itd\_xfade\_counter} \quad (6.3\text{-}167)$$

The fading counter $itd\_xfade\_counter$ is increased by one for each frame the fade is being performed and reset to zero when there has been at more than $N_{xfade\_reset} = 2$ active frames. Following segments of at most N_{xfade_reset} active frames, the counter is not reset and the ITD fade is resumed from the ITD of the previous CNG frame instead of being restarted from the last active frame ITD, i.e.

$$ITD_{syn}(m) = ITD_{prev} + ITD_{step} \quad if \quad itd\_xfade\_counter < L_{xfade} \quad (6.3\text{-}168)$$

where ITD_{prev} is the latest ITD value of the gradual fade from the previous CNG period. If a new ITD target is received, the step size is updated as

$$ITD_{step} = \frac{ITD_{target} - ITD_{prev}}{L_{xfade} - itd\_xfade\_counter} \quad (6.3\text{-}169)$$

The IPD is decoded according to

$$gIPD_{SID}(m) = \frac{I_{IPD}\pi}{2} - \pi \quad (6.3\text{-}170)$$

where I_{IPD} is the decoded IPD index. As for the ITD, if residual coding is enabled for the active frames, $gIPD_{SID}(m)$ is used directly for stereo CNG synthesis, $gIPD_{syn}(m) = gIPD_{SID}(m)$. However, for active frame bitrates ≤ 24.4 $kbps$ where residual coding is not utilized, the IPD is adjusted by a gradual fading towards $gIPD_{target} = gIPD_{SID}(m)$, as

$$gIPD_{syn}(m) = gIPD_{syn}(m-1) + gIPD_{step} \quad if \quad ipd\_xfade\_counter < L_{xfade} \quad (6.3\text{-}171)$$

where $ipd\_xfade\_counter$ is a counter of number of frames for which the fade has been performed and $L_{xfade} = 100$ is an upper threshold for the number of fading frames. $gIPD_{syn}(m-1)$ denotes the IPD of the previous frame, being the latest IPD value of the fading and starting from the IPD of the last active frame prior the CNG period. The size of the steps taken towards the target IPD is set in the beginning of the CNG period, and updated whenever a new $gIPD_{target}$ is received, according to

$$gIPD_{step} = \frac{gIPD_{target} - gIPD_{syn}(m-1)}{L_{xfade} - ipd\_xfade\_counter} \quad (6.3\text{-}172)$$

The fading counter $ipd\_xfade\_counter$ is increased by one for each frame the fade is being performed and reset to zero when there has been at more than $N_{xfade\_reset} = 2$ active frames. Following segments of at most N_{xfade_reset} active frames, the counter is not reset and the IPD fade is resumed from the IPD of the previous CNG frame instead of being restarted from the last active frame IPD, i.e.

$$gIPD_{syn}(m) = gIPD_{prev} + gIPD_{step} \quad if \quad ipd\_xfade\_counter < L_{xfade} \quad (6.3\text{-}173)$$

where $gIPD_{prev}$ is the latest IPD value of the gradual fade from the previous CNG period. If a new IPD target is received, the step size is updated as

$$gIPD_{step} = \frac{gIPD_{target} - gIPD_{prev}}{L_{xfade} - ipd\_xfade\_counter} \quad (6.3\text{-}174)$$

6.3.5.1.4 Stereo CNG coherence decoding

The intra-frame predictor index q is obtained from the bitstream and the intra-frame predictor $p^{(q)}[b, i]$ is selected. Based on the available bit budget for the encoded stereo coherence B_C for the current frame m, the weighting factor α is obtained according to Table 5.3-23, where the decoded bit now indicates whether to select α_{low} or α_{high}. The coherence for each band b is obtained according to

$$\hat{C}_{band}[m, b] = \alpha \hat{C}^{(q)}_{intra}[m, b] + (1-\alpha)\hat{C}_{band}[m-1, b] + \hat{C}_{res}[m, b] \quad (6.3\text{-}175)$$

where the coherence prediction residual $\hat{C}_{res}[m, b]$ is now obtained from the bitstream and decoded according to Table 5.3-24. The intra-frame prediction $\hat{C}^{(q)}_{intra}[m, b]$ and the inter-frame prediction $\hat{C}_{band}[m-1, b]$ are obtained in the same way as in (5.3-319), using the previously reconstructed coherence values. In case the reconstructed values fall outside of the valid range $[0,1]$, the value is clamped to this range according to

$$\hat{C}_{band}[m, b] := \begin{cases} 0, & \hat{C}_{band}[m, b] < 0 \\ 1, & \hat{C}_{band}[m, b] > 1 \\ \hat{C}_{band}[m, b], & otherwise \end{cases} \quad (6.3\text{-}176)$$

The coherence $\hat{C}_{band}[m, b]$ is used together with the remaining decoded stereo parameters and the decoded CNG down-mix signal to produce a stereo CNG synthesis as described in clause 6.3.5.1.5.

6.3.5.1.5 Stereo CNG synthesis

The stereo CNG synthesis is done in frequency domain based on the stereo parameters, as described in clause 6.3.5.1.3 here omitting the frame index m, and the uncorrelated noise spectra $N_{CNG-0}(k)$ and $N_{CNG-1}(k)$, as generated in clause

6.3.5.1.2. Compared to the active frame stereo upmix, see 6.3.2.3.7, the upmix is done for a coarser frequency resolution as described in Table 5.3-22, and with the same stereo parameters for both synthesis subframes i, as

$$\hat{L}_i[k] = N_{CNG-0}(k)\left(1 + \hat{g}_{S,LP}[b]\right) + \gamma[b]\, N_{CNG-1}(k) \quad (6.3\text{-}177)$$

and

$$\hat{R}_i[k] = N_{CNG-0}(k)\left(1 - \hat{g}_{S,LP}[b]\right) - \gamma[b]\, N_{CNG-1}(k) \quad (6.3\text{-}178)$$

for k ∈ I_b, where I_b is the set of coefficients k belonging to frequency band b and $\gamma[b]$ is based on the coherence as

$$\gamma[b] = \begin{cases} \sqrt{\frac{\hat{C}_{band}[b]}{1-\hat{C}_{band}[b]} + 1 - \hat{g}_{S,LP}^2[b]} - \sqrt{\frac{\hat{C}_{band}[b]}{1-\hat{C}_{band}[b]}} & if \quad \hat{C}_{band}[b] < 0.9 \\ 0 & otherwise \end{cases} \quad (6.3\text{-}179)$$

The channels are then rotated for reinjecting the global IPD:

$$\begin{cases} \hat{L}_i[k] = \hat{L}_i[k] e^{j\, g\, IPD_{syn}} \\ \hat{R}_i[k] = \hat{R}_i[k] \end{cases} \quad (6.3\text{-}180)$$

The left and right channel waveforms are then generated by applying STFT synthesis to the reconstructed left and right spectra as specified in clause 6.3.2.3.9. Finally, the ITD between the channels is synthesized as described in clause 6.3.2.1 using ITD_{syn} as obtained in clause 6.3.5.1.3.

6.3.5.2 DTX in MDCT-based stereo

6.3.5.2.1 General overview

In MDCT-based stereo, the SID payload consists of parametric noise data for the two input channels in a mid/side representation, a coherence value indicating how correlated the background noise seen at the encoder is and a one-bit value indicating whether the energy of the side representation of the noise data is lower than a threshold (the "no-side flag").

6.3.5.2.2 Decoding of parametric noise data

Decoding of the noise data follows the MSVQ decoding approach described in clause 5.3.5.2.4. For decoding the mid noise data, all 6 stages of the MSVQ are used while for the side noise data only the first four are used. The no-side flag is represented by a single bit and thus read directly from the bitstream. If the value of the no-side flag is 1, the mid noise data vector is set to zero. Then, the MSVQ-decoded parametric noise data is reconverted from a mid/side representation to a left/right representation. Thus, the reconstructed parametric noise data for each channel is calculated as

$$N_{L,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 1, \\ N_{M,FD-CNG}^{SID}(i) + N_{S,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 0. \end{cases} \quad (6.3\text{-}181)$$

$$N_{R,FD-CNG}^{SID}(i) = \begin{cases} N_{M,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 1, \\ N_{M,FD-CNG}^{SID}(i) - N_{S,FD-CNG}^{SID}(i) \; if \; no-side \; is \; 0. \end{cases} \quad (6.3\text{-}182)$$

Before generating the actual stereo CNG signal, processing steps described in clauses 6.7.3.1.2 and 6.7.3.1.3 of [3] are applied separately on the noise data for each channel.

6.3.5.2.3 Stereo CNG

The two channels in the CNG output signal of high-rate stereo are generated using a stereo signal generator which consists of three uncorrelated gaussian noise sources and a mixer to mix the generated noise signals controlled by the coherence value decoded from the last SID.

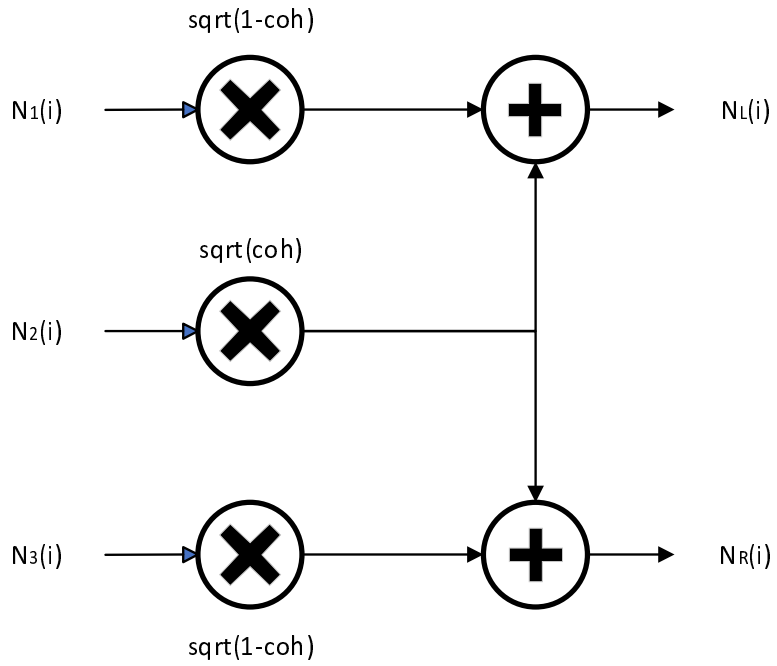


Figure 6.3-18: Stereo noise signal generator for CNG in MDCT-based Stereo DTX

The comfort noise generation process is illustrated in Figure 6.3-18. For each of the two channels, a separate gaussian noise source generates a separate noise signal ($N_1(i)$ or $N_3(i)$, respectively, the "channel noise signals") with both noise signals being decorrelated between each other. A third gaussian noise source (the "mixing noise source", $N_2(i)$) generates a third noise signal that is also decorrelated from each of the two other noise signals. The two channel noise signals are then mixed with the mixing noise signal, i.e., the mixing noise signal is added to both channel noise signals after weighting all of the signals. The coherence value decoded from the last received SID frame serves as a control parameter for the weighting. With $N_l(i)$ being the noise signal for the left channel, $N_r(i)$ the noise signal for the right channel and *coh* being the coherence value, mixing of the noise signals is done like so:

$$\mathrm{N_L(i)} = \sqrt{1-\mathrm{coh}} \times \mathrm{N_1(i)} + \sqrt{\mathrm{coh}} \times \mathrm{N_2(i)} \quad (6.3\text{-}183)$$

$$\mathrm{N_R(i)} = \sqrt{1-\mathrm{coh}} \times \mathrm{N_3(i)} + \sqrt{\mathrm{coh}} \times \mathrm{N_2(i)} \quad (6.3\text{-}184)$$

A higher coherence between the channels thus leads to more correlated noise being generated in the channels, while a lower coherence value leads to a higher amount of uncorrelated noise in the stereo output. This way, the inter-channel coherence in the comfort noise signal follows the coherence of the background noise seen in the encoder and a similar spatial expression is achieved.

Next, both channel noise signals are spectrally shaped by employing FD-CNG as described in clause 6.7.3.3 of [3]. This is done separately on both channels.

6.3.5.2.4 Smoothing of transitions from DTX frame to active TCX

Especially for very low-frequency background noises with high spectral tilt such as car noise, there can be transition artifacts at the border between an inactive frame and an active frame coded with TCX. To mitigate these artifacts, a mean filter is applied on the output signal around the transition borders by crossfading between the unprocessed output signal and the mean-filtered output signal. This way, the maximum smoothing effect is applied on the transition part only and no adjacent signal parts are affected by the lowpass effect of the mean filter. The smoothing is applied before resetting the CLDFB memories in the course of switching from ACELP core (used for CNG) to the MDCT core.

First, the unprocessed output signal s is filtered using a mean filter:

$$s_{filtered}(i) = \frac{1}{w}\sum_{j=i-\left\lfloor\frac{L}{2}\right\rfloor}^{i+\left\lfloor\frac{L}{2}\right\rfloor} s(j),\ i = 0, \ldots, 2d \quad (6.3\text{-}185)$$

where *d* is the delay compensation length in samples and $s(j) = s(0), if\ j < 0$. L denotes the length of the filter memory. It depends on the output sampling rate and is 15 for 16 kHz output sampling rate, 31 for 32 kHz output sampling rate and 47 for 48 kHz output sampling rate. This filtered signal is then crossfaded with the unprocessed signal so that the filtered signal is faded in completely at the frame transition border and faded out again afterwards:

$$s_{smooth}(i) = fade(i)\, s_{filtered}(i) + \big(1 - fade(i)\big)\, s(i), i = 0, \dots, 2d \quad (6.3\text{-}186)$$

The crossfade function is defined as:

$$fade(i) = \begin{cases} \frac{i}{d}, for\ 0 < i \leq d \\ 2 - \frac{i}{d}, for\ d < i \leq 2d \end{cases} \quad (6.3\text{-}187)$$

6.3.6 Stereo output format conversion

6.3.6.1 Overview

The stereo format supports the following output channel configurations: mono, stereo, multichannel. The output to stereo does not require any further processing while a conversion to other output configurations is described in following clauses.

6.3.6.2 Stereo-to-Mono output

6.3.6.2.1 DFT stereo mono output

6.3.6.2.1.1 13.2 – 24.4 kbps: direct output of decoded core signal

In most cases, outputting a mono signal is trivial in DFT Stereo. Since it is a parametric approach to stereo that relies on a single downmix channel and parametric side information for upmixing back to stereo output, a suitable mono signal is already available after the core-decoder. For DFT Stereo bitrates ≤ 24.4 kbps, this mono downmix signal was generated at the encoder entirely via an active downmixing scheme (as described in 5.3.2.4) and can be used as mono output directly, bypassing the need for any stereo upmix processing in DFT domain.

6.3.6.2.1.2 32 kbps: conversion of residual downmix to active downmix

For the bitrate of 32 kbps, the DFT Stereo processing is not entirely parametric, anymore. While the upper part of the downmix spectrum (above 1 kHz) is still generated with the aforementioned active downmixing scheme, the lower part of the spectrum is downmixed via an M/S transform. The mid-signal of this transform becomes the downmix signal in this frequency range while the side (or residual) signal of the M/S downmix is separately coded and also transmitted in the bitstream. This is described at length in 5.3.2.4.12.

This means that for 32 kbps a mono signal suitable for direct decoder output is not available, as the transmitted core signal was generated by 2 different downmixing schemes. Particularly, the mid-signal used in the lower part of the spectrum does not always give the best quality for playback compared to the active downmix of the upper part of the spectrum. Therefore, it is desirable to harmonize all parts of the spectrum of the decoded core signal by also converting the lower part to an active downmix. The necessary decoder processing for this conversion is illustrated in Figure 6.3-19.

Unlike the lower bitrates – DFT domain processing cannot be skipped at 32 kbps. Instead, both the decoded core signal and the decoded residual signal (see clause 6.3.2.3.4) have to be converted to DFT domain. There, the part of the spectrum that uses the residual signal is upmixed in the same way as for regular stereo output, as described in clause 6.3.2.3.7. After obtaining the stereo representation, the according frequency bands can be downmixed again using the active downmixing scheme (the same way as it is done at the encoder for the higher bands, see again clause 5.3.2.4). This results in a final harmonized downmix signal where the combined lower and upper parts of the spectrum are based on the same downmixing scheme. This signal is now transformed back to time-domain via inverse DFT and rendered as mono output.

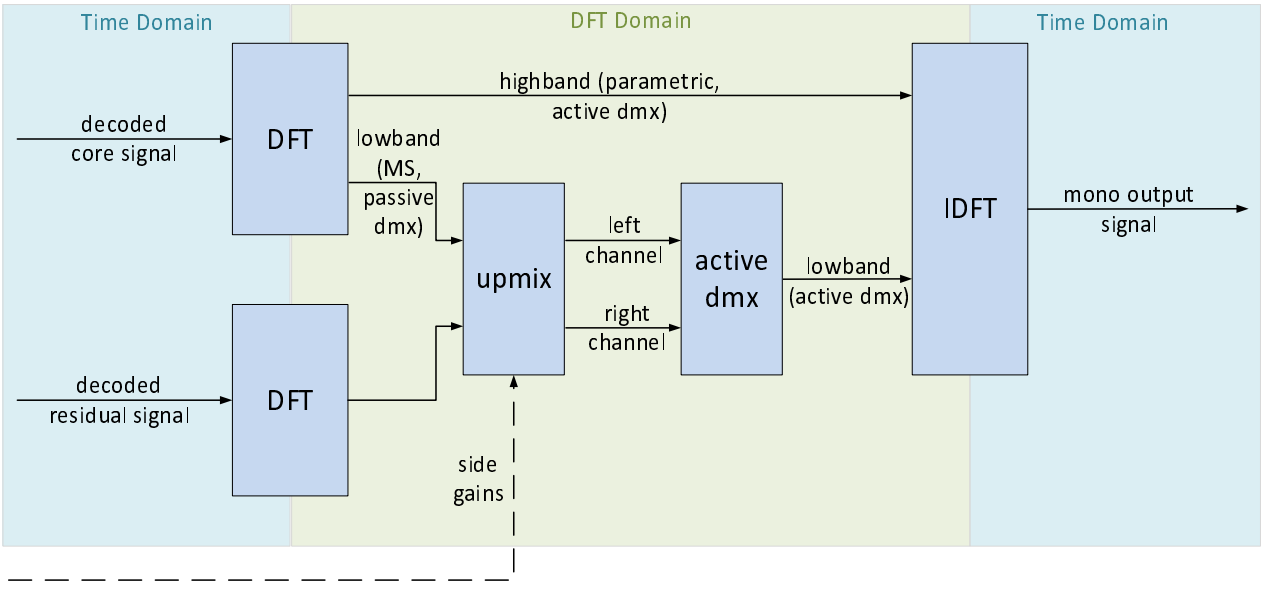


Figure 6.3-19: Mono output at 32 kbps DFT Stereo

6.3.6.2.2 TD stereo mono output

For generating mono audio output in the TD stereo mode, the stereo decoding as described in clause 6.3.2.2 is followed while several particularities are taken into account:

- YX up-mixing scheme is exclusively used to avoid issues with near-out-of-phase signals
- Only the ICA target gain is decoded in the ICA decoder
- The up-mixed right channel of the TD stereo audio signals is scaled up using the decoded ICA target gain value
- A passive stereo to mono downmix is performed to convert left and right TD stereo output channels to a mono output audio using equation (5.3-58).

6.3.6.2.3 MDCT stereo mono output

6.3.6.2.3.1 Introduction

For generating mono output from an MDCT Stereo bitstream, a dedicated downmixing scheme is employed towards the end of the MDCT Stereo decoding process (see clause 6.3.3). To benefit the quality of the mono output and to avoid the issues coming with a simple passive downmix, the downmix is done in an active manner by estimating and applying adaptive bandwise weights for the decoded stereo channels in MDCT domain. The actual downmix to a single mono channel is done later after transforming back to time-domain since direct downmixing in spectral domain cannot be done if the time resolution between the channels is different (TCX20 vs TCX5/10). The general scheme of this hybrid approach is illustrated in Figure 6.3-20.

6.3.6.2.3.2 Computation of downmix weights

The first step of this downmixing process (estimating and applying the downmix weights) happens after all the core-decoder and stereo decoder processing is done and directly before the IMDCT back transform. The weights are computed based on a given target energy corresponding to the energy of the phase-rotated mid-channel (in a manner similar to the one employed in the downmix of the DFT-based stereo encoder (as described in 5.3.2.4.10):

$$E_{target} = \left|\frac{L+Re^{-j\varphi}}{2}\right|^2 = \frac{\langle L,L\rangle+\langle R,R\rangle+2|\langle L,R\rangle|}{4} = \frac{|L|^2+|R|^2+2|\langle L,R\rangle|}{4}, \quad (6.3\text{-}188)$$

where L and R represent the left and right channel spectral magnitudes. Based on this target energy the channel weights can be computed for each spectral band as follows:

$$w_R = \frac{1}{2\sqrt{2}} \frac{\sqrt{|L|^2+|R|^2+2|\langle L,R\rangle|}}{|L|+|R|} \quad (6.3\text{-}189)$$

and

$$w_L = w_R + 1 - \frac{|L+R|}{|L|+|R|}. \quad (6.3\text{-}190)$$

These weights or band-wise weighting values w_R and w_L are computed per spectral band with each band encompassing several MDCT bins where the size of the bands increases towards higher frequencies. Concretely, the same bitrate-dependent band configurations as in the regular MDCT Stereo En-/Decoding (see Table 5.3-17) are used also for the mono downmixing.

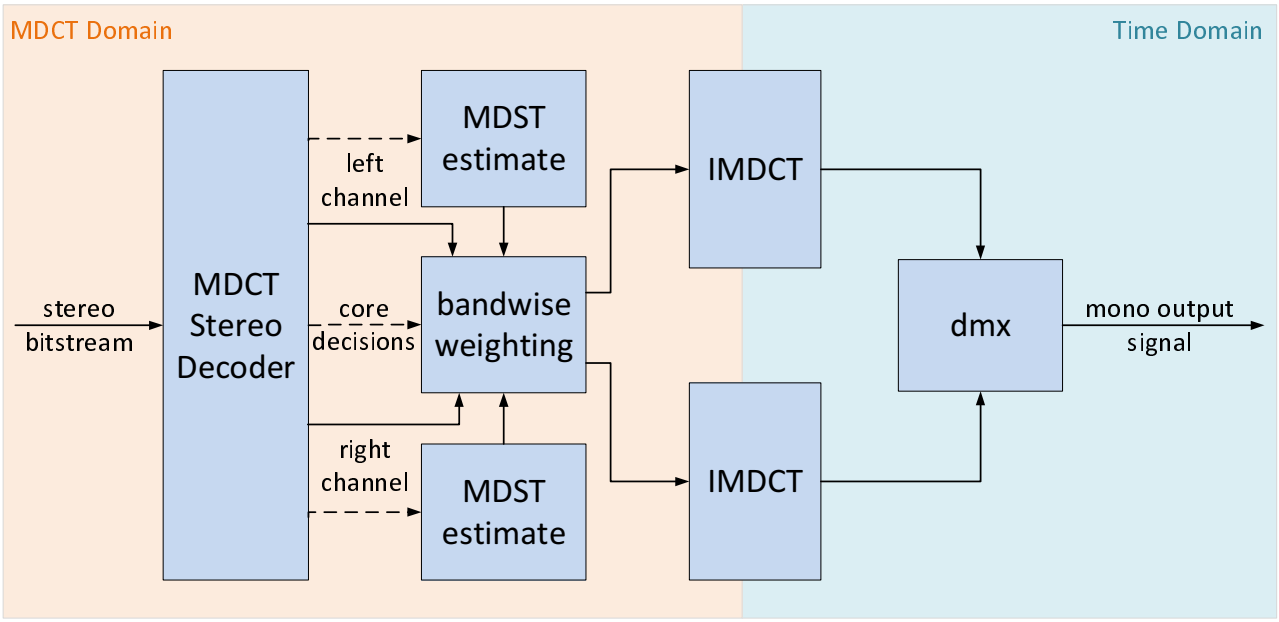


Figure 6.3-20: Mono output for MDCT Stereo

As the transmitted MDCT coefficients are only real-valued, the complementary MDST values that are required for energy-preserving weighting are obtained for each channel by the estimate

$$MDST_i = MDCT_{i+1} - MDCT_{i-1}, \quad (6.3\text{-}191)$$

where i specifies the spectral bin number.

Using this estimate |L| and |R| are computed for each band i as

$$|L| = \sqrt{\sum_{i\ in\ b}(MDCT_{i,l}{}^2 + MDST_{i,l}{}^2)}\ ,\ |R| = \sqrt{\sum_{i\ in\ b}(MDCT_{i,r}{}^2 + MDST_{i,r}{}^2)} \quad (6.3\text{-}192)$$

$|L+R|$ is computed as

$$|L+R| = \sqrt{|L|^2+|R|^2+2\left(\sum_{i\ in\ b}(MDCT_{i,l}MDCT_{i,r}+MDST_{i,l}MDST_{i,r})\right)^2} \quad (6.3\text{-}193)$$

and $|\langle L,R\rangle|$ is computed as the magnitude or absolute value of the complex dot product

$$|\langle L,R\rangle| = \sqrt{\left(\sum_{i\ in\ b}(MDCT_{i,l}MDCT_{i,r}+MDST_{i,l}MDST_{i,r})\right)^2+\left(\sum_{i\ in\ b}(MDST_{i,l}MDCT_{i,r}-MDCT_{i,l}MDST_{i,r})\right)^2} \quad (6.3\text{-}194)$$

where i specifies the bin number inside spectral band b.

6.3.6.2.3.3 Weighting of stereo channels

The calculated weights $w_{L,b}$ and $w_{R,b}$ are then applied to every MDCT bin *i* inside their respective band *b*, thereby creating weighted MDCT spectra of the stereo channels:

$$MDCT_{i,l,weighted} = w_{L,b} MDCT_{i,l} \quad (6.3\text{-}195)$$

and

$$MDCT_{i,r,weighted} = w_{R,b} MDCT_{i,r} \quad (6.3\text{-}196)$$

6.3.6.2.3.4 Time-domain downmix

In a second step, the two weighted channels *L* and *R* are then transformed back to time domain via IMDCT and downmixed to a single downmix channel *D* by simple summing and scaling of each time-domain sample *i* of *L* and *R*:

$$D_i = \frac{1}{\sqrt{2}}(L_i + R_i) \quad (6.3\text{-}197)$$

6.3.6.2.3.5 Conversion of MDCT coefficients to higher resolution

For the special case of different cores in the two channels the computation of the cross-spectra correlation as part of the weight calculation has to be slightly adapted. Due to the different frequency and time resolutions of TCX20 and TCX10, directly calculating the dot product between left and right is not possible. In order to make it possible, the spectrum of the (sub)frame with the lower spectral resolution is converted into an approximation of a spectrum with twice the spectral resolution by computing:

$$MDCT_{2i,k0} = \frac{1}{2}\left(MDCT_{i,k1} + (-1)^i MDCT_{i,k0}\right) \quad (6.3\text{-}198)$$

and

$$MDCT_{2i+1,k0} = \frac{1}{2}\left(MDCT_{i,k1} - (-1)^i MDCT_{i,k0}\right) \quad (6.3\text{-}199)$$

where i specifies the spectral bin number and k0 and k1 the subframes with lower resolution. These additions and subtractions can be seen as high- and lowpass filtering operations that split one lower-resolution bin into two higher-resolution bins where the filtering depends on whether the bin number i is even or odd (starting with i = 0 for the lowest bin).

This means that if one channel is TCX20, the other channel is converted to the same spectral resolution. If one or both of the subframes of the other channel are subdivided again into two TCX5 “sub-subframes”, these are first converted to TCX10 resolution by the same filtering before splitting again to arrive at the final TCX20 representation. An example of such a conversion of bins with lower time resolution to higher resolution is shown in Figure 6.3-21.

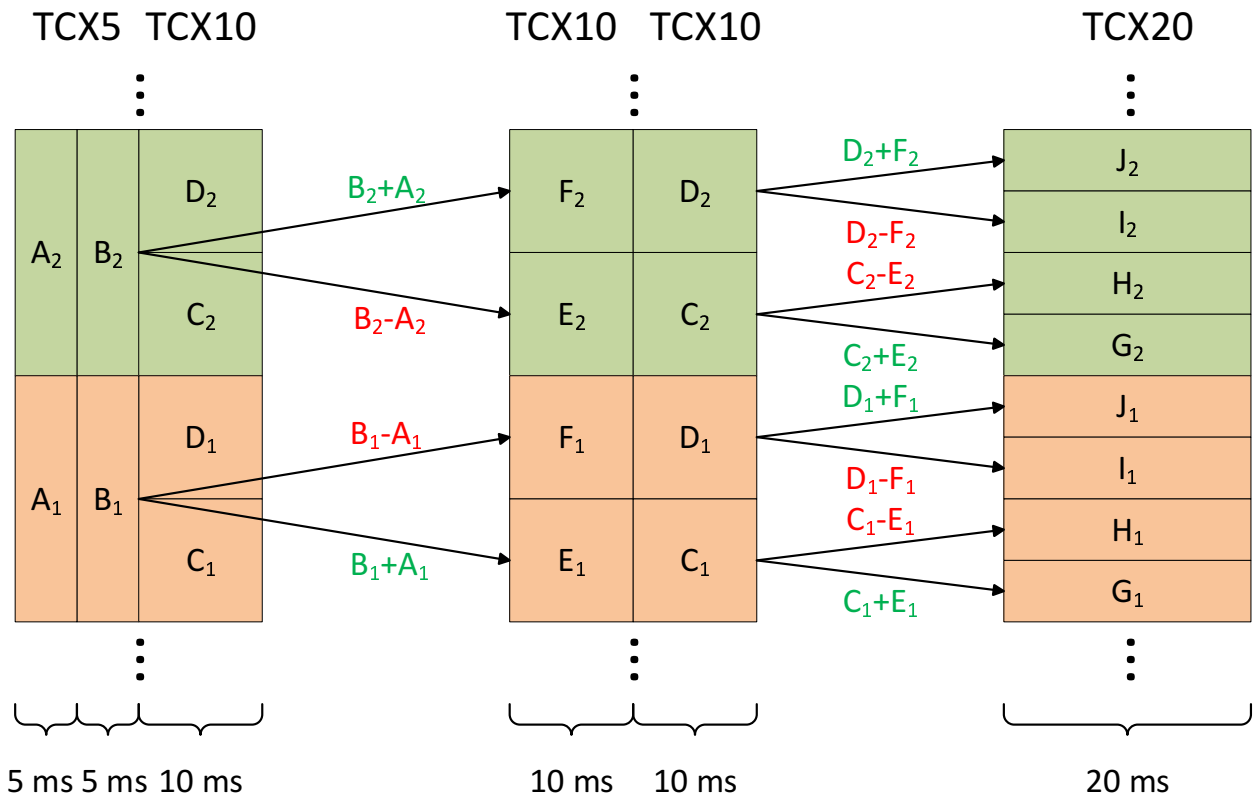


Figure 6.3-21: Example of conversion of MDCT coefficients

Even if none of the channels is TCX20, conversion to the higher resolution may still be necessary for one or both subframes in case there is TCX10 in one channel and TCX5 in the other. As an example, if the left channel is TCX10 in subframe A and 2 x TCX5 in subframe B, while the right channel is 2 x TCX5 in subframe A and TCX10 in subframe B, both channels will be converted to have TCX10 resolution in both subframes (convert subframe B for left channel, A for right channel). If in the same example the right channel is also TCX 10 for subframe A and 2 x TCX5 for B, then no conversion is done; i.e. subframe A will be downmixed with TCX10 resolution, B with TCX5.

The MDST estimates (see equation (6.3-191)) and the final channel weights (see equations (6.3-189) and (6.3-190) are then computed using these converted spectra. The weights themselves are applied to the original input spectra which means that in case of a conversion each computed weight is applied to all bins covering the same frequency range in the original lower resolution for every subframe.

6.3.6.2.3.6 CNG for mono output

In CNG operation, instead of always generating two channels of comfort noise and downmixing to a single channel afterwards, only one channel of comfort noise is generated in the first channel and shaped using an average noise parameter data vector. Additionally, some buffers are synchronized or passively downmixed. In transition frames (inactive frames following an active frame or active frames following an inactive frame), a crossfade is performed between the passive downmix of the two channels and the mono comfort noise signal in the first channel.

In a transition-to-CNG frame, two channels of comfort noise are still generated and the passive downmix is applied as in active frame decoding. Afterwards, a passive downmix with equal weighting is applied to the noise parameter data vector to generate a mid channel parametric noise data vector which will be used for CNG in this inactive frame period:

$$N^{SID}_{M,FD-CNG}(i) = \frac{N^{SID}_{L,FD-CNG}(i) + N^{SID}_{R,FD-CNG}(i)}{2}. \quad (6.3\text{-}200)$$

For all following SID frames in this inactive frame period, this conversion is done before CNG operation. The CNG operation itself in all following frames of this inactive-frame period is done only once in the first channel to generate a single output channel using the mid channel parametric noise data vector. This process is the same as FD-CNG for mono as described in [3], clause 6.7.3.3. To keep buffers consistent in both channels for possibly switching back to

active frame decoding in the next frame, the output buffers of the first channel are copied over to the second channel after CNG operation is finished for the current frame. This way, output buffers for the second channel always contain meaningful data even though CNG synthesis is only done in the first channel.

In transition frames, decoding operation switches between generating only one channel (in mono CNG) and generating two channels and applying the downmix. To achieve smooth transitions between these two output scenarios, a crossfade is performed in every transition frame which blends between the single output channel of the CNG operation and the downmix of the actively decoded channels. For transitions **to** active decoding frames, the following equation applies.

$$D_{out,i} = \begin{cases} D_{CNG,i} \times \frac{i}{L_{crossfade}} + D_i \times \left(1 - \frac{i}{L_{crossfade}}\right), for\ i < L_{crossfade} \\ D_i, for\ i \geq L_{crossfade} \end{cases} \quad (6.3\text{-}201)$$

where $L_{crossfade}$ depends on the output sampling rate and is 96 for 48 kHz, 64 for 32 kHz and 32 for 16 kHz. For transitions from active decoding frames, the following equation is used

$$D_{out,i} = \begin{cases} D_{CNG,i} \times \left(1 - \frac{i}{L_{crossfade}}\right) + D_i \times \frac{i}{L_{crossfade}}, for\ i < L_{crossfade} \\ D_{CNG,i}, for\ i \geq L_{crossfade} \end{cases} \quad (6.3\text{-}202)$$

Here, $L_{crossfade}$ is set to a quarter of the current frame length and is 240 for 48 kHz, 160 for 32 kHz and 80 for 16 kHz. In both equations, D_i denotes the passively downmixed signal as given by equation (6.3-197).

6.3.6.3 Stereo-to-MC output

The conversion from the stereo format to a multi-channel output is a subcase of the MC output format conversion described in clause 6.7.7.

6.3.7 Stereo audio decoding with TSM

The render granularity is always 1.25 ms for stereo audio decoding. If the output format is also stereo, the stereo frames are decoded fully before the TSM, the transport channel buffer acts as a simple output buffer. If the output format is mono, the stereo frames are decoded and the stereo-to-mono output according to clause 6.3.6.2 before the TSM and the transport channel buffer acts as a simple output buffer. On a bit rate switch nothing has to be done for the transport channel buffer, since the number of channels handled by it will stay constant regardless of the bit rate.

6.4 Scene-based audio (SBA) decoding

6.4.1 Combined DirAC and SPAR based SBA decoding overview

6.4.1.1 Overview

Figure 6.4-1 shows the overview of SBA decoding in IVAS. The decoder receives IVAS bitstream as described in clause 8.3. From the IVAS bitstream, the decoder extracts and decodes the IVAS format from the 3-bit common header section of the bitstream. The IVAS format points to SBA format mode if the bitstream is generated by SBA encoder. Then 2-bit SBA order and 1-bit planar SBA information is extracted and decoded from the SBA format header section of the bitstream. The SBA order information is then used to identify the row index of IVAS bitrate distribution table in SPAR mode as described in clause 6.4.3.3. The DirAC and SPAR metadata bits are extracted from the metadata payload section of the bitstream and decoded by the spatial metadata decoder to generate DirAC and SPAR metadata. The SBA decoder reads core-coder bits, which can correspond to SCE, CPE or MCT, from the core coding payload section of the bitstream. The selection of core-coder is as described in Table 5.4-1. Core-coder bits are decoded by corresponding core-decoder outputting a first set of transport channels representing the downmix signal $s^{DMX}(n)$ with N_{dmx} channels (first set of transport channels).

The downmix signal $s^{DMX}(n)$, DirAC and SPAR metadata are then upmixed and rendered to one of the supported output formats in IVAS.

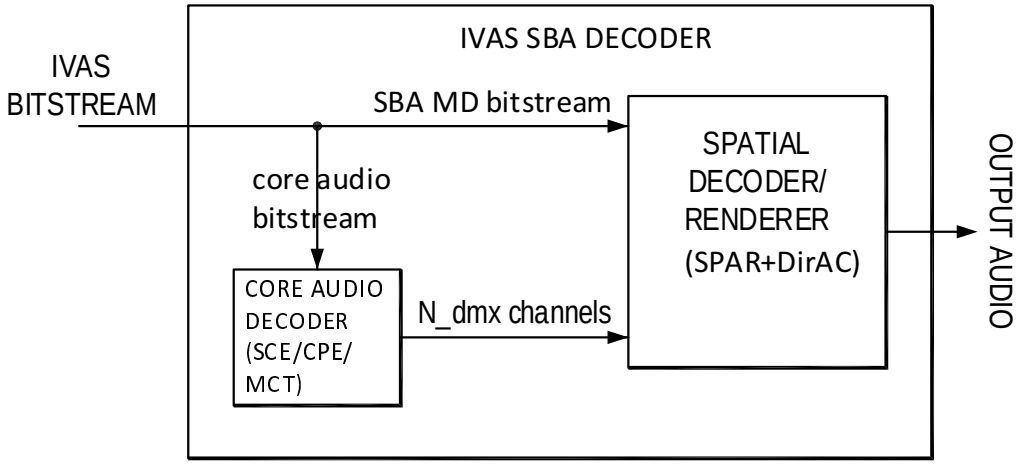


Figure 6.4-1: SBA decoder architecture in IVAS

6.4.1.2 FOA signal decoding at all bitrates and HOA2, HOA3 signal decoding at bitrates below 256 kbps

6.4.1.2.1 Overview

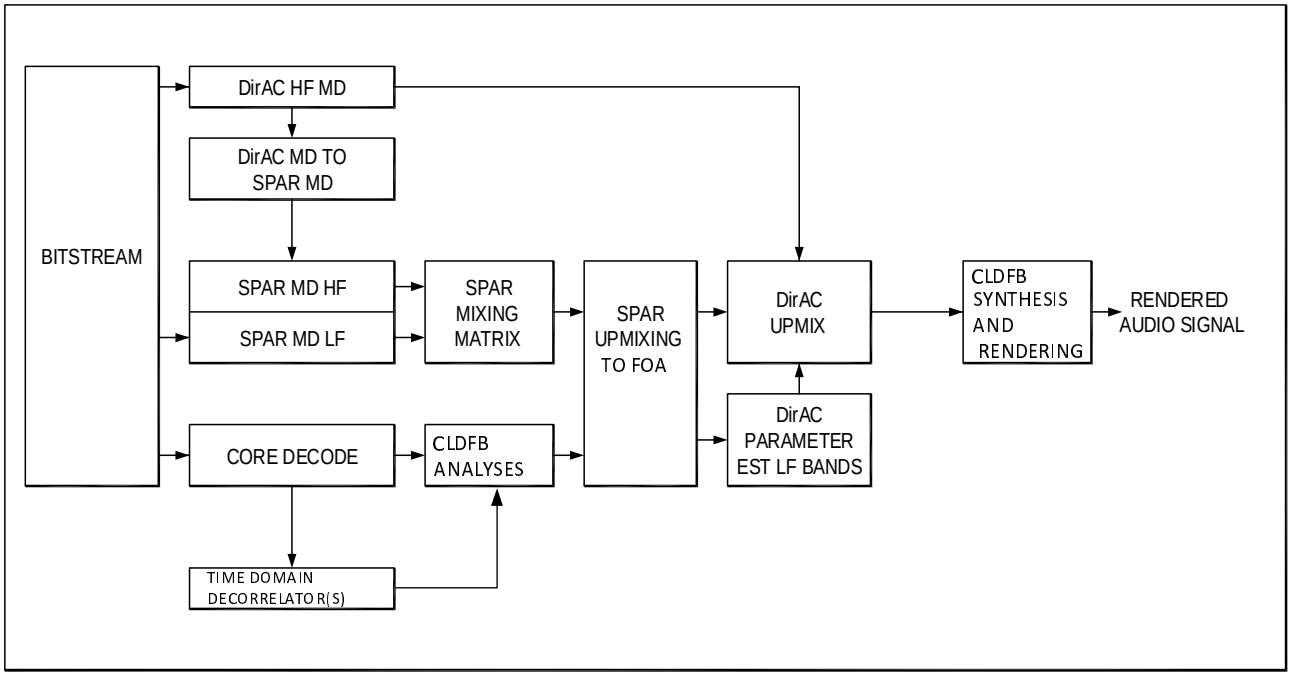


Figure 6.4-2: FOA signal decoding at all bitrates and HOA2, HOA3 signal decoding at bitrates below 256 kbps

Figure 6.4-2 shows the block diagram of SBA decoding of FOA signal at all bitrates and HOA2, HOA3 signal at bitrates below 256 kbps. The decoding is done in following stages.

- Metadata decoding (clause 6.4.1.2.2)
- Core-coder decoding (clause 6.4.1.2.3)
- SBA upmix and rendering (clause 6.4.1.2.4)

6.4.1.2.2 Metadata decoding

DirAC metadata decoder block (DirAC HF MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in high frequency range f^{DirAC}, such that f^{DirAC} > 4.4 kHz, as described in clause 6.4.2. The decoded DirAC parameters are processed through DirAC to SPAR converter, as described in clause 6.4.4.3, to generate frequency banded SPAR parameters in high frequency range f^{DirAC}.

SPAR metadata decoder block (SPAR MD LF) reads SPAR metadata bits from IVAS bitstream and decodes frequency banded SPAR parameters in low frequency range f^{SPAR}, such that $f^{SPAR} \leq$ 4.4 kHz, as described in clause 6.4.3. SPAR parameters in low and high frequency bands are then converted into the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ as described in clause 6.4.5 where SPAR metadata analysis channels, N_{spar_ana}, are described in Table 5.4-2 and downmix channels N_{dmx} are described in Table 5.4-1.

The DirAC model parameters are transmitted in the bitstream in encoded form. They are decoded by the spatial metadata decoder according to clause 6.2.4. The decoded parameters comprise a DoA and a diffuseness parameter for each of the 4 highest parameter bands (DirAC) of the MDFT filterbank in the encoder, where the two highest bands are averaged and transmitted as one band, effectively transmitting 3 parameter bands.

These parameters are understood to apply to the frequency bands of the encoder-side filterbank. On the decoder side, a CLDFB filterbank (see clause 6.2.5.1) is employed. This filterbank features a frequency resolution of 60 bands and a time resolution of 1.25 ms. The 3 transmitted parameter bands are mapped to the 60 CLDFB bands according to the mapping in Table 5.4-5. To take into account the different number of bands between the MDFT and the CLDFB, the bin indices in Table 5.4-5 are scaled by the corresponding factor 60/240. This approximately maps the DirAC parameters to the frequency ranges spanned by the bands of the decoder-side filterbank (CLDFB).

6.4.1.2.3 Core-coder decoding

Core-coder decoder block (clause 6.4.6.1) reads core-coder bits from IVAS bitstream and decodes SBA downmix signal S'_{dmx} using either SCE (clause 6.2.3.2), CPE (clause 6.2.3.3) or MCT (clause 6.2.3.4) decoding tools based on the mapping described in Table 5.4-1. S'_{dmx} is then high pass filtered as per clause 6.2.1.1 and outputs signal S_{dmx}^{hp20}.

6.4.1.2.4 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in clause 6.2.1.3, which generates N_{decorr} -channel signal S_{decorr} that is decorrelated with respect W'', where $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S'_{dmx} and S_{decorr}, as described in clause 6.2.5.1, that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs FOA signal $S_{umx,foa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in clause 6.4.6.4.5. If the output format is FOA then $S_{umx,foa}^{cldfb}$ is synthesized as described in clause 6.2.5.2 to generate IVAS output signal S_{out}.

For output formats other than FOA or MONO or STEREO, the SPAR upmixed signal $S_{umx,foa}^{cldfb}$ is processed through the DirAC decoder, which reconstructs the HOA output channels using the second set of transport channels, $S_{umx,foa}^{cldfb}$, and the DirAC metadata decoded from the bitstream. These channels together with the channels in $S_{umx,foa}^{cldfb}$ comprise the IVAS output signal vector S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ is processed by format conversion as described in clause 6.4.6.5.8. For cases with $N_{dmx} \leq 2$, an optimized output processing is applied according to clauses 6.4.6.5.8 and 6.4.6.5.8.5.

6.4.1.3 HOA2 and HOA3 signal decoding at bitrate 256 kbps

6.4.1.3.1 Overview

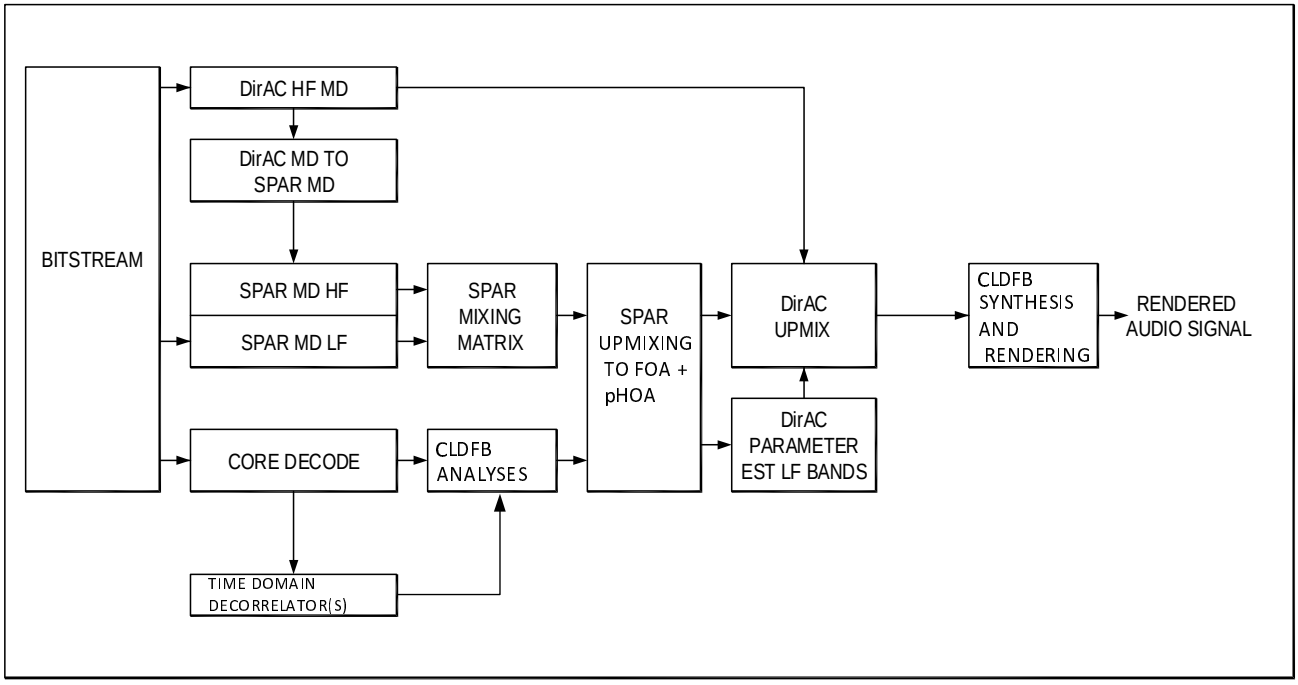


Figure 6.4-3: HOA2 and HOA3 signal decoding at bitrate 256 kbps

Figure 6.4-3 shows the block diagram of SBA decoding of HOA2 and HOA3 signal at IVAS bitrate 256 kbps. The decoding is done in following stages:

- Metadata decoding (clause 6.4.1.3.2)
- Core-coder decoding (clause 6.4.1.3.3)
- SBA upmix and rendering (clause 6.4.1.3.4)

6.4.1.3.2 Metadata decoding

DirAC metadata decoder block (DirAC HF MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in high frequency range f^{DirAC}, such that $f^{DirAC} > 4.4$ kHz, as described in clause 6.4.2. The decoded DirAC parameters are processed through DirAC to SPAR converter, as described in clause 6.4.4.3, to generate frequency banded SPAR parameters corresponding to FOA channels in the high frequency range f^{DirAC}.

SPAR metadata decoder block (SPAR MD LF) reads SPAR metadata bits from IVAS bitstream and decodes frequency banded SPAR parameters, as described in clause 6.4.3. Decoded SPAR parameters include SPAR parameters corresponding to FOA channels in the low frequency range f^{SPAR}, such that $f^{SPAR} \leq 4.4$ kHz, and SPAR parameters corresponding to HOA channels in both f^{SPAR} and f^{DirAC} frequency ranges. SPAR parameters in low and high frequency bands are then converted into the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ as described in clause 6.4.5, here SPAR metadata analysis channels, N_{spar_ana}, are described in Table 5.4-2 and downmix channels N_{dmx} are described in Table 5.4-1.

6.4.1.3.3 Core-coder decoding

Same as clause 6.4.1.2.3.

6.4.1.3.4 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in clause 6.2.1.3, which generates N_{decorr} -channel signal S_{decorr} that is decorrelated with respect W'', where $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S'_{dmx} and S_{decorr} as described in clause 6.2.5.1 that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs FOA and planar HOA signal $S_{umx,hoa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in clause 6.4.6.4.5. If the output format is FOA then FOA component of $S_{umx,hoa}^{cldfb}$ is synthesized as described in clause 6.2.5.2 to generate IVAS output signal S_{out}.

For output formats other than FOA or MONO or STEREO, the SPAR upmixed signal $S_{umx,hoa}^{cldfb}$ is processed through the DirAC decoder, which reconstructs the HOA output channels using the second set of transport channels, $S_{umx,hoa}^{cldfb}$, and the DirAC metadata decoded from the bitstream. These channels together with the channels in $S_{umx,hoa}^{cldfb}$ comprise the IVAS output signal vector S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ is processed by format conversion as described in clause 6.4.6.5.8.

6.4.1.4 HOA2 and HOA3 signal decoding at bitrates 384 kbps

6.4.1.4.1 Overview

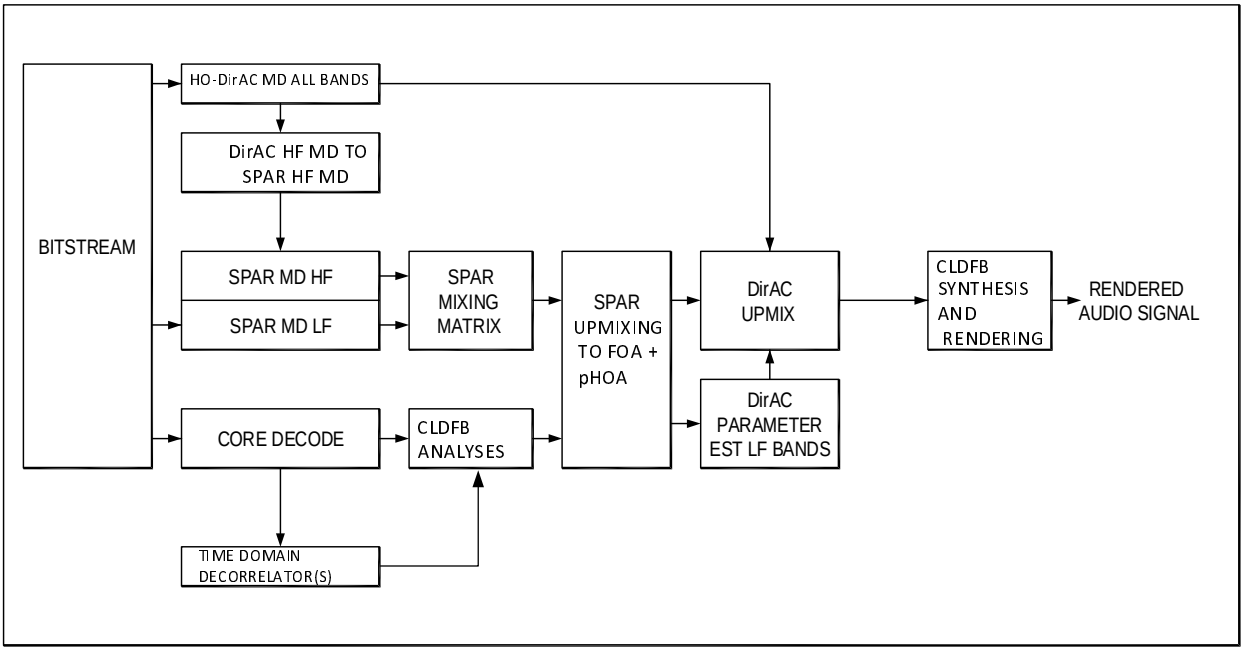


Figure 6.4-4: HOA2 and HOA3 signal decoding at bitrate 384 kbps

Figure 6.4-4 shows the block diagram of SBA decoding of HOA2 and HOA3 signal at IVAS bitrate 384 kbps. The decoding is done in following stages:

- Metadata decoding (clause 6.4.1.4.2)
- Core-coder decoding (clause 6.4.1.4.3)
- SBA upmix and rendering (clause 6.4.1.4.4)

6.4.1.4.2 Metadata decoding

Higher Order DirAC metadata decoder block (HO- DirAC MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in both f^{SPAR} and f^{DirAC} frequency ranges, as described in clause 6.4.2. The decoded DirAC parameters in f^{DirAC} frequency range are processed through DirAC to SPAR converter, as described in clause 6.4.4.3 to generate frequency banded SPAR parameters corresponding to FOA channels in the high frequency range f^{DirAC}.

SPAR metadata decoding is same as clause 6.4.1.3.2.

6.4.1.4.3 Core-coder decoding

Same as clause 6.4.1.2.3.

6.4.1.4.4 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in clause 6.2.1.3, which generates N_{decorr} channel signal S_{decorr} that is decorrelated with respect W'', here $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S'_{dmx} and S_{decorr}, as described in clause 6.2.5.1, that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs FOA and planar HOA signal $S_{umx,hoa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in clause 6.4.6.4.5. If the output format is FOA then FOA component of $S_{umx,hoa}^{cldfb}$ is synthesized as described in clause 6.2.5.2 to generate IVAS output signal S_{out}. For output formats other than FOA or MONO or STEREO, the SPAR upmixed signal $S_{umx,hoa}^{cldfb}$ is processed through the DirAC decoder, which reconstructs the HOA output channels using the second set of transport channels, $S_{umx,hoa}^{cldfb}$, and the DirAC metadata decoded from the bitstream. These channels together with the channels in $S_{umx,hoa}^{cldfb}$ comprise the IVAS output signal vector S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x N_{dmx}]}$ is processed by format conversion as described in clause 6.4.6.5.8.

6.4.1.5 HOA2 and HOA3 signal coding at bitrates 512 kbps

6.4.1.5.1 Overview

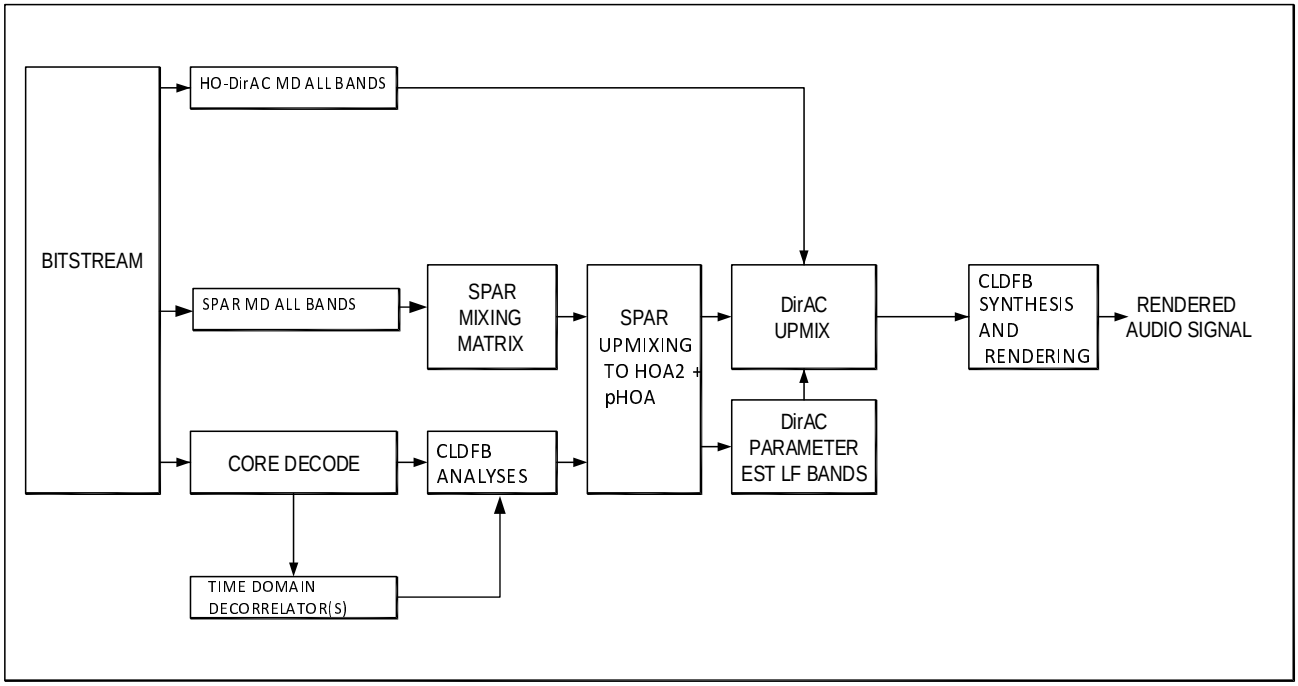


Figure 6.4-5: HOA2 and HOA3 signal decoding at bitrate 512 kbps

Figure 6.4-5 “Figure 6.4-5” shows the block diagram of SBA decoding of HOA2 and HOA3 signal at IVAS bitrate 512 kbps. The decoding is done in following stages:

- Metadata decoding (clause 6.4.1.5.2)
- Core-coder decoding (clause 6.4.1.5.3)
- SBA upmix and rendering (clause 6.4.1.5.4)

6.4.1.5.2 Metadata decoding

Higher Order DirAC metadata decoder block (HO- DirAC MD) reads DirAC metadata bits from IVAS bitstream and decodes frequency banded DirAC parameters in in both f^{SPAR} and f^{DirAC} frequency ranges, as described in clause 6.4.2.

SPAR metadata decoder block (SPAR MD) reads SPAR metadata bits from IVAS bitstream and decodes frequency banded SPAR parameters corresponding to HOA2 and planar HOA3 channels in both f^{SPAR} and f^{DirAC} frequency ranges as described in clause 6.4.3. SPAR parameters are then converted into the upmix matrix $umx_{[N_{spar_{ana}} x N_{dmx}]}$ as described in clause 6.4.5, here SPAR metadata analysis channels, N_{spar_ana}, are described in Table 5.4-2 and downmix channels N_{dmx} are described in Table 5.4-1.

6.4.1.5.3 Core-coder decoding

Same as clause 6.4.1.2.3.

6.4.1.5.4 SBA upmix and rendering

The high pass filtered downmix signal S_{dmx}^{hp20} , upmix matrix $umx_{[N_{spar_{ana}} x N_{dmx}]}$ and decoded metadata are then processed by SBA upmix and rendering signal chain to generate IVAS output signal S_{out}. SBA upmix and rendering signal chain includes following steps.

First, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} is processed by Time Domain Decorrelator block, as described in detail in clause 6.2.1.3, which generates N_{decorr} channel signal S_{decorr} that is decorrelated with respect W'', where $N_{decorr} = N_{spar\_ana} - N_{dmx}$. Then CLDFB analysis is done on S'_{dmx} and S_{decorr}, as described in subclause 6.2.5.1, that generates CLDFB domain signal S^{cldfb}. SPAR upmixing block then applies the upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ to the CLDFB signal S^{cldfb} and outputs HOA2 and planar HOA3 signal $S_{umx,hoa}^{cldfb}$ in CLDFB domain, the upmixing process is described in detail in clause 6.4.6.4.5. If the output format is FOA then the FOA component of $S_{umx,hoa}^{cldfb}$ is synthesized as described in clause 6.2.5.2 to generate IVAS output signal S_{out}.

For output formats other than FOA or MONO or STEREO, the SPAR upmixed signal $S_{umx,hoa}^{cldfb}$ is processed through the DirAC decoder, which reconstructs the HOA output channels using the second set of transport channels, $S_{umx,hoa}^{cldfb}$, and the DirAC metadata decoded from the bitstream. These channels together with the channels in $S_{umx,hoa}^{cldfb}$ comprise the IVAS output signal vector S_{out}.

For MONO and STEREO output formats, the primary downmix channel W'' of downmix signal S_{dmx}^{hp20} and upmix matrix $umx_{[N_{spar_{ana}} x\, N_{dmx}]}$ is processed by format conversion as described in clause 6.4.6.5.8.

6.4.2 DirAC parameter decoding

SBA parameters are decoded based on metadata active/inactive mode flag W_{vad} that is read from the bitstream. W_{vad} flag indicates whether metadata was coded in active or inactive mode.

The decoding of the DirAC parameters is performed as described in clause 6.2.4. In the low-order operation mode, this decoding provides one DoA angle $\theta_D(m, b)$ per subframe m and DirAC parameter band b and one global diffuseness parameter per frame $\Psi(b)$ and DirAC parameter band b. The parameters are only received from the bitstream for the high-frequency parameter bands (DirAC bands). Hence, the time resolution of the DoAs is 5 ms, whereas that of the diffuseness is 20 ms.

In the high-order operation mode, two DoA angles $\theta_0(m, b)$ and $\theta_1(m, b)$ are obtained per subframe m and parameter band b. One relative directionality $a_0(b)$, and one global diffuseness $\Psi(b)$ parameter are decoded per frame and parameter band b. These metadata are decoded from the bitstream for all parameter bands.

The decoded parameters are then expanded to the CLDFB bands. In the low-order mode, dithering is applied.

6.4.3 SPAR parameter decoding

6.4.3.1 General

SBA ambisonics order, that is stored as a 2 bit field in SBA header section of IVAS bitstream as described in clause 8.3 and SBA planar mode, that is stored as a 1 bit field in SBA header section of IVAS bitstream as described in clause 8.3, are decoded. Based on IVAS bitrate and SBA order, SBA analysis order is set as per Table 5.4-2. Then SPAR bitrate distribution table row index is computed based on IVAS bitrate and SBA analysis order as per Table 5.4-4.

6.4.3.2 Set active W prediction flag

Dynamic active W flag, $Dyn_{activeW}$, and active W residual index, Res_{ind}^{bs}, are decoded from metadata section of IVAS bitstream as described in Table 8.3-3 and Table 8.3-4. The active W prediction flag is then set as per Equation (5.4-27) and Res_{ind}^{bs} is set as follow.

$$Res_{ind} = \begin{cases} Res_{ind}^{bs} & , if\ N_{dmx} = 2 \\ 3 & , if\ N_{dmx} = 3 \end{cases} \quad (6.4\text{-}1)$$

6.4.3.3 Decode quantization and coding strategy bits

Metadata active/inactive flag MD_{active} is read from the bitstream as described in clause 8.3. If MD_{active} is set to one, then SPAR quantization strategy index and coding strategy index are decoded from IVAS bitstream from the SPAR quantization strategy section and SPAR coding strategy section as described in clause 8.3. Number of SPAR quantization strategy bits, $spar_{qbits}$, depend on IVAS bitrate as given below.

$$spar_{qbits} = \begin{cases} 2 & , if\ IVAS\ bitrate < 256\ \text{kbps} \\ 1 & , otherwise \end{cases}$$

Number of SPAR coding strategy bits, $spar_{CSbits}$, are fixed as given below.

$$spar_{CSbits} = 3$$

The SPAR bitrate distribution table row index is determined using IVAS bitrate and SBA order. From the SPAR bitrate distribution table row index and quantization strategy index, quantization strategy is decoded as $\{qlvl_{pr}, qlvl_{cp}, qlvl_{d}, 1\}$, where $qlvl_{pr}$ is the number of quantization points for PR coefficients, $qlvl_{cp}$ is the number of quantization points for CP coefficients, $qlvl_{D}$ is the number of quantization points for D coefficients. Based on the values of $qlvl_{pr}, qlvl_{cp}, qlvl_{d}$ and SPAR coding strategy, corresponding Arithmetic and Huffman coder probability distribution models are read as described in clause 5.4.3.7.16 and clause 5.4.3.7.17. SPAR indices corresponding to PR, CP and D coefficients are then decoded with either arithmetic or Huffman decoder as described in clause 6.4.3.6 and clause 6.4.3.7. Number of PR, CP and D indices to be read from bitstream, depend on SPAR analysis channels. For FOA channels, SPAR parameters are coded for low frequency bands with band index ranging from 1 to nB_{spar}^{foa}, where $nB_{spar}^{foa} = \frac{8}{bw_{bands}}$ and bw_{bands} is given in equation (6.4-2), SPAR parameters for frequency band index greater than nB_{spar}^{foa} are computed from DirAC parameters as described in clause 6.4.4.3. For HOA2 and HOA3 channels, SPAR parameters are coded for all frequency bands with band index ranging from 1 to nB_{spar}^{hoa}, where $nB_{spar}^{hoa} = \frac{12}{bw_{bands}}$ and bw_{bands} is given in equation (6.4-2).

Number of PR, CP and D indices are given as follows.

$$N_{PR}^{indices} = 3\ nB_{spar}^{foa} + (N_{spar\_ana} - 4)(nB_{spar}^{hoa})$$

$$N_{CP}^{indices} = \begin{cases} (N_{dmx} - 1)\,(N_{spar\_ana} - N_{dmx})nB_{spar}^{foa} & , if\ N_{spar\_ana} = 4 \\ (N_{dmx} - 1)\,(N_{spar\_ana} - N_{dmx})nB_{spar}^{hoa} & , if\ N_{spar\_ana} > 4 \end{cases}$$

$$N_{D}^{indices} = \begin{cases} (N_{spar\_ana} - N_{dmx})nB_{spar}^{foa} & , if\ N_{spar\_ana} = 4 \\ (N_{spar\_ana} - N_{dmx})nB_{spar}^{hoa} & , if\ N_{spar\_ana} > 4 \end{cases}$$

where bw_{bands} is computed as

$$bw_{bands} = \begin{cases} 4 & , if\ W_{vad} = 0 \\ 2 & , if\ W_{vad} = 1\,, ivas\ bitrate \leq 16.4\ kbps \\ 1 & , otherwise \end{cases} \quad (6.4\text{-}2)$$

The decoded SPAR indices are then de-indexed and converted to quantized SPAR parameters as per Equation (5.4-69).

6.4.3.4 Time differential decoding

6.4.3.4.1 General

If SPAR coding strategy is one of the four time differential coding schemes as described in Table 5.4-9 and IVAS bitrate is greater than 16.4 kbps, then the time-differentially coded bands are decoded with time-differential decoding method. Time differential decoding where quantization strategy is same between previous frame and current frame is described in clause 6.4.3.4.2. Time differential decoding where quantization strategy is different between previous frame and current frame is described in clause 6.4.3.4.3.

6.4.3.4.2 Time differential decoding across same quantization strategies

When quantization strategy is same between previous frame and current frame, non-differential index is computed from time-differential index in three steps. First the time-differential index is decoded from the bitstream, then the time-differential index is added to the previous frame's non-differential index as follows.

$$A_{current}^{index} = A_{TD}^{index} + A_{previous}^{index} \quad (6.4\text{-}3)$$

Then modulo $qlvl_{A}$ is performed to keep the indices range as $\{A_{min}^{index}, A_{max}^{index}\}$

$$A_{current}^{index} = \begin{cases} A_{current}^{index} + qlvl_A & if\ A_{current}^{index} < A_{min}^{index} \\ A_{current}^{index} - qlvl_A & if\ A_{current}^{index} > A_{max}^{index} \\ A_{current}^{index} & otherwise \end{cases}, \quad (6.4\text{-}4)$$

where $A_{min}^{index} = round\left(\frac{((qlvl_A-1)*\mathrm{A}_{min})}{A_{max}-A_{min}}\right)$, $A_{max}^{index} = round\left(\frac{((qlvl_A-1)*\mathrm{A}_{max})}{A_{max}-A_{min}}\right)$, A refers to PR, CP and D coefficients and $qlvl_A$ is the number of quantization points as per the current frame's quantization strategy read from SPAR bitrate distribution table. $A_{max} and\ A_{min}$ are the minimum and maximum quantized values as per the current frame's quantization strategy as per Equations (5.4-65), (5.4-66) and (5.4-67).

6.4.3.4.3 Time differential decoding across different quantization strategies

When quantization strategy is different between previous frame and current frame, non-differential index is computed from time-differential index in four steps. First the time-differential index is decoded from the bitstream, then the previous frame's non-differential index is mapped to current frame's quantization strategy as per Equation (5.4-74), then the time-differential index is added to the mapped previous frame's non-differential index as follows.

$$A_{current}^{index} = A_{TD}^{index} + A_{previous,mapped}^{index} \quad (6.4\text{-}5)$$

Then modulo $qlvl_A$ is performed as described in Equation (6.4-4).

6.4.3.5 Band interleaved decoding at 13.2 kbps and 16.4 kbps

If the SPAR coding strategy is one of the four time-differential coding schemes described in Table 5.4-9 and the IVAS bitrate is either 13.2 kbps or 16.4 kbps, then SPAR parameters are coded with a 40ms metadata update rate using the band-interleaved method described in clause 5.4.3.7.15. In this case, the number of bands to be read from bitstream is $\frac{nB_{spar}^{foa}}{2}$ and, depending on the coding scheme indicated in the bitstream (4a, 4b, 4c or 4d), the corresponding parameters are either read to band indices that are labelled A or band indices that are labelled B in Table 5.4-11 and Table 5.4-12. The remaining band indices use the quantized parameters from the previous frame.

6.4.3.6 Arithmetic decoder

If the coding strategy is either BASE non-differential entropy coding scheme or one of the four time differential coding schemes, then that indicates Arithmetic coding of SPAR parameters as described in clause 5.4.3.7.16.

For each type of SPAR parameter (PR, CP or D) a 2 bit probability distribution model index is read from the bitstream and then the corresponding probability distribution model is read from Table 5.4-13, Table 5.4-14, Table 5.4-15 depending on the coding strategy and quantization strategy. Metadata bitstream is then decoded with Arithmetic decoder.

6.4.3.7 Huffman decoder

If the coding strategy is BASE_NOEC, then that indicates Huffman coding of SPAR parameters as described in clause 5.4.3.7.17. Huffman codebook is read from Table 5.4-16, Table 5.4-17 and Table 5.4-18 depending on quantization strategy. Metadata bitstream is then decoded with Huffman decoder.

6.4.4 SPAR and DirAC parameter merge

6.4.4.1 SBA mono handling

Decoder Mono Detector/Preserver

The mono detector/preserver in the decoder looks at the signalling through the bitstream (or implicit signalling through the metadata). If mono is signalled, then the mechanism sets the DirAC energy ratio to be zero and diffuseness to be one for every frequency band and subframe in the decoder. This ensures that the W channel energy is not spread across the other channels and the signal remains as mono content.

The metadata values that the mono detector/preserver examines are:

- SPAR metadata

- Prediction coefficients (PR)
- Cross-prediction coefficients (CP)
- Decorrelation coefficients (D)

– DiRAC metadata

- energyratio ($1 - \Psi$)
- azimuth (ϕ)
- elevation (θ)

The detector/preserver looks for these 6 values in each band, if any of the values in any band do not match the mono expected values, then the whole block is declared non-mono. Otherwise the block is declared mono. All the expected values for mono are zero, except for energy_ratio which is expected to be < 0.15 due to quantisation of that value.

When mono is detected in the decoder, the energy_ratio values are reset to zero for all bands and time slices in the block, and the diffuseness vector is set to all ones. This results in the correct behaviour in the deocoder for mono content contained within an ambisonics format.

6.4.4.2 SPAR to DirAC parameter conversion

In bitrate switching scenarios when bitrate is switched between IVAS bitrate greater than or equal to 384 kbps and IVAS bitrate less than 384 kbps, then in the transition frame, DirAC parameters are estimated from quantized SPAR parameters, in bands 1 to nB_{spar}^{foa} , using SPAR to DirAC metadata converter. The direction vector is generated as given below.

$$dv_x = \frac{PR_x}{\sqrt{\left(PR_x^2 + PR_y^2 + PR_z^2\right)}}$$

$$dv_y = \frac{PR_y}{\sqrt{\left(PR_x^2 + PR_y^2 + PR_z^2\right)}}$$

$$dv_z = \frac{PR_z}{\sqrt{\left(PR_x^2 + PR_y^2 + PR_z^2\right)}}$$

where PR_x, PR_y and PR_z are SPAR prediction coefficients corresponding to X, Y and Z channel respectively.

Azimuth and elevation angles are then generated as

$$Az = \arctan\left(\frac{dv_y}{dv_x}\right) * \frac{180}{\pi} \quad (6.4\text{-}6)$$

$$El = \arctan\left(\frac{dv_z}{\sqrt{dv_x^2 + dv_y^2}}\right) * \frac{180}{\pi} \quad (6.4\text{-}7)$$

Energy ratio computation depends on number of downmix channels N_{dmx} as given Equation (6.4-9) and (6.4-11).

Number of downmix channels greater than one

In this case energy ratio and diffuseness parameters are computed directly from SPAR prediction coefficients as follows.

$$en_i = \left(PR_x^2 + PR_y^2 + PR_z^2\right)$$

$$en_{lt} = 0.25en_i + 0.75en_{lt,-1} \quad (6.4\text{-}8)$$

$$en_{ratio} = \sqrt{en_{lt}} \quad (6.4\text{-}9)$$

where en_{lt} is long term average of en_i, $en_{lt,-1}$ is previous frame's en_{lt}.

Diffuseness parameter is then computed as

$$\psi = 1 - en_{ratio} \quad (6.4\text{-}10)$$

Number of downmix channels equal to one

In this case energy ratio and diffuseness parameters are computed directly from SPAR active W prediction coefficients and decorrelation coefficients as follows.

$$en_i = \left(PR_x^2 + PR_y^2 + PR_z^2\right)$$

where PR_x, PR_y and PR_z are SPAR prediction coefficients corresponding to X, Y and Z channel respectively.

$$diff_i = \left(D_x^2 + D_y^2 + D_z^2\right)$$

where D_x, D_y and D_z are SPAR decorrelation coefficients corresponding to X, Y and Z channel respectively.

$$en_{lt} = 0.25en_i + 0.75en_{lt,-1}$$

where en_{lt} is long term average of en_i, $en_{lt,-1}$ is previous frame's en_{lt}.

$$pow_i = 0.5((1 - f_s en_i)^2 + en_i + diff_i)$$

$$pow_{lt} = 0.25pow_i + 0.75pow_{lt,-1}$$

where pow_{lt} is long term average of pow_i, $pow_{lt,-1}$ is previous frame's pow_{lt}. f_s is given in Equation (6.4-15).

$$en_{ratio} = \frac{\sqrt{en_{lt}}}{\epsilon + pow_{lt}} \quad (6.4\text{-}11)$$

Diffuseness parameter is then computed as

$$\psi = 1 - en_{ratio} \quad (6.4\text{-}12)$$

6.4.4.3 DirAC model for the SPAR parameters

6.4.4.3.1 Overview

The inter-channel covariance matrix employed in the calculation of the upmix matrix is derived from the acoustic model parameters in the same way as on the encoder side in clause 5.4.3.8.5. The calculation is repeated independently based on the model parameters received from the bitstream.

DirAC to SPAR parameter conversion is done differently for parameters corresponding to downmix channels, that is channel index i with $1 \leq i \leq N_{dmx}$, and for parameters corresponding to parametric FOA channels, that is channel index i with $N_{dmx} \leq i \leq 4$ as follows.

6.4.4.3.2 DirAC to SPAR conversion for parameters corresponding to downmix channels

SPAR parameters corresponding to downmix channels are only PR coefficients and they are generated with 20 ms time resolution as done in encoder as described in clause 5.4.3.8.2 and clause 5.4.3.8.5.

6.4.4.3.3 DirAC to SPAR conversion for PR, CP and D coefficients corresponding to parametric channels

SPAR parameters are generated with same time resolution as the DirAC parameters obtained from the bitstream by evaluating the steps described in clause 5.4.3.8.2 for every subframe instead of averaging over all subframes. The spherical harmonics are evaluated with the DirAC DoA:

$$Y_{lm}(\theta_i) \quad (6.4\text{-}13)$$

With the subframe index $1 \le i \le 4$. θ_m is the quantized DirAC azimuth angle in subframe i. Here, this contains both the azimuth and elevation angles of the DoA. $Y_{lm}(\theta_i)$ is then used to generate covariance matrix in subframe m as described in clause 5.4.3.8.2. The covariance matrix in each subframe is then converted to SPAR parameters as described in clause 5.4.3.8.5.

6.4.5 Upmix matrix calculation

6.4.5.1 Overview

The upmix matrix is calculated using different methods for different frequency bands. For the low frequency bands (SPAR), the matrix is derived from the SPAR parameters decoded from the bitstream. For the high frequency bands (DirAC), the same calculation of the covariance matrix as on the encoder side is repeated independently. From this covariance matrix, the SPAR parameters (prediction and decorrelation) are re-computed in the same way as on the encoder side. The upmix matrix is then derived according to clause 6.4.5.2.

6.4.5.2 SPAR matrix generation (P and C matrices)

In order to generate and upmix matrix, first P and C matrices are generated, where P matrix is generated with the wet parameters (D coefficients) as given below and C matrix is generated with dry parameters (PR and CP coefficients) with some elements of C matrix generated with the PR and CP coefficients as decoded from bitstream and remaining elements generated with pre-defined combination of PR coefficients as described below.

First prediction upmix matrix u created as follows.

$$u_{[N_{spar_{ana}} x N_{spar_{ana}}]}(m,b) = \begin{pmatrix} s(1 - f_s g^2) & -f_s\ PR^{quantized}_{[N_{PR}\ x\ 1]}(m,b)^H \\ PR^{quantized}_{[N_{PR}\ x\ 1]}(m,b) & I_{[N_{PR} x\ N_{PR}]} \end{pmatrix} \left(remix_{[N_{spar_{ana}} x N_{spar_{ana}}]}\right)^{-1} \tag{6.4-14}$$

where m is the subframe index in a frame with $1 \le m \le 4$, b is band index with $1 \le b \le nB^{hoa}_{spar}$, s is scaling factor is given in Equation (6.4-16), f_s is active W scaling factor as given in Equation (6.4-15), $PR^{quantized}_{[N_{PR}\ x\ 1]}$ are the decoded SPAR prediction coefficients, $remix_{[N_{spar_{ana}} x N_{spar_{ana}}]}$ matrix is as given in Equation (5.4-57).

$$f_s = \begin{cases} 0 & , if\ Flag_{activeW} = 0 \\ 0.25 & if\ ivas\ bitrate \le 16.4\ kbps\ OR\ W_{vad} = 0 \\ 0.5 & otherwise \end{cases} \tag{6.4-15}$$

$$s = \begin{cases} 0.0039 & , if\ Flag_{activeW} = 1, IVAS\ bitrates \ge 48kbps \\ 1 & , otherwise \end{cases} \tag{6.4-16}$$

Then decorrelation upmix matrix d and cross prediction upmix matrix v are created as follows.

$$d_{i,j}(m,b) = \begin{cases} D_j(m,b) & , if\ i > N_{dmx}, j = i - N_{dmx} \\ 0 & , otherwise \end{cases} \tag{6.4-17}$$

$$v_{i,k}(m,b) = \begin{cases} CP_{i-N_{dmx},k-1}(m,b) & , if\ i > N_{dmx}, k > 1 \\ 1 & , if\ i = k \\ 0 & , otherwise \end{cases} \tag{6.4-18}$$

where $i, j\ and\ k$ are channel indices with $1 \le i \le N_{spar\_ana}$, $1 \le j \le N_{dec}$ and $1 \le k \le N_{dmx}$.

SPAR P and C matrices are then created as follows.

$$P''_{[N_{spar_{ana}} x N_{dec}]}(m,b) = u(m,b)\, d(m,b) \tag{6.4-19}$$

$$C''_{[N_{spar_{ana}} x N_{dmx}]}(m,b) = u(m,b)\, v(m,b) \tag{6.4-20}$$

When active W is enabled, upmix matrix is further modified to just scale the W channel. This results in a different upmix strategy at the decoder than the downmix strategy at the encoder. The modifications to P and C matrix are follows.

$$P''_{i,j}(m,b) = \begin{cases} 0 & ,if\ Flag_{activeW} = 1, i = 1 \\ P''_{i,j}(m,b) & ,otherwise \end{cases} \quad (6.4\text{-}21)$$

$$C''_{i,j}(m,b) = \begin{cases} \max\left(0, C''_{i,j}(m,b)\right) & ,i = j = 1 \\ C''_{i,j}(m,b) & ,otherwise \end{cases} \quad (6.4\text{-}22)$$

Then band unmixing is performed as follows.

$$P(m, i + (b1 - 1) * bw_{bands}) = P''(m, b1) \quad (6.4\text{-}23)$$

$$C(m, i + (b1 - 1) * bw_{bands}) = C''(m, b1) \quad (6.4\text{-}24)$$

where $1 \le b1 \le nB_{spar}^{foa}$, $1 \le i \le bw_{bands}$.

6.4.5.3 Upmix matrix generation

The SPAR upmix matrix is generated as follows.

$$U_{i,j}(m,b) = \begin{cases} C_{i,j}(m,b) & ,j \le N_{dmx} \\ P_{i,j}(m,b) & ,N_{dmx} \le j \le N_{spar_{ana}} \end{cases} \quad (6.4\text{-}25)$$

where $i\ and\ j$ are channel indices with $1 \le i \le N_{spar_{ana}}$, $1 \le j \le N_{spar_{ana}}$, P matrix is given in Equation (6.4-23) and C matrix is given in Equation (6.4-24).

This upmix matrix U is applied to the downmix signal and decorrelated signal to generated SPAR upmix signal as described in clause 6.4.6.4.

6.4.6 Decoded audio processing

6.4.6.1 Downmix signal decoding (core-decoding)

The core-decoder generates the decoded audio downmix signal $\boldsymbol{s}'_{dmx}$ from the IVAS bitstream. Depending on the bitrate, this is done using an SCE (clause 6.2.3.2), a CPE (clause 6.2.3.3), or the MCT (clause 6.2.3.4). Table 5.4-1 lists the encoding/decoding tool for each IVAS bitrate. The decoded time-domain signal $\boldsymbol{s}'_{dm\,x}$ is then high-pass filtered as per clause 6.2.1.1, obtaining the signal signal $\boldsymbol{s}_{dmx}^{hp20}$.

6.4.6.2 AGC

The decoder counterpart of AGC performs an inverse gain operation to the decoded downmixed audio signal $s_i^{DMX\_dec}(n)$ following clause 6.4.6.2, to be subsequently upmixed following clause 6.4.6.4. This inverse operation is a lossless operation provided that a lossless core codec is used and that all overload conditions are properly compensated. This operation requires an inverse operation of the gain transition function used at the encoder side as described in clause 5.4.7. As previously described, the following information is needed to initially construct the gain transition function, i.e., the gain scaling factor $e(j) - e(j-1)$, gain transition step size $DBSTEP$ and the prototype smoothing function $p(\cdot)$. They are derived and obtained from the decoded AGC metadata bitstream (i.e., the gain parameters) and predefined functions/values.

The inverse gain transition function is then formulated as:

$$t^{-1}(l,j) = \begin{cases} 1, & l = 0 \ldots D-1 \\ p(l-D,j)^{-(e(j)-e(j-1))}, & l = D \ldots L-1, \end{cases} \quad (6.4\text{-}26)$$

Note that the inverse gain transition function still retains the properties of the original function, i.e., it comprises a transitory portion (lower part) and a steady-state portion (upper part). Moreover, the length of the transitory portion is limited by an overall core codec delay $D = 12$ ms or 576 samples at 48 kHz sampling frequency. Figure 6.4-6 depicts the inverse gain transition function at 48 kHz sampling frequency having the same scaling factors specified in Figure 5.4-12 at the encoder counterpart.

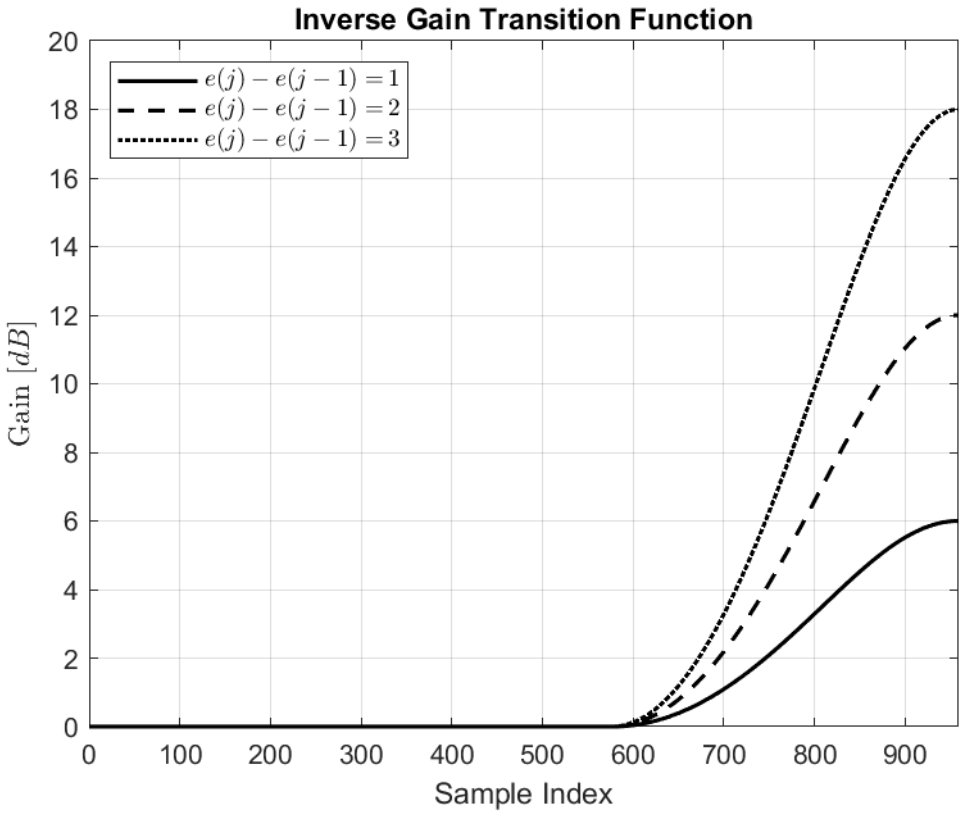


Figure 6.4-6: Inverse gain transition function having the scaling factor $e(j)-e(j-1)$ set to 1, 2 and 3

The final construct of the inverse gain transition function showing all the dependencies is as follows.

$$t^{-1}(DBSTEP, l, j) = t^{-1}(l, j) \cdot t^{-1}(DBSTEP, L-1, j-1), \quad l = 0 \cdots L-1, \tag{6.4-27}$$

where $t^{-1}(DBSTEP, L-1, j-1)$ is initialized to 1.

Setting the sample index l to n, and introducing the decoded downmix channel index $i = 1$, indicating a single downmixed channel, gives the following formulation:

$$s_{i=1}^{AGC\_dec}(n) = t^{-1}(DBSTEP, n, j) \cdot s_{i=1}^{DMX\_dec}(n). \tag{6.4-28}$$

The gain adjusted frame $s_{i=1}^{AGC\_dec}(n)$ is then subject for upmixing.

The gain parameter for constructing the inverse gain transition function is obtained from the bitstream. Upon receiving the encoded AGC metadata bitstream, the following steps are performed. Firstly, 1 bit is read indicating whether the gain control is applied to the current frame or not. If it is set to 0, a unity gain is applied to the current frame. If it is set to 1, β_E bits are read from the bitstream. These β_E bits are binary decoded to yield the gain parameter $e(j)$. For a given $\beta_E = 2$ bits, the β_E bits having the binary code from the set $\{(00), (01), (10), (11)\}$ are converted to $e(j)$ having a value of $\{0,1,2,3\}$.

When dealing with a single or multiple frames packet loss, the decoder reconstructs the lost frames and AGC defaults the gain parameter of the lost frame to that of the last received (good) frame having no gain control applied. Upon receiving an actual (good) frame, AGC internal state is updated with the information corresponding to the decoded AGC metadata bitstream. This mechanism is considered as the best effort gain control handling ensuring a smooth transition of gain parameters in the event of packet loss.

6.4.6.3 PCA

6.4.6.3.1 Handling of bit rate switching and other configurations than FOA

If the bitrate in the current frame is different from 256 kbit/s, or if the bitrate is 256 kbit/s but the number of input channels is higher than 4, the PCA processing is restricted to the following steps:

- The PCA decoder state is reset by setting the unit quaternions are set to $\hat{\boldsymbol{q}}_i^{prev} = (1,0,0,0)$.
- If the bit rate in the previous frame is different from 256 kbit/s, the output frame is identical to the input frame; otherwise, the PCA matrixing described in clause 6.4.6.3.4 is applied with quaternions $\hat{\boldsymbol{q}}_i$ set to $(1,0,0,0)$ for interpolation. The PCA decoder state is updated (see clause 6.4.6.3.5).

6.4.6.3.2 Decoding of PCA by-pass indicator

The PCA by-pass indicator is set according to the PCA bit read from the SBA bitstream. If PCA operates in inactive mode (`PCA_MODE_INACTIVE`), the PCA processing is the same as described in clause 6.4.6.3.1.

Otherwise, the PCA by-pass indicator is set to `PCA_MODE_ACTIVE`.

6.4.6.3.3 Decoding of double quaternion

The reconstructed double quaternion is denoted $(\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2)$ is decoded based on the two indices of 18 and 19 bits read from the SBA bitstream.

The notations from the encoder (see clause 5.4.6.8.3) are reused. The decoding of a unit quaternion, $\hat{\boldsymbol{q}} = (\hat{a}, \hat{b}, \hat{c}, \hat{d})$, is done sequentially by spherical coordinates $(1, \hat{\phi}_1, \hat{\phi}_2, \hat{\phi}_3)$. The index i is found by table lookup using the search :

$$\mathrm{i} = \max\{l \in \{0, \dots, N_1 - 1\} |\ index < offset_1(l+1)\} \tag{6.4-29}$$

Then, the angle ϕ_1 is reconstructed as follows:

$$\hat{\phi}_1(i) = \frac{i}{N_1 - 1}\pi \tag{6.4-30}$$

The number of levels used to quantize ϕ_2 is $N_2(i)$. The index j of angle ϕ_2 is found after updating $index \leftarrow index - offset_1(i)$ by table lookup:

$$j = \max\{l \in \{0, \dots, N_2(i) - 1\} |\ index < offset_2(i, l+1)\} \tag{6.4-31}$$

Then, the angle ϕ_2 is reconstructed as follows:

$$\hat{\phi}_2(i,j) = \begin{cases} \frac{j}{N_2(i)-1}\pi & N_2(i) > 1 \\ 0 & N_2(i) = 1 \end{cases} \tag{6.4-32}$$

The number of levels used to quantize ϕ_3 is $N_3(i,j)$. The index k of angle ϕ_3 is given by:

$$k = index - offset_2(i,j) \tag{6.4-33}$$

Then, the angle ϕ_3 is reconstructed as follows:

$$\hat{\phi}_3(i,j,k) = \begin{cases} \frac{k}{N_3(i,j)}2\pi & N_3(i,j) > 1 \\ 0 & N_3(i,j) = 1 \end{cases} \tag{6.4-34}$$

The unit quaternion is converted to Cartesian coordinates as follows:

$$\begin{cases} \hat{a} = \cos\hat{\phi}_1(i) \\ \hat{b} = \sin\hat{\phi}_1(i)\cos\hat{\phi}_2(i,j) \\ \hat{c} = \sin\hat{\phi}_1(i)\sin\hat{\phi}_2(i,j)\cos\hat{\phi}_3(i,j,k) \\ \hat{d} = \sin\phi_1\sin\hat{\phi}_2(i,j)\sin\hat{\phi}_3(i,j,k) \end{cases} \tag{6.4-35}$$

6.4.6.3.4 Interpolation of double quaternions, conversion to rotation matrices and PCA matrixing

Same as clause 5.4.6.9 where the matrix $\boldsymbol{V}^{[k]}$ is replaced by its transpose in the PCA matrixing.

6.4.6.3.5 Decoder state update

The quantized quaternions $\widehat{\boldsymbol{q}}_{\boldsymbol{1}}^{prev}$ and $\widehat{\boldsymbol{q}}_{\boldsymbol{2}}^{prev}$ in the previous frame are replaced by $\widehat{\boldsymbol{q}}_1$ and $\widehat{\boldsymbol{q}}_2$. The PCA by-pass decision in the current frame is also saved for the next frame.

6.4.6.4 SPAR upmix processing

6.4.6.4.1 General

The SPAR upmixing process reconstructs the N_{spar_ana} channel signal from N_{dmx} downmix signal, where $1 \leq N_{dmx} \leq 4,\ N_{spar\_ana} \geq N_{dmx}$, as follows. First, the primary channel of the downmix signal is provided as an input to a time domain decorrelator block which generates $N_{spar\_ana} - 1$ slow-fluctuating decorrelated channels by applying ducking gains and respective all-pass filters to the input to decorrelator as described in clause 6.4.6.4.2. Then, the downmix signal and the slow-fluctuating decorrelated signal is converted to filterbank domain using CLDFB analysis. Then, the SPAR upmix matrix elements, comprising wet coefficients (or P matrix as given in clause 6.4.5.2) and dry coefficients (or C matrix as given in clause 6.4.5.2), are mapped to CLDFB filterbank banding. The mapped wet coefficients are applied to the decorrelator output to create the wet upmix signal and the mapped dry coefficients are applied to the downmix signal to create the dry upmix signal. The slow-fluctuating wet upmix signal is then added to the dry upmix signal, which contains a transient component, to reconstruct the SBA signal with all FOA channels and selected HOA channels as given in Table 5.4-2.

6.4.6.4.2 Time domain decorrelator

The time-domain decorrelator used for SPAR upmix processing is described in clause 6.2.1.3. For bitrates 24.4 kbps and above, the decorrelator is configured with parameter α_{duck} set to 2.0 whereas, for bitrates below 24.4kbps, α_{duck} is set to 3.0.

6.4.6.4.3 CLDFB analysis

The decoded time domain transport channels u_d are transformed into the CLDFB domain *U* as:

$$U_l^d(k) = \sum_{n=0}^{N-1} p(n) u_d(kS - n) \exp\left(j \frac{\pi}{L}(l + 0.5)(n - \frac{D}{2})\right) \quad (6.4\text{-}25)$$

Using the following notation:

$l = 0, 1, \ldots, L - 1$ is the CLDFB frequency band index where $L = 60$
$N = 600$ is the length of the FIR prototype filter *p*.
$S = 60$ is the processing stride in samples.
k is the CLDFB domain time slot index.
n is the input time sample index.
D = 299 is the analysis-synthesis (sample-by-sample) delay in samples.
d = 0,1,2,3 is the transport channel index.

6.4.6.4.4 CLDFB upmixing

The number of transport channels depends on the target bitrate. In case of one transport channel, a modified version of the FOA W signal is transmitted. Otherwise, the first transport channel corresponds to the unmodified W signal. Other transport channels correspond to residuals with respect to the FOA signals X, Z, Y. Missing transport channels will be approximated by running the decoded W signal through a time domain decorrelator as described in clause 6.2.1.3.

Based on the transport channels and transmitted metadata, FOA signals will be reconstructed.

The metadata time resolution depends on the SPAR frequency band. For SPAR bands 0, 1, …, 7 SPAR metadata at 20 ms time resolution (one frame) is transmitted. For SPAR bands 8, 9, 10, 11, DirAC metadata at 5 ms time resolution is transmitted. However, to reconstruct FOA signals from transport channels the DirAC metadata is subsampled to 20 ms

time resolution similarly at the encoder and the decoder allowing for waveform reconstruction. For missing transport channels and SPAR bands 8, 9, 10, 11, parametric reconstruction works at 5 ms time resolution using DirAC metadata.

Therefore, the reconstruction of FOA signals is described in terms of 5 ms subframes corresponding to 4 CLDFB time slots each and associated upmix matrices. If the effective metadata time resolution is 20 ms, then the associated subframe upmix matrices are held constant over the period of one frame (4 subframes).

For the reconstruction of the FOA signals, the 4 transport channels (or its replacements from the decorrelator processing) are run through the CLDFB analysis resulting in 4 complex-valued signals $U_l^{in\_ch}$, each consisting of 60 frequency bands and 16 time slots for one audio frame. Each frame is reconstructed based on one previous frame and 4 current frame upmix-matrices which are derived from the transmitted SPAR and DirAC metadata as described in clause 6.4.5. The subframe metadata sets are stored in a delay line.

The FOA signals $V_l^{out\_ch}$ are computed as

$$V_l^{out\_ch}(k) = \sum_{in\_ch=0}^{3} U_l^{in\_ch}(k) M_{out_{ch},in_{ch},l}^{CLDFB}(k). \quad (6.4\text{-}26)$$

The upmix matrix $M_{out_{ch},in_{ch},l}^{CLDFB}$ in the CLDFB domain is computed as

$$M_{out_{ch},in_{ch},l}^{CLDFB}(k) = \sum_{b=0}^{11} H_l^b \left((1 - G(k)_{out_{ch}}) M_{out_{ch},in_{ch},b}^{sf(k)-mdi_{out_{ch}}} + G(k)_{out_{ch}} M_{out_{ch},in_{ch},b}^{sf(k)} \right) \quad (6.4\text{-}27)$$

where the subframe index *sf* is computed as $sf(k) = INT(k/4)$.

The helper bool variable *isResChan* indicates if a certain output channel has an associated residual transport channel. It is defined as a function of number of transport channels (which depends on bitrate) as shown in Table 6.4-1.

The metadata update interval *mdi* is given as

$$mdi_{out_{ch}} = \begin{cases} 4 & \text{if } isResChan(out_{ch}) = 1 \\ 1 & \text{if } isResChan(out_{ch}) = 0 \end{cases} \quad (6.4\text{-}28)$$

Table 6.4-1: Bool variable isResChan

#Transport Channels	Output channel			
	0 (W)	1 (Y)	2 (Z)	3 (X)
1	0	0	0	0
2	0	1	0	0
3	0	1	0	1
4	0	1	1	1

Note, that the indication of residual transport channels dependent on the number of transport channels. 1 indicates residual transport channel. 0 indicates not a residual transport channel. Note also that W is always a transport channel but not a residual transport channel.

The cross-fading weights $G(k)_{out_{ch}}$ are defined in Table 6.4-2, depending on the bool variable *IsResChan*. Details on how the cross-fading weights are computed are given in clause 6.4.6.4.6.

Table 6.4-2: Cross-fading weights G(k) used to interpolate between subframe upmix matrices

Time slot index *k*	isResChan	
	0	1
0	0.00	0.00
1	0.25	0.00
2	0.50	0.00
3	0.75	0.00
4	0.00	0.00
5	0.25	0.00
6	0.50	0.00
7	0.75	0.00
8	0.00	0.00
9	0.25	0.00
10	0.50	0.00
11	0.75	0.00
12	0.00	0.16102859
13	0.25	0.65157765
14	0.50	1.00
15	0.75	1.00

The SPAR band contribution weights H_l^b to a particular CLDFB band are obtained through table lookup. They are illustrated in Figure 6.4-7. Details how the table values are derived are given in clause 6.4.6.4.5.

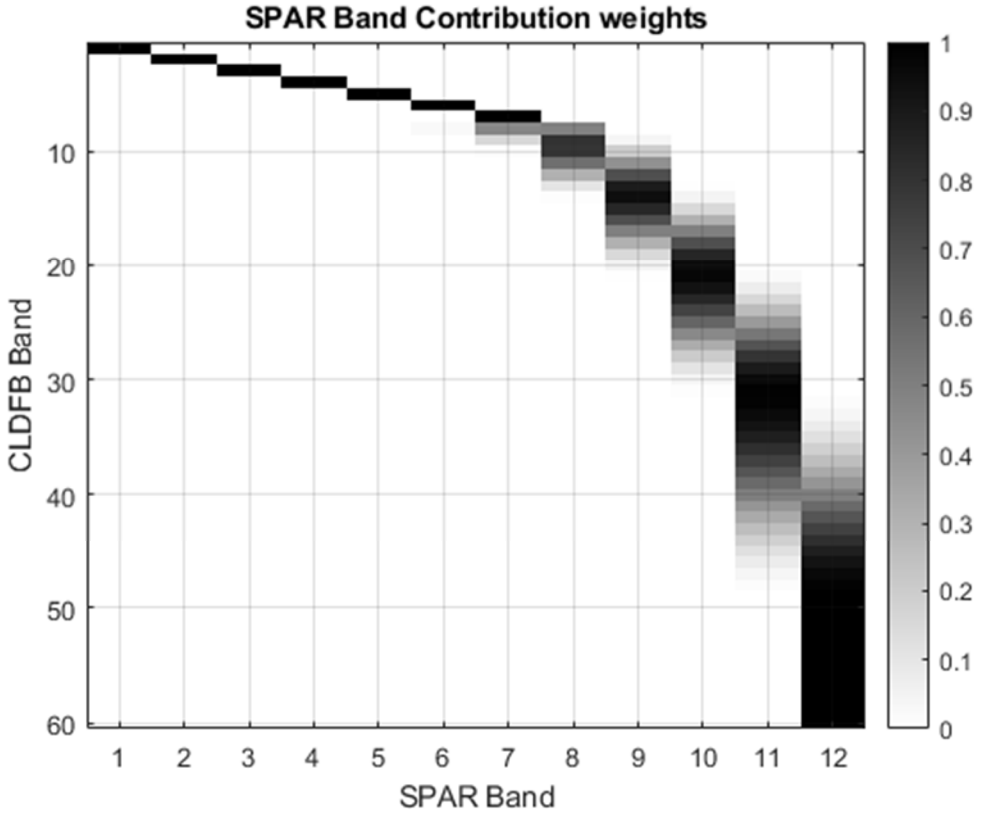


Figure 6.4-7: SPAR Band contributions H_l^b

6.4.6.4.5 MDFT filterbank banded upmix matrix to CLDFB banded upmix matrix conversion

The upmix matrix used to reconstruct one FOA signal from the decoded signals, for a particular CLDFB band is derived as a linear combination of upmix matrices corresponding to contributions from multiple SPAR bands. This is due to the overlapping nature of the SPAR filterbank bands and since the SPAR filterbank synthesis consists of the summation over all SPAR filter outputs. Ideally, for best waveform reconstruction, the SPAR filters would be run in the CLDFB domain as complex multi-tap FIR filters to best match the encoder processing. Since the computational complexity of

running multi-tap filters in the CLDFB domain is high, the SPAR filters are approximated by real-valued single tap filters H_l^b for CLDFB band l as follows.

$$H_l^b = \frac{M_l^b}{\sum_{b=0}^{11} M_l^b} \tag{6.4-29}$$

with

$$M_l^b = \sqrt{\sum_{f=lN/L}^{(l+1)N/L} \left| \sum_{n=0}^{N_h-1} h_b(n) e^{-j2\pi(f+0.5)n/N} \right|^2}. \tag{6.4-30}$$

In the above equation (6.4-30), h_b is the SPAR FIR filter for band b of length N_h samples, N is the size of the MDFT of the SPAR FIR filter (960, 640 or 320 for sampling frequencies of, respectively, 48 kHz, 32 kHz or 16 kHz) and L is the number CLDFB bands (60).

For certain CLDFB bands it is observed that the frequency responses of SPAR Filters and CLDFB bands match sufficiently well. This is the case for the first 7 CLDFB bands such that only single SPAR bands contribute to CLDFB bands

$$M_l^b = \begin{cases} 0 & \text{if } b = l \\ 1 & \text{if } b \neq l \end{cases} \quad l = 0,\ldots,6. \tag{6.4-31}$$

6.4.6.4.6 Crossfading in CLDFB

For transmitted residual transport channels, the goal of the decoder is to perform optimal waveform reconstruction based on the transmitted metadata given the known encoder prediction (residual) processing using the SPAR (encoder) filterbank and time cross-fade between frames as described in clause 5.4.5.3. Theoretically the encoder processing can be reversed perfectly at the decoder side if the same SPAR encoder filterbank and time domain cross-fade were applied. However, due to system and complexity reasons, the encoder operation is reversed in the CLDFB domain and cross-fading between frames in the time domain with the cross-fade gain as shown in Figure 5.4-11 is performed by cross-fading the corresponding CLDFB domain parameters. The cross-fading weights $G(k)$ for that operation are given in the third column in Table 6.4-2. These weights were derived such that after CLDFB synthesis they optimally (in a least mean squares sense) approximate the fading operation in the SPAR encoder. More specifically, the broad band gains for all CLDFB time slots in an audio frame as shown in Table 6.4-2 are computed such that, using a pseudo-random noise input, the total squared difference between the target time domain cross-faded signal and the signal obtained by CLDFB analysis, gain application and subsequent CLDFB synthesis is minimized. The optimization is defined by the training noise input signal, the target time domain cross-fade and the CLDFB analysis and synthesis prototype filters. Rather than cross-fading time domain signals after CLDFB synthesis the corresponding reconstruction parameters are cross-faded using the tabulated broad band gain values in the CLDFB domain.

6.4.6.4.7 Additional VLBR smoothing

For FOA output at the two lowest IVAS bitrates, 13.2 and 16.4 kbps, an additional adaptive smoothing is applied to the CLDFB-domain upmix matrix according to clauses 6.4.5.3 and 6.4.6.4.4.

This method improves the perceptual quality by smoothing the elements of the mixing matrix in equation (6.4-30) less strongly when a fast change in signal energy is detected. Conversely, stronger smoothing is applied when the signal energy changes more slowly.

A detailed description of the method and the formulas for the smoothing factor are given clause 6.4.6.5.8.2.4 for the case of stereo output.

In case of FOA outbut at these bitrates, the smoothing factor β is evaluated every 5 ms subframe. Consequently, $N_{\text{long}} = 10$ subframes then corresponds to 500 ms and $N_{\text{short}} = 3$ subframes to 15 ms.

6.4.6.5 DirAC synthesis and rendering

6.4.6.5.1 Overview

At the decoder side, the SPAR upmix according to clause 6.4.6.4 produces a larger second set of transport channels from a smaller first set of transport channels, which are decoded using the core-decoder tools according to clause 6.2.4. The channels that comprise this second set are listed in the fourth column of Table 5.4-2 as a function of the total IVAS

bitrate. The numbers refer to the real spherical harmonics (spatial basis functions) in ACN order. Figure 6.4-8 shows sketches of these functions.

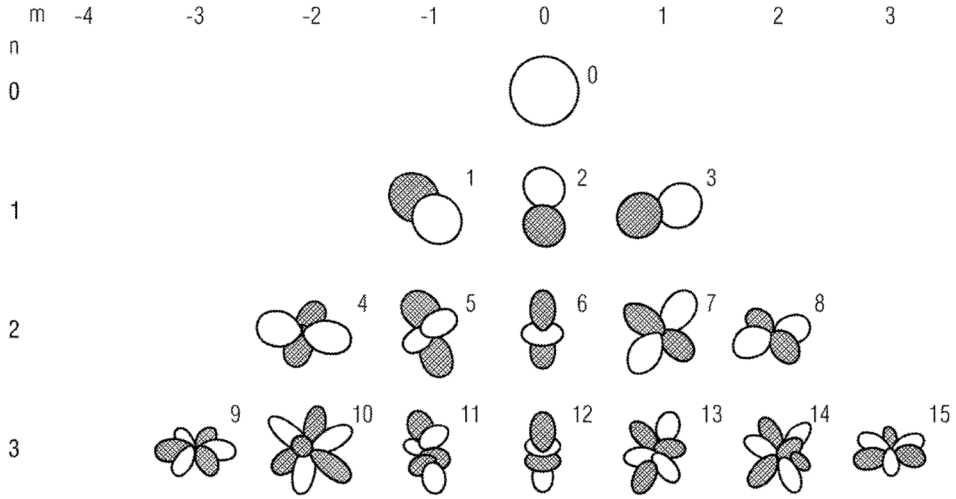


Figure 6.4-8: Basis functions of the ambisonic components up to third order. Shaded portions represent regions where the polarity is inverted.

These channels are obtained directly in the domain of the CLDFB, and the synthesis of the output signal is performed in the same domain, thereby avoiding the need for extra synthesis and analysis steps.

The remaining ambisonics channels are then reconstructed (synthesized) using the DirAC method.

6.4.6.5.2 Mode selection

For the two highest bitrates (384 kbps and 512 kbps), the high-order mode of the DirAC decoder is activated. It is based on higher-order directional audio coding (HO-DirAC). In this mode, two sector decoding paths are activated. For lower bitrates, the low-order mode (first-order DirAC) is activated. Then only one sector decoding path is activated and the other one is deactivated.

6.4.6.5.3 High-order operation mode

6.4.6.5.3.1 General

In the high-order operation mode, the IVAS SBA decoder unit processes multiple directional sector signals and a global diffuse signal. It generates a decompressed ambisonics spatial audio signal representation from a compressed ambisonics spatial audio signal representation, which includes 4 or more transport channels (see above) and side information.

The side information includes sound field parameters: directional parameters providing information on a direction of arrival in each spatial sector, sector diffuseness parameter for each spatial sector, and a global diffuseness parameter.

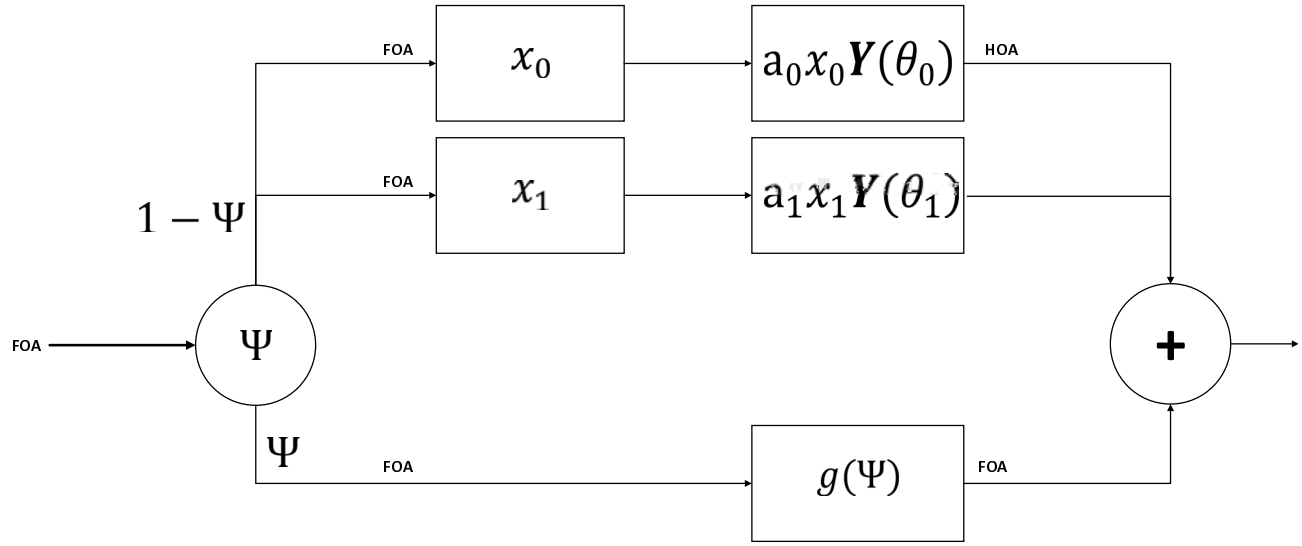


Figure 6.4-9: Block diagram of the DirAC decoder in the high-order operation mode

6.4.6.5.3.2 Directional sector decoding paths

A set of sector decoding paths, one for each spatial sector, decodes the directional sector signals into the decompressed ambisonic signal representation (see figure 6.4-9). In IVAS, two spatial sectors are used. Each path derives the corresponding sector signal from the transport channels by means of spatial filtering. The directional (DoA) parameters and sector diffuseness parameters are applied to weight and pan this signal into the ambisonic output stream:

$$\boldsymbol{S}_{hoa,s}^{cldfb} = a_s \boldsymbol{Y}(\theta_s) s_s = a_s [s_s Y_{00},\ s_s Y_{1-1}, s_s Y, s_s Y_{11}, \dots] \tag{6.4-32}$$

where s_s is the scalar-valued sector pressure signal. It results from the inner product of the beamforming matrix element of the spatial sector s, $\boldsymbol{w}_s^T$, with the FOA signal $\boldsymbol{s}^{cldfb}{}_{umx,foa}(m,k)$:

$$s_s(m,k) = \boldsymbol{b_s} \cdot \boldsymbol{S}_{umx,foa}^{cldfb}(m,k) \tag{6.4-33}$$

$\boldsymbol{S}_{umx,foa}^{cldfb}(m,k)$ is the vector of the FOA channels in the second set of transport channels in the CLDFB domain, and $\boldsymbol{b_s}$ is the vector of the beamformer weights. The matrix can be understood as the projector on this vector, i.e., $\boldsymbol{w}_s^T = \boldsymbol{b_s} \boldsymbol{b_s}^T$.

In IVAS, the two sector pressure signals are effectively calculated as

$$x_0 = 1.772454\, S_{umx,foa,w}^{cldfb} + 1.023327\, S_{umx,foa,x}^{cldfb}\ , \tag{6.4-34}$$

$$x_1 = 1.772454\, S_{umx,foa,w}^{cldfb} - 1.023327\, S_{umx,foa,x}^{cldfb}\ , \tag{6.4-35}$$

$\boldsymbol{Y}(\theta_s)$ is the vector of the real spherical harmonics up to the configured output order evaluated θ_s, the DoA angle of the sector s. a_s is the relative directionality of the spatial sector s defined in clause 5.4.3.4.2. Thereby the sound field components described by the higher-order (non-planar) ambisonics channels are obtained by evaluating the second set of transport channels (direct sound signals) the SPAR upmix (reference sound signals) and the above response functions.

This panning is performed for each time-frequency tile of the CLDFB, which are grouped into the parameter bands according to Table 5.4-5 and subframes of 5 ms length. For each group of time-frequency tiles, the real spherical harmonic functions (response functions) $\boldsymbol{Y}(\theta_s)$ are calculated from the sound direction (DoA) in the sector s.

In order to avoid highly complex calculations of trigonometric functions and Legendre polynomials, the spherical harmonics (spatial basis functions) are evaluated by table look up. The tables are indexed by the degree and index l and m, and the azimuth and elevation angles corresponding to the sound direction of arrival (DoA) parameters of the acoustic model.

6.4.6.5.3.3 Global diffuseness decoding

The global diffuseness decoding path generates the global diffuseness signal. This is achieved by applying the global diffuseness parameter Ψ to the second set of transport channels from the SPAR upmix. Specifically, the subset of the

transport channels that represents the first order of the ambisonic output signal is amplified to generate the correct amount of diffuse energy.

All diffuse energy is exclusively contained in the first ambisonics order. The directional paths synthesize only directional components into higher ambisonics orders. To compensate for the absence of diffuse energy in the directional path (see Figure 6.4-9), an adaptive gain is applied according to the formula

$$g(\Psi) = \sqrt{1 + \Psi * (f_{comp} - 1)} \ , \tag{6.4-36}$$

with the diffuse compensation factor f_{comp}. With this gain, the correct ratio between directional and diffuse energy in the higher-order ambisonics output signal is restored.

f_{comp} is derived from the energies of the sound components in the first and second group (ambisonics channels of first and higher orders). Specifically, the energy in the first group (low-order channels) is given by

$$E_L = \sum_{l=0..L} \frac{1}{\sqrt{2*l+1}}, \tag{6.4-37}$$

Analogously that in the second group (high-order channels) is

$$E_H = \sum_{l=0..H} \frac{1}{\sqrt{2*l+1}}, \tag{6.4-38}$$

where H is the ambisonics order of the output signal and L that of the second set of transport channels (SPAR upmix). The diffuse compensation factor is then calculated as

$$f_{comp} = \frac{E_H}{E_L}, \tag{6.4-39}$$

This means that the gain increases when the output order H (number of components in the second group) increases and decreases when the input order L increases (number of sound field components in the first group). It also increases when the diffuses increases. Therefore, the diffuse energy missing in the reconstructed higher-order channels is compensated for in the FOA channels.

In case the first set of transport channels, $\boldsymbol{s}_{dmx}$, does not contain all channels of the first group of sound field components, $\boldsymbol{S}_{umx,foa}$, the SPAR decoder generates these sound field components by upmixing from the first set of transport channels. This involves forming weighted combinations of the input channels decoded with MCT and the decorrelator channels.

To obtain a perceptually faithful reproduction of the diffuse sound components in the scene, it is required to have the necessary diffuse signals in the first set of transport channels. Therefore, additional channels are generated to complement this first set (the component channels of $\boldsymbol{s}_{dmx}$) when the number of channels in the second set (the component channels of $\boldsymbol{S}_{umx,foa}$) is greater than in the first. Specifically, decorrelation is applied to the channels in the first set. These decorrelated channels are then mixed up to the second set of channels in SPAR (cf. clause 6.4.6.4).

6.4.6.5.3.4 Signal inserter

The global diffuseness signal is then inserted into the output ambisonic output stream $\boldsymbol{S}_{HOA}^{cldfb}$ together with the directional signals, obtaining the final higher-order ambisonics output signal

$$\boldsymbol{S}_{hoa}^{cldfb} = (1 - \Psi) \sum_s \boldsymbol{S}_{hoa,s}^{cldfb} + g(\Psi) \boldsymbol{S}_{umx,foa}^{cldfb}. \tag{6.4-40}$$

The addition of the higher-order signal $\boldsymbol{S}_{hoa,s}^{cldfb}$ and the first-order signal $\boldsymbol{S}_{umx,foa}^{cldfb}$ is understood such that the FOA components come from the latter and the other components of the HOA signal from the former.

Together with Equation (6.4-33) this becomes

$$\boldsymbol{S}_{hoa}^{cldfb} = \sum_s \boldsymbol{g}_{\boldsymbol{s}}^{dir}(\theta_s)\, s_s + g(\Psi) \boldsymbol{S}_{umx,foa}^{cldfb} \tag{6.4-41}$$

with the directional gain $\boldsymbol{g}_{\boldsymbol{s}}^{dir}(\theta_s) = (1 - \Psi) a_s \boldsymbol{Y}(\theta_s)$.

In other words, the SBA decoder generates two groups of sound field components. The spatial basis functions in this representation are the real spherical harmonics, which are pair-wise orthogonal. In particular, any spatial basis function from the first group is orthogonal to any one from the second group.

The first group includes the channels of the first ambisonics order. It contains both direct and diffuse sound field components. The second group comprises the channels of the higher ambisonics orders 2 and 3. It contains only direct sound field components. To mitigate the absence of diffuse sound field components in the second group, an energy compensation is performed based on the diffuseness parameter and the ambisonics order in the first (L) and second (H) group.

6.4.6.5.3.5 Gain smoothing

In order to avoid artifacts, a gain smoothing is applied to the gains Equations (6.4-40) and (6.4-41). Specifically, and onset filter and an interpolation between the gains of two subsequent subframes over the time slots of the CLDFB filter bank are employed.

The onset filter controls the weight of the gain of the previous subframe

$$w_1 = 0.3679 + \text{onset\_filter} * (0.1175 - 0.3679) \quad (6.4\text{-}42)$$

and that of the current subframe

$$w_2 = 1 - w_1. \quad (6.4\text{-}43)$$

onset_filter is equal to 1 during normal operation of the codec and, hence, the update rate is very slow. When an onset is detected, the weight of the current gains is increased, leading to a faster update and avoid a smearing out of the onset. The onset filter and the gain weights are evaluated individually for each band of the CLDFB

The onset detection is based on the signal energy. Specifically, the ratio between the minimum and the maximum of the envelope is computed. In each frame the maximum and minimum values are updates as

$$P_{\max} \leftarrow \max(P_{\max} * \alpha, P),$$

$$P_{\min} \leftarrow \beta P_{\min} + (1 - \beta) P_{\max},$$

$$P_{\min} \leftarrow \min(P_{\min}, P_{\max}),$$

and

$$\text{onset\_filter} = * \frac{P_{\min}}{P_{\max} + \epsilon}.$$

Then, onset_filter is constrained to values between 0 and 1. α has a fixed value of 0.95, β has a fixed value of 0.995, and g_{onset} has a fixed value of 4.0.

Additionally, the smoothened gains, $\boldsymbol{g}_s^{dir}(\theta_s)$ and $g(\Psi)$ in Equation (6.4-41), are limited to a range value range between a fixed minimum threshold of 0.99 and a fixed maximum threshold of 2.00. Therefore, too strong deviations from the original signal characteristics of $\boldsymbol{S}_{umx,foa}^{cldfb}$ are avoided.

The filtered gains are then applied to the signals in each time slot of the CLDFB. As length of this time slot is 1.25 ms but the resolution of the DoA and diffuseness information is 5 ms (one subframe), the gains are linearly interpolated over the time slots in a subframe.

6.4.6.5.4 Low-order operation mode

6.4.6.5.4.1 Overview

In the low-order operation mode, one sector decoding path is deactivated and only one is activated. Hence, the block diagram of the DirAC decoder simplifies to that in right hand-side of Figure 6.4-9. In the left-hand side of Figure 6.4-10, additional processing is described that is required to save on the metadata bitrate. Specifically, a hybrid decoding is employed based on a split of the processing in accordance with the core-coder (cf. clause 5.4.3.5).

6.4.6.5.4.2 Hybrid decoding

6.4.6.5.4.2.1 General concept

The hybrid decoder processes two different encoded representations of two portions of the input ambisonics signal. The first portion comprises the low frequencies in the range f^{SPAR} (the 8 lower parameter bands in Table 5.4-5) and f^{DirAC} (the 4 higher parameter bands in Table 5.4-5).

The encoded representation of the first portion of the first set of transport channels, $\boldsymbol{s}_{dmx}(t)$, is decoded in a wave-form preserving way by the core-decoder up to a crossover frequency (cf. clauses 5.2.2 and 6.2.2). The crossover frequency for the BWE depends on the bitrate and can be found in Tables 5.2-18 and 5.2-19. This first portion is fed into the SPAR upmixer (cf. clause 6.4.6.4) to obtain the vector of the components of the first-order ambisonics signal, $\boldsymbol{S}_{umx,foa}^{cldfb}$. From this signal, the sound field parameters (DirAC) can be accurately re-estimated on the decoder side. Hence, no sound field parameters (DirAC) are received from the input interface (bitstream) for this portion.

The encoded representation of the second portion of the first set of transport channels, $\boldsymbol{s}_{dmx}(t)$, is decoded using parametric methods (cf. clause 6.2.3). Hence, an accurate re-estimation of the sound field parameters (DirAC) on the decoder side would not be guaranteed. Therefore, the sound field parameters are received from the bitstream by the input interface. The DirAC metadata decoder decodes the DirAC parameters (DoA and diffuseness for the 0th sector) for the second portion (high frequency bands). The upmix to is computed based on the SPAR (low frequency bands) and DirAC (high frequency bands) parameters.

This hybrid decoding method avoids the redundant transmission of SPAR and DirAC parameters for the low parameter bands, while the transmission of the SPAR parameters can be avoided via the use of the model covariance according to clauses 5.4.3.8.3.1and 6.4.4.3. Therefore, the total amount of metadata bits is reduced.

For each CLDFB time-frequency tile in the low-frequency bands (first portion), a spatial parameter estimator (input signal analyzer), including a direction determiner and a diffuseness estimator, then obtains the sound field parameters (including direction and diffuseness) from $S_{umx,foa}^{cldfb}$ (second set of transport channels). The necessary conversion of the sound field components into the time-frequency representation is already performed prior to the SPAR upmix. The time and frequency resolution of the estimated DirAC parameters is that of the CLDFB itself. No parameter grouping or quantization is applied for the parameters estimated in the decoder.

The encoded representation of the second portion of the first set transport channels is decoded using the parametric DirAC method and upmixed to obtain the high-frequency portion of $\boldsymbol{S}_{umx,foa}^{cldfb}$. The DirAC metadata decoder decodes the DirAC parameters (DoA and diffuseness for the 0th sector) for the second portion (high frequency bands). The DoA and diffuseness values corresponding to the parameter bands transmitted are expanded to the TF tiles that fall into these groups and subframes. The upmix to $S_{umx,foa}^{cldfb}$ is computed based on the SPAR (low frequencies) and DirAC (high frequencies) parameters.

6.4.6.5.4.2.2 Processing steps

A spatial renderer (see Figure 6.4-10) then generates a sound field description in the form of higher-order ambisonics channels from components of the sound field represented by the first order ambisonics channels (second larger set of transport channels) from the SPAR upmix, $\boldsymbol{S}_{umx,foa}^{cldfb}$. After the upmixing, the signal is present in the CLDFB domain. For the low frequencies (first) portion, the DirAC parameters are derived from the parameter estimator. For the high-frequencies (second portion), it uses the DirAC parameters, sound direction (DoA) and diffuseness, decoded from the bitstream.

The spatial rendering to higher-order ambisonics is described in details in the following. It consists of the generation the high-order components above the first-order components, which inserts only the directional contribution. The diffuseness contribution is not generated for higher orders, but only its energy is compensated by scaling the first-order components.

As stated above, the first-order components comprise channels from the first and second set of transport channels. The first set (the component channels of $\boldsymbol{S}_{dmx}$) is directly derived directly from coded and transmitted transport channels. The second set (the component channels of $\boldsymbol{S}_{umx}$) is generated based on SPAR parameters and DirAC DoA and diffuseness parameters. It is a complete is first-order representation of the sound field, comprising both directional and diffuse contributions.

6.4.6.5.4.3 Directional sound field decoding

The directional sound field components of the higher ambisonics orders are reconstructed via the panning using the direction of arrival according to Equation (6.4-42). However, there is only one sector pressure signal. As this one sector spans the whole sphere, the sector pressure signal x_0 is equal to the w component of the first-order ambisonics signal from the first set of transport channels, $S_{umx,foa,w}^{cldfb}$.

Hence, the sound field components described by the higher-order (non-planar) ambisonics channels are obtained by evaluating the direct sound signals (reference sound signals) and the response functions (spherical harmonics). The reference sound signals are the component channels of $\boldsymbol{S}_{umx,foa}^{cldfb}$. The response functions are obtained via table look up.

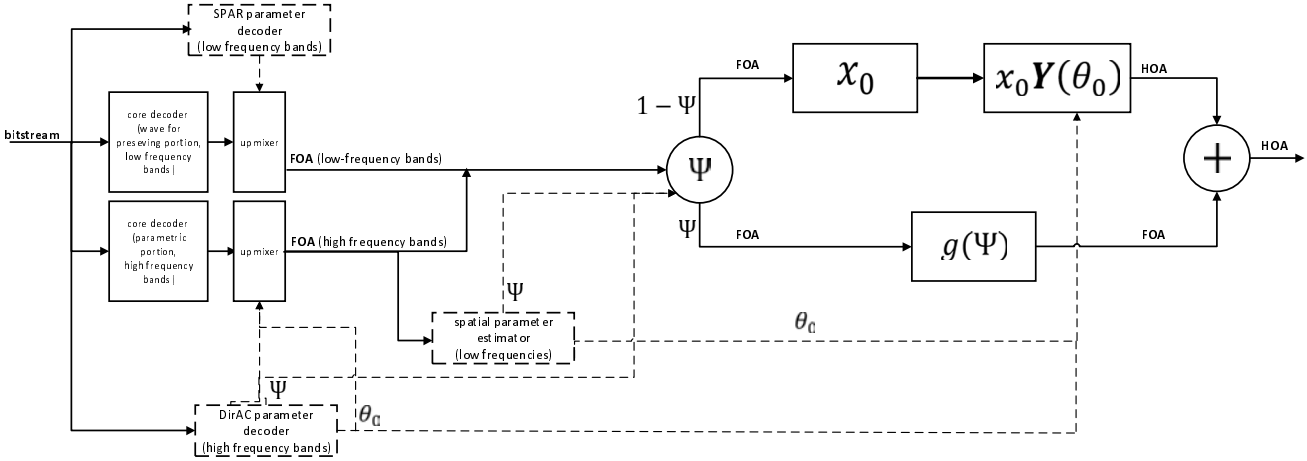


Figure 6.4-10: Block diagram of the DirAC decoder in the low-order mode

6.4.6.5.4.4 Signal inserter

The directional and diffuseness paths are then added by a signal inserter, which sums up the directional and diffuse components to obtain the final output signal at the configured output order in the spatial basis of the real spherical harmonics. The block diagram of the DirAC decoder in the low-order mode is depicted in Figure 6.4-3.

The weights of the directional and diffuse paths are, in this case, calculated as $\sqrt{1-\Psi}$ and $\sqrt{\Psi}$, respectively. Therefore, the vectors of the expansion coefficients of the sound field in the spatial basis of spherical harmonics is given by

$$\boldsymbol{S}_{hoa}^{cldfb} = \sqrt{(1-\Psi)}\,\boldsymbol{S}_{hoa,0}^{cldfb} + \boldsymbol{S}_{foa}^{diff} \tag{6.4-44}$$

with the directional sector signal $\boldsymbol{S}_{hoa,0}^{cldfb}$ and the diffuse signal component $\boldsymbol{S}_{foa}^{diff}$. Again, the addition of the higher-order signal $\boldsymbol{S}_{hoa,s}^{cldfb}$ and the first-order signal $\boldsymbol{S}_{foa}^{diff}$ is understood such that the FOA components come from the latter and the other components of the HOA signal from the former.

Together with Equation (6.4-32) this becomes

$$\boldsymbol{S}_{hoa}^{cldfb} = \boldsymbol{g}_0^{dir}(\theta_0)\, s_0 + \boldsymbol{S}_{foa}^{diff} \tag{6.4-45}$$

with the directional gain $\boldsymbol{g}_0^{dir}(\theta_s) = \sqrt{1-\Psi}a_0\boldsymbol{Y}(\theta_0)$. In this case with one sector decoding path enabled, $a_0 = 1$.

6.4.6.5.4.5 Diffuseness decoding path

The diffuse signal component $\boldsymbol{S}_{foa}^{diff}$ is computed as

$$\boldsymbol{S}_{foa}^{diff} = \begin{bmatrix} g_w(\Psi)\, S_{umx,foa,w}^{cldfb} \\ g_x(\Psi) S_{umx,foa,x}^{cldfb} \\ g_y(\Psi)\, S_{umx,foa,y}^{cldfb} \\ g_z(\Psi)\, S_{umx,foa,z}^{cldfb} \end{bmatrix} \tag{6.4-46}$$

The diffuse gains are calculated as

$$g_w(\Psi) = \sqrt{1 + \Psi * (f_{comp} - 1)}, \quad (6.4\text{-}47)$$

and

$$g_{x;y;z}(\Psi) = \sqrt{\Psi\, c + (1 - \Psi) Y_{x;y;z}^2(\theta_0) b_{x;y;z}}, \quad (6.4\text{-}48)$$

The variables b and c are defined as $b_{x;y;z} = \frac{E}{E_{x;y;z}+\epsilon}$ and $c = 1 + \frac{1}{2}(f_{comp} - 1)$. E is the signal energy of all channels in $\boldsymbol{S}_{umx,foa}^{cldfb}$ in the current frame and $E_{x;y;z}$ that of the respective channel x, y, or z.

6.4.6.5.4.6 Gain smoothing

To the gains $\boldsymbol{g}_0(\theta_0)$ and $g_{w;x;y;z}(\Psi)$ gain smoothing is applied in the same way as in the high-order operation mode.

Additionally, the smoothed gains are constrained to the value range between 0.85 and 1.15. Therefore, too strong deviations from the original signal characteristics of $\boldsymbol{S}_{umx,foa}^{cldfb}$ are avoided.

6.4.6.5.4.7 Summary

In summary, the DirAC decoder calculates the sound field components described by the higher-order (non-planar) ambisonics channels by evaluating the direct and diffuse sound signals from the SPAR upmix (reference sound signals) and the above response functions.

6.4.6.5.5 Bitrate switching

The DirAC decoding unit can operate at different bitrates and dynamically switch between them in case of varying bandwidth of the connection. It supports two principal operation modes and can switch between them: low-order operation mode and a high-order operation mode.

In the low-order operation mode, only one sector decoding path is activated and the other one is deactivated. In this case, the side information does not contain the sound field parameters for the deactivated sector. The active sector then spans the whole sphere.

In the high-order operation mode, all two sector paths are activated, and the side information contains the sound field parameters for both sectors and the global diffuseness parameters.

For the re-configuration of the other processing blocks of the SBA decoder, see clause 6.4.9.

6.4.6.5.6 Binaural rendering

6.4.6.5.6.1 Rendering modes

Binaural rendering of decoded SBA output signals is realized using different methods. These are selected depending on Output configuration and bitrate. The mapping of bitrates to renderers is shown in Table 6.4-3.

Table 6.4-3: Binaural rendering method for SBA depending on the bitrate

Bitrate [kbps]	Binaural	Binaural room
< 96	Parametric renderer	Parametric room renderer
≥ 96	Fastconv	Fastconv with room effect

The parametric renderer directly evaluates the first-order ambisonics signal from second set of transport channels and the DoA and diffuseness parameters. The sector decoding and diffuseness paths according to clauses 6.4.6.5.3 and 6.4.6.5.4 are not used. The parameter estimator (spatial analyzer) is not run. Instead, the DirAC parameters are derived from the SPAR parameters according to clause 6.4.4.2. A detailed description of this renderer can be found in clause 7.2.2.3.

The FastConv based renderers process the third-order ambisonics signal generated by DirAC decoder and computes a convolution with a HRIR in the spherical harmonics domain. A detailed description of this renderer can be found in clause 7.2.2.4.

6.4.6.5.6.2 Scene rotation

When head-tracking is enabled, the sound scene is rotated to account for rotations of the listeners head to prove a realistic VR experience. The rotation of the third-order ambisonics signal resulting from the DirAC synthesis can be achieved by applying the appropriate rotation matrix in the spherical-harmonics domain. This requires, however, a matrix multiplication with a high computational complexity.

The DirAC decoder supports a low-complexity head tracking mode to reduce this complexity. Only to the vector those channels belonging to the lowest ambisonic order that contains transport channels from the second larger set are pre-multiplied by the rotation matrix in the spherical-harmonics domain:

$$\boldsymbol{s}_{umx,L}^{cldfb} \leftarrow R_L \, \boldsymbol{s}_{umx,L}^{cldfb}.$$

$\boldsymbol{s}_{umx,L}^{cldfb}$ is the signal vector $\boldsymbol{s}_{umx}^{cldfb}$ truncated at the order L, which is chosen such that all channels of degree $l \leq L$ are contained, i.e., only the complete ambisonics orders are selected. R_L is the rotation matrix in the spherical-harmonics domain.

The channels belonging to higher ambisonics orders are rotated more efficiently: the parameters of the psychoacoustic model are transformed by applying the rotation matrix to the DoA.

First the direction vector pointing to the source direction is constructed

$$\boldsymbol{r}_{DoA} = \begin{pmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos(\theta) \\ \sin(\theta) \end{pmatrix}.$$

Then the rotation is applied

$$\boldsymbol{r}_{DoA} \leftarrow \boldsymbol{R}\, \boldsymbol{r}_{DoA}.$$

Finally, the vector is converted back to the direction angles:

$$\phi = \operatorname{atan}\left(\frac{r_{DoA,y}}{r_{DoA,x}}\right)$$

and

$$\theta = \operatorname{atan}\left(\frac{\mathrm{r}_{\mathrm{DoA,z}}}{\sqrt{\mathrm{r}_{\mathrm{DoA,x}}^2 + r_{DoA,y}^2}}\right).$$

The DirAC reconstruction then automatically yields the channels of the rotated scene.

6.4.6.5.7 Loudspeaker rendering

6.4.6.5.7.1 Overview

The rendering of the first set of transport channels ($\boldsymbol{s}_{dmx}^{cldfb}$) requires the upmixing to the second set of transport channels ($\boldsymbol{s}_{umx,foa}^{cldfb}$) according to clause 6.4.6.4. This second set is a multi-channel representation of the sound scene in the spatial basis of the spherical harmonics. The channels in this representation are listed in the fourth column of Table 5.4-2.

For loudspeaker output, a conversion of this multi-channel representation into a different output multi-channel representation is required. Specifically, this new multi-channel representation comprises one channel to be output on each loudspeaker of the configured layout. The conversion is achieved by a parameter analyzer and a signal composer.

In accordance with clause 6.4.6.5.4, the spatial analyzer (parameter estimator) derives a parametric representation of the scene comprising DoA and diffuseness information in addition to the downmix channels in $\boldsymbol{s}_{dmx}^{cldfb}$ (first set of transport channels).

The signal composer generates the output multi-channel representation of the spatial audio signal in the domain of the configured loudspeaker layout, as opposed to the domain of the real spherical harmonics in the input representation. Specifically, an upmix is performed from the second set of transport channels to the output loudspeaker layout using the spatial parameters from an intermediate representation, which consists of the transport channels and the spatial parameters.

The method used for the upmixing is selected based on the total IVAS bitrate. Table 6.4-4 shows the conversion method for each bitrate.

Table 6.4-4: Conversion method from ambisonics to loudspeaker domain depending on the bitrate

IVAS total bitrate [kbps]	multi-channel conversion method
< 96	SHD
96 – 256	PSD
384	SHD
512	SHD

The spherical harmonics domain (SHD) synthesis is always performed in the same way as in clause 6.4.6.5.3 or 6.4.6.5.4. The resulting third-order ambisonics signal is then decoded to the configured loudspeaker layout according to clause 6.4.6.5.7.2. The power spectral density (PSD) synthesis directly generates a multi-channel signal in the spatial domain of the output loudspeaker layout according to clause 6.4.6.5.7.3.

6.4.6.5.7.2 ALLRAD decoding

The method SHD is based on a first synthesis to an ambisonics signal of order 3 with subsequent conversion of the multi-channel ambisonics signal to the spatial domain of the configured loudspeaker layout. This conversion is performed using matrix multiplication of a decoder matrix to the multi-channel vector of the ambisonics signal $\boldsymbol{S}_{HOA}^{cldfb}$:

$$\boldsymbol{S}_{ls}^{cldfb}(m,k) = \boldsymbol{D}^{ALLRAD}\boldsymbol{S}_{hoa}^{cldfb}(m,k) \quad . \tag{6.4-49}$$

The decoder matrix $\boldsymbol{D}^{ALLRAD}$ is calculated once during decoder initialization or bitrate switching and stored in memory.

6.4.6.5.7.3 PSD rendering

6.4.6.5.7.3.1 Upmixing

As a first step, the reference power of the first-order ambisonics signal from the SPAR upmix (second set of transport channels), $\boldsymbol{S}_{umx,foa}^{cldfb}$, is calculated according to

$$P^{\text{ref}}(m,k) = \sum_{i=w,x,y,z}\left\|S_{foa,i}^{cldfb}(m,k)\right\|^2. \tag{6.4-50}$$

This is done for each frequency band of the CLDFB denoted by the band index k.

Then $S_{foa,i}^{cldfb}$ decoded to the output loudspeaker layout with a decoding matrix according to Equation (6.4-49).

From this signal, the prototype power is computed according to

$$P^{\text{proto}}(m,k) = \sum_{i=w,x,y,z}\left\|s_{LS,i}^{cldfb}\ (m,k)\right\|^2. \tag{6.4-51}$$

Finally, a per-channel gain is multiplied to the vector of the multi-channel signal $\boldsymbol{S}_{LS}^{cldfb}$, i.e., the components of output multi-channel signal become

$$x_{LS,i}(m,k) \leftarrow g_i(k)x_{LS,i}(\text{m,k}) \tag{6.4-52}$$

for each channel in the loudspeaker domain indexed by i and each CLDFB band indexed by k.

6.4.6.5.7.3.2 Gain calculation

The gains comprise a directional and a diffuse part:

$$g_i(k) = g_i^{\mathrm{dir}}(k) + g_{i,}^{\mathrm{diff}}(k). \quad (6.4\text{-}53)$$

The gains g_i (k) are determined by VBAP panning of the DoA in the CLDFB band k:

$$g_i^{\mathrm{dir}}(k) = \sqrt{P^{\mathrm{proto}}(k) * (1 - \Psi\ \ (k))^2 * P^{\mathrm{ref}}(k)\ * (\ \mathrm{VBAP}(\theta\ \ (k), \boldsymbol{r}_i\ ))^2}$$

The function $\mathrm{VBAP}(\theta\ \ (k), \boldsymbol{r}_i)$ denotes the VBAP panning gains of a signal from the direction $\theta\ \ (k)$ to the loudspeaker at the position $\boldsymbol{r}_i$. $\Psi\ \ (k)$ is the diffuseness in the CLDFB band k . The DoA and diffuseness values is obtained from a direction determiner for low frequency bands, and from the decoding of the bitstream for the high frequency bands (cf. clause 6.4.6.5.4). The bar $^-$ denotes a smoothing over time.

Similarly, the diffuse gains are given by

$$g\ \ (k) = \sqrt{P^{\mathrm{proto}}(k) * \Psi\ \ ^2(k) * P^{\mathrm{ref}}(k) * h_i^2}$$

with the diffuse response function $h_i = \frac{1}{\sqrt{N_{LS}}}$, where N_{LS} is the number of loudspeakers in the configure output layout.

6.4.6.5.8 Stereo and mono output

6.4.6.5.8.1 Overview

This subclause describes an optimized decoding path for stereo output from an encoded SBA input. It is activated when stereo output is requested and there are 1 or 2 channel in the first set of transport channels. This applies to the SBA bitrates 13.2 – 32 kbps and 48 – 80 kbps, respectively.

Specifically, these optimizations reduce the delay and computational complexity. The CLDFB analysis and synthesis steps and the time-domain decorrelation can be avoided. Instead, the processing of the upmix of the one transport channel is performed in the DFT domain of the DFT stereo decoder, which is described in clause 6.3.2.3.

The encoded audio scene comprises the transport channel signals (first set) and a first set of parameters. The transport channels are encoded using DFT stereo. The parameters consist of SPAR parameters for the low-frequency parameter bands and DirAC parameters, which describe a DoA and a diffuseness of the scene at the virtual listener position, for the high-frequency parameter bands.

This first set of parameters is converted to a second set of parameters. These parameters are employed for generating a representation of the scene by audio channels for reproduction at predefined spatial positions of a stereo loudspeaker of headphone setup. The conversion is described in clause 6.4.6.5.8.2

An output interface generates the processed audio scene using the second set of parameters and the first set of transport-channel signals by means of an upmix to stereo. This is described in clause 6.4.6.5.8.4. The complete processing diagram of the SBA-to-stereo decoding is shown in Figure 6.4-11 and Figure 6.4-12 for ACELP and TCX frames, respectively.

6.4.6.5.8.2 Parameter conversion

6.4.6.5.8.2.1 Overview

The parameters received and decoded from the bitstream comprise two different types: SPAR parameters (prediction and decorrelation) and DirAC parameters (DoA and diffuseness). The former are available for the 8 lower parameter bands, whereas the latter are available for each of the 4 higher parameter bands. A parameter conversion is performed in order to obtain a second set of parameters to describe the scene in the form of stereo audio channels. This conversion is achieved in the following steps.

6.4.6.5.8.2.2 Calculation of the model covariance

For the high-frequency parameter bands the inter-channel covariance matrix is estimated from the DirAC parameters according to clause 5.4.3.7.2.

6.4.6.5.8.2.3 Calculation of the SPAR parameters

From the model covariance for the high-frequency bands, the SPAR parameters comprising prediction and decorrelation coefficients are calculated according to clause 6.4.4.3.

6.4.6.5.8.2.4 Calculation of the SPAR upmix matrix

From the combined input parameters from the bitstream and clause 6.4.4.3, the upmix matrix is calculated according to clause 6.4.5.3. It is of the form

$$\boldsymbol{U} = \begin{pmatrix} P_w & 0 & 0 & 0 \\ P_x & C_x & 0 & 0 \\ P_y & 0 & C_y & 0 \\ P_z & 0 & 0 & C_z \end{pmatrix}, \tag{6.4-54}$$

where P_w is the inverse of the active W weighting, and the parameters P_x, P_y, P_z are the residual prediction gains to restore the first-order ambisonics components that are correlated with the W channel in the encoder input signal. The matrix **U** is computed for each parameter band, but the band index is omitted from the notation here.

Pre-Multiplication of the internal channel vector

$$\boldsymbol{x}_{\text{int}} = \begin{pmatrix} x_w \\ d_x \\ d_y \\ d_z \end{pmatrix}. \tag{6.4-55}$$

By the matrix $\boldsymbol{U}$ for each parameter band yields the first order ambisonics signal $\boldsymbol{s}_{umx,foa}^{dft}$. x_w is the audio signal from the one transport channel in the first set. In the full SPAR decoding path, $d_{x;y;z}$ are decorrelator signals. In the optimized stereo output mode, they are replaced by the allpass-filtered signals from the Enhanced stereo filling according to clause 6.3.2.3.6.4.

6.4.6.5.8.2.5 Adaptive smoothing

To the matrix $\boldsymbol{U}$ an adaptive smoothing is applied. Specifically, the smoothed upmix matrix is obtained recursively from an averaging filter according to

$$\overline{\boldsymbol{U}} = \beta\overline{\boldsymbol{U}} + (1-\beta)\boldsymbol{U}_{\text{delayed}} \tag{6.4-56}$$

in each DFT-stereo subframe (10 ms).

$\boldsymbol{U}_{\text{delayed}}$ is a delayed version of the matrix $\boldsymbol{U}$ for the current frame, i.e., values from a preceding time frame, which is obtained from a delay line in the decoder. The time resolution of the parameters encoded in this matrix is 1 frame (20 ms) in the 8 lower parameter bands and 1 subframe (5 ms) in the higher 4 parameter bands. Equation (6.4-56) is evaluated in each output DFT-stereo subframe (10 ms).

In this way, the parameters for the high parameter bands are combined, providing an effective time resolution of 20 ms for the low and 10 ms for the high frequency parameter bands. Hence, the smoothed second set of parameters has a time resolution higher than one IVAS input frame but lower than one IVAS subframe.

The length of the delay line in IVAS subframes (5 ms) is

$$N_{delay} = \frac{t_{delay}}{\frac{L_{\text{frame}}}{N_{\text{sf}}}} \tag{6.4-57}$$

with the delay of the audio transport channel in nanoseconds, t_{delay}, the length of a frame in nanoseconds, L_{frame}, and the number of subframes per frame in the IVAS framework, $N_{\text{sf}} = 4$.

The delay for the optimized stereo output mode is given by

$$t_{\text{delay}} = t_{delay}^{SPAR} - t_{delay}^{CLDFB}. \tag{6.4-58}$$

Hence, a reduction of the codec delay by the contribution of the CLDFB filterbank is achieved as compared to the full SBA decoding (t_{delay}^{SPAR}). The lower delay with the optimized stereo output is 32 ms.

The adaptivity of the smoothing enters via the smoothing factor (update rate) β, which is dynamically calculated for each DFT-stereo subframe based on the signal energy. It is given by

$$\beta = \min\left(1, \frac{E_{\text{long}}}{E_{\text{short}}}\right). \quad (6.4\text{-}59)$$

The long and short energies E_{long} and E_{short} are obtained by averaging the signal energy E in the W channel of the transport channel vector, $s_{dmx,w}^{dft}$, in the first set over a longer and a shorter time interval. The longer time interval is $N_{\text{long}} = 10$ subframes (100 ms) and the shorter is $N_{\text{short}} = 3$ subframes (30 ms). Equations (6.4-58) and (6.4-59) are evaluated separately for each parameter band group.

The factor β is then re-mapped to the range of [0,1] using the formula

$$\beta \rightarrow \max\left(1, \beta - \frac{N_{\text{short}}}{N_{\text{long}}}\right) * \left(\frac{N_{\text{long}}}{N_{\text{long}} - N_{\text{short}}}\right).$$

For less extreme cases, the smoothing is now reduced excessively, so the factor is compressed with a root function (compression function) towards value 1. As stability is particularly important, the in lowest bands. the 4th root is used in bands b=0 and b=l

$$\beta \rightarrow \sqrt[4]{\beta},$$

while all other bands are compressed by a square root

$$\beta \rightarrow \sqrt{\beta}.$$

This way the compression is stronger for the lower frequency band than for the higher frequency bands. Extreme cases remain close to 0 while a less rapid increase in energy does not decrease smoothing so strongly.

Finally, the maximum smoothing is set depending on the band (a factor of 1 would simply repeat the previous value):

$$\beta \rightarrow \min(\beta, \beta_{\max}).$$

The value of $\beta_{\max}$ for each band is listed in Table 6.4-5.

Table 6.4-5: Maximum smoothing factor per parameter band

band index	β_{max}
0	0.98
1	0.97
2	0.95
3-12	0.90

The adaptive smoothing according to Equations (6.4-56) and (6.4-59) describes a weighted combination of the parameters of different subframes, where the weighting factors depend on the signal energy (and amplitude) in the transport channel. The weighting factor of $\boldsymbol{U}_{\text{delayed}}$ (update rate) in Equation (6.4-56) is greater in a subframe with a higher energy in the transport signal compared to a subframe with a lower signal energy.

Therefore, the weighted combination has the property that a relatively strong smoothing is acquired for some portions of the audio scene, while a relatively weak smoothing is acquired for other portions of the scene. The first portion consists of the slowly varying time intervals of the signal, whereas the second consists of the more rapidly varying ones.

6.4.6.5.8.3 Derivation of the W and Y channel weights

From the smoothed matrix $\overline{\boldsymbol{U}}$ mixing weights can be obtained to describe the audio scene in the form of stereo output channels depending on the one transport channel x_w and the Enhanced-stereo-filling signal d . The left and right stereo channels x_L and x_R follow from the internal ambisonics channels x_w and x_y as

$$x_L = x_w + x_y, \quad (6.4\text{-}60)$$

$$x_R = x_w - x_y. \quad (6.4\text{-}61)$$

Evaluating the x- and y-rows of the matrix vector product $\boldsymbol{U}\,\boldsymbol{x}_{\mathrm{int}}$ and inserting them into (6.4-60) and (6.4-61), the stereo channel signals become

$$x_L = \bar{P}_x x_w + \bar{P}_y x_w + \bar{C}_y d, \quad (6.4\text{-}62)$$

$$x_R = \bar{P}_x x_w - \bar{P}_y x_w - \bar{C}_y d. \quad (6.4\text{-}63)$$

Hence, the converted parameters in the second set are the smoothed matrix elements $\bar{P}_x$, $\bar{P}_y$, and $\bar{C}_y$. They describe a representation of the encoded audio scene in terms of two stereo output channels using the audio channels x_w and d from the first set of transport channels.

The parameters $\bar{P}_x$, $\bar{P}_y$, and $\bar{C}_y$ refer to the input parameter bands given in Table 5.4-5. As the channel signals refer to the DFT bins (output parameter bands), the parameters are expanded onto to DFT bins such that they apply approximately to the same frequency ranges.

6.4.6.5.8.4 Stereo output processing

6.4.6.5.8.4.1 General

The actual processing of the stereo output signal depends on the bitrate and the number of transport channels in the first set (1 or 2).

The processing also depends on the type of core-coder that is activated in the SCE. In the case of 1 transport channel, depending on the signal characteristics of the channel, each frame can be coded with either ACELP or TCX. This is described in detail in clause 6.2.3.2 . The processing for ACELP and TCX frames is described in clause 6.4.6.5.8.4.2 and 6.4.6.5.8.4.3, respectively. In the case of 2 transport channels, only TCX coding is employed.

Generally, Equations (6.4-62) and (6.4-63) are evaluated in the DFT domain. The differences arise from the sources of the signals x_w and d and the core-coder processing.

6.4.6.5.8.4.2 ACELP frames

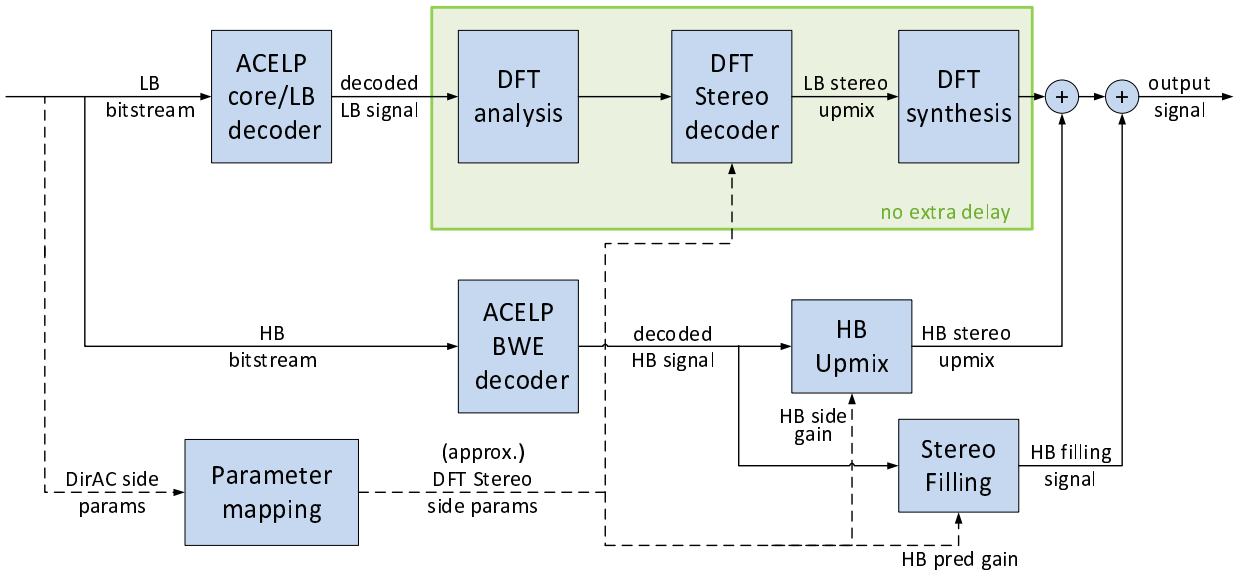


Figure 6.4-11: Block diagram of the SBA-to-stereo decoding path for ACELP frames

The decoding of the one transport channel consists of an ACELP core and a bandwidth extension (BWE) (see Figure 6.3-12). The generation of the BWE signal is performed in parallel to the DFT-Stereo processing for the raw representation (transport channel and artificial residual signal). Thus, the delays incurred by both algorithms do not accumulate, but only the given delay incurred by one algorithm will be the final delay.

The decoded transport signal from the ACELP core is fed into two branches: one for the processing of the low-frequency portion (upmix to the output signal) and one for the generation and upmix of the enhancement signals for the high frequency portion.

The low-frequency portion of the transport signal x_w is generated by the ACELP core-decoder. The low-frequency portion of d comes from the enhanced stereo filling with the low-frequency portion of x_w. These two signals are fed to the upmixer (output interface) in the DFT domain. After the upmixing, the output stereo signal is transformed back to the TD.

The high-frequency portion of x_w is generated in TD by the BWE. The high-frequency portion of d comes from the stereo filling in TD. These two high-frequency signals are upmixed in the time domain. The parameters $\bar{P}_x$, $\bar{P}_y$, and $\bar{C}_y$ for this broadband TD mixing obtained by averaging the values from the parameter bands 8,9, and 10 (see Table 5.4-5). Then the stereo decoding can be processed with a delay of 32 ms.

The low- and high-frequency (enhancement) portions of the output stereo signal are then combined in the TD.

6.4.6.5.8.4.3 TCX frames

The decoding simplifies for TCX frames. The processing diagram is shown in Figure 6.4-12 for the case of 1 transport channel. All frequencies are processed in the same way. The transport-channel signal x_w is decoded by the TCX core for all DFT bins up to configured audio bandwidth. The residual signal d is obtained from enhanced stereo filling for all DFT bins. Its bandwidth is limited to for perceptual reasons. For details see clause 6.3.2.3.

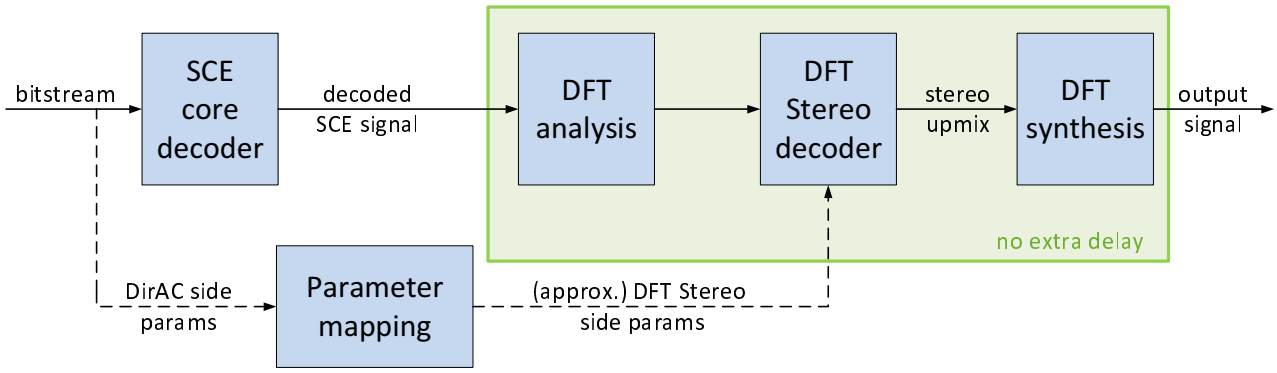


Figure 6.4-12: Block diagram of the SBA-to-stereo decoding path for TCX frames

In the case of 2 transport channels, these are decoded with a CPE. The second transport channel then contains the residual signal d, such that no enhanced stereo filling is required.

The stereo signal is generated by upmixing according to Equations (6.4-62) and (6.4-63) in each DFT bin of the spectral representation. It is then transformed back to the TD.

6.4.6.5.8.5 Mono output processing

6.4.6.5.8.5.1 Passive W

With passive W mixing, mono output becomes trivial. The W channel is output as the mono channel. It is already contained in the first set of transport channels.

6.4.6.5.8.5.2 Active W

To restore the W channel of the ambisonics input approximately, the band-weighting due to the active-W downmix must be reverted on the decoder side. For this purpose, the same processing as for stereo output is used. This is described in clause 6.4.6.5.8. However, only the W channel in Equations (6.4-62) and (6.4-63) is computed. Consequently, it is not necessary to run the stereo-filling processing.

6.4.6.6 CLDFB synthesis

The generation of the audio output channels in the SPAR and DirAC decoders in clauses 6.4.6.4 and 6.4.6.5 is generally performed in the CLDFB domain. This includes the rendering to binaural and loudspeaker output according to clauses 6.4.6.5.6 and 6.4.6.5.7, respectively. These output audio channels are then transformed back to the time domain with a CLDFB synthesis that is the inverse transform according to clause 6.2.5.2.

6.4.7 DTX operation

6.4.7.1 Overview

During the DTX operation, spatial comfort noise is generated during SID and NO_DATA frames. The process of spatial comfort noise generation includes:

- At the encoder
    - Computing noise estimates based on the primary downmix channel, where primary downmix channel is computed from Ambisonics input to SPAR and DirAC as described in clause 5.4.5.
    - Generating spectral shaping filter, in the core coding tool, based on the noise estimates. Coding the spectral shaping filter coefficients in SID frames and transmitting the coded bitstream to decoder.
- At the decoder
    - Generating a multi-channel comfort noise signal with a noise distribution such that all channels of the comfort noise signal start with a different seed value resulting in diffused multi-channel noise signal with uncorrelated channels.
    - Spectrally shaping the multi-channel comfort noise signal with the spectral shaping filter coefficients received in the SID frame bitstream.
    - Decoding the SBA metadata from the SID frame bitstream. The metadata contains prediction and decorrelation coefficients indicating the noise ambience of the Ambisonics input. The computation of prediction and decorrelation coefficients in SID frames is described in clause 5.4.9.
    - Spatially shaping the spectrally shaped multi-channel comfort noise signal with the prediction and decorrelation coefficients to generate the ambisonics comfort noise signal.
    - During the onset and offset of DTX, the Ambisonics comfort noise signal is crossfaded, as described in clause 6.4.6.4.6, and added to the Ambisonics audio signal that is transmitted from encoder to decoder during the adjacent non-DTX frames. In other DTX frames, the comfort noise signal is the Ambisonics output of decoder. The ambisonics output can then be rendered to the desired output format, e.g., loudspeakers, binaural etc.

6.4.7.2 DTX in DirAC

The decoding of the DirAC parameters in DTX frames is described in clause 6.2.4.5.

6.4.7.3 DTX in SPAR

The current frame is decoded as active or inactive frame based on active/inactive MD bit in SBA bitstream. This bit sets W_{vad} flag.

In active MD frames, SPAR PR and D coefficients are read from the bitstream as per the base2 and pair-wise coding described in clause 5.4.9.3.3. SPAR parameters in DirAC bands are generated as per the clause 6.4.4.3.

From the coefficients, P and C matrices are generated as described in clause 6.4.5.2. If the current frame is SID or NO_DATA frame, then further smoothing is applied to the P and C matrices. The P and C matrices that are generated with the SPAR MD received in the current SID frame are applied to the subsequent NO_DATA frames with a linear ramp function.

$$P_{curr} = P_{prevSID} + ramp(P_{currSID} - P_{prevSID})$$

where P_{curr} is the current SID or NO_DATA frame, $P_{currSID}$ is the P matrix as per latest SID frame, $P_{prevSID}$ is the P matrix as per the previous SID frame, $ramp = \frac{counter}{8}$, $counter$ is the number of frames since latest SID frame.

C matrix is ramped in a similar way

$$C_{curr} = C_{prevSID} + ramp(C_{currSID} - C_{prevSID})$$

The ramped P and C matrices are then converted to upmix matrix as per clause 6.4.5.3).

6.4.8 SBA PLC

6.4.8.1 PLC in DirAC

During packet loss, the DirAC rendering and upmixing is unchanged, except that the DirAC parameters cannot be read from the bitstream. Instead, previously well received DirAC parameters, i.e., DoAs and diffuseness, are used and hold for generating the output for the current lost frame. However, when the sound field is more diffuse, the directions are less meaningful and can be considered as the realization of a stochastic process. Dithering then helps make more natural and more pleasant, rendered sound field by injecting a random noise to the previous directions before using it for the lost frames. The inject noise and its variance σ_{azi} or σ_{ele} are function of the previous well-received diffuseness index, and the current direction is computed as:

$$\begin{cases} azi'[i] = azi[b] + rand\_tri() * \sigma_{\text{azi}}(\text{diff\_idx[b]}), \forall\, i \in I_b \\ ele'[i] = ele[b] + rand\_tri() * \sigma_{\text{ele}}(\text{diff\_idx[b]}), \forall\, i \in I_b \end{cases}$$

where b is the parameter band index, azi and ele are the previously well received and decoded direction angles, and $rand\_tri()$ is simulate random process following the triangle probability density using the following pseudocode:

```
rand_tri()
{
    rand_val = uniform_random();
    if( rand_val <= 0.0f )
    {
        return 0.5f * sqrt(rand_val + 1.0f) - 0.5f;
    }
    else
    {
        return 0.5f - 0.5f * sqrt(1.0f - rand_val);
    }
}
```

The variance of the injected noise is defined in tables given in Table 6.4-6:

Table 6.4-6: Variance of the dithering noise in function of the diffuseness index, for the azimuth and for the elevation

$\sigma_{\text{azi}}[]$	$\sigma_{\text{ele}}[]$
6.716062e-01, 1.011837e+00, 1.799065e+00, 2.824915e+00, 4.800879e+00, 9.206031e+00, 1.469832e+01, 2.566224e+01	6.716062e-01, 1.011804e+00, 1.796875e+00, 2.804382e+00, 4.623130e+00, 7.802667e+00, 1.045446e+01, 1.379538e+01

For HO-Dirac, no dithering is performed, and only the replacement of the current directions and diffuseness by the previously well-received directions and diffuseness is done.

6.4.8.2 PLC in SPAR

6.4.8.2.1 SPAR processing of erroneous frames

Packet loss concealment in SPAR is a second concealment step which is performed after the downmix channels have been provided by the core-decoder concealment. The method of creating replacements for lost SPAR parameters is applied to avoid a sudden change in spatial image during a sequence of one or more lost frames and to enable a quick recovery when receiving valid frames afterwards.

SPAR decoding of frames marked with the BFI flag as lost or erroneous comprises the following modifications.

When one or more lost frames are encountered by the decoder, which is indicated by the BFI flag, there will be no update of the SPAR parameters. Instead, the parameters of the last frame are kept and the SPAR upmix matrix remains unchanged.

The number of consecutive lost frames *num_lost_frames* is counted by increasing a corresponding counter if the BFI flag is raised. Otherwise, the counter is reset.

If the counter indicates 9 or more lost frames in a row, the following processing takes place:

- The spatial image will be faded over a period of 9 frames towards the predefined spatial configuration of a direction independent spatial image by muting all channels except the W channel. With

$$num\_fade\_frames = \min(num\_lost\_frames - 9, 0) \quad (6.4\text{-}64)$$

the channel dependent gain factor that is applied to the SPAR upmix matrix (see 0) is calculated according to

$$gain\_fade\_target(ch) = \begin{cases} 1, & \text{if } ch = 0 \\ 1 - min\left(\frac{num\_fade\_frames}{9}, 1\right), & \text{for all other channels} \end{cases} \quad (6.4\text{-}65)$$

- At the same time, a fade out of the decoded output is started. This is achieved by applying another gain $gain\_fade\_out$ to the SPAR upmix matrix. The ramp down extends over 33 frames and during that period the gain is decreased by 3dB per frame. The gain parameter is calculated as follows:

$$gain\_fade\_out = 10^{-\min(num\_fade\_frames, 33)\frac{3}{20}} \quad (6.4\text{-}66)$$

6.4.8.2.2 Recovery after one or more lost frames

As SPAR uses different time differential coding schemes for the SPAR parameters depending on the bitrate, slightly different SPAR parameter recovery strategies for the bands with directly coded SPAR parameters are defined after a sequence of one or more lost frames. The recovery logic as described in clauses 6.4.8.2.3, 6.4.8.2.4 and 6.4.8.2.5 is first applied to the invalid bands of decoded SPAR coefficients (prediction, cross-prediction and decorrelation coefficients). This is done so that SBA mono detection and SPAR to DirAC conversion happens on recovered frames. Then the same recovery logic as described in clauses 6.4.8.2.3, 6.4.8.2.4 and 6.4.8.2.5 is applied independently to the invalid bands of SPAR P and C matrices.

SPAR parameters that are derived from a DirAC to SPAR conversion are always marked as valid in good frames and no particular recovery method is needed for that case.

6.4.8.2.3 SPAR parameter recovery for 24.4 kbps and above

Given the time differential coding scheme for SPAR parameters which is defined in clause 5.4.3.7.14 for bitrates of 24.4 kbps and above, after the loss of one or more frames, the decoder will get at least ¼ valid parameters with the first good frame. If the last known valid parameter of a given band is more than 3 frames old, the time differentially coded parameters cannot be reconstructed and are consequently recovered by interpolation from adjacent bands with valid parameters. In case that for a specific band the last known value of the parameter is not older than three frames, this last good value will be kept.

6.4.8.2.4 SPAR parameter recovery for 13.2 kbps and 16.4 kbps

Low bitrates of 13.2 kbps and 16.4 kbps use a method of band interleaved 40 ms update rate transmission for efficient parameter coding as described in clause 5.4.3.7.15. Despite the total number of bands is 6 for these configurations, packet loss concealment operates on the normal SPAR banding of 12 bands. For this, the array indicating the bands with valid parameters in a frame is generated by converting the 6 bands representation to an upsampled version with 12 bands. This is, the 12-band parameters for band $2k - 1$ and band $2k$ are obtained from the respective 6-band parameters for band k, where $k = 1 \ldots 6$.

6.4.8.2.5 Interpolation of missing SPAR parameters from current existing SPAR parameters

To discriminate between good and not yet known parameters after bad frames, the decoder keeps track of the status with a 12-band variable of the valid bands. In addition, to identify bands that are still invalid, the number of frames since the last received set of valid parameters is counted.

Interpolation of missing SPAR parameters from current existing SPAR parameters then works as follows:

- Starting from zero, for all bands that contain invalid parameters with a band index below the first band containing valid parameters, the SPAR parameters of this first valid band are copied over to the invalid bands.

- Bands with invalid parameters that are located in between a lower band and a higher band with valid parameters are updated with a linear interpolation of the two valid band parameters. If id_0 is the band index of the lower band with valid parameters and id_1 corresponds to the index of the next higher band with valid parameters, the interpolation factor *w* for parameters of band *b* is:

- $$w = \frac{b - id_0}{id_1 - id_0} \quad (6.4\text{-}67)$$

  The missing SPAR parameter $p(b)$ is interpolated with the help of the valid parameters $p(id_0)$ and $p(id_1)$ by:

- $$p(b) = (1 - \mathrm{w})\, p(id_0) + w\, p(id_1) \quad (6.4\text{-}68)$$

- The parameters of all bands that contain invalid parameters with a band index higher than the last band containing valid parameters, are set to these valid parameters.

6.4.9 SBA bitrate switching

When the received bitstream signals a change in bitrate, the SBA decoder is reconfigured. This involves multiple steps.

First, the number of channels in the signal vectors $\boldsymbol{S}_{dmx}$, $\boldsymbol{S}_{umx}$, N_{dmx} and N_{spar_ana}, and the SBA analysis order are determined according to Table 5.4-1 and Table 5.4-2. Then the CLDFB memory is allocated and initialized accordingly.

If PCA is used with the current bitrate but not the previous, the corresponding structures are set up according to clause 6.4.6.3. If it is used in the previous but not the current bitrate, the corresponding buffers are freed.

If the number of transport channels changes to 1 with bitrate switching, then AGC is initialized otherwise AGC is deallocated if the number of transport channels changes from 1 to a value greater than 1.

If the DirAC metadata time resolution is 20ms or if the DirAC to SPAR conversion is disabled with the new IVAS bitrate, then SPAR parameter buffers and matrices are reallocated with 1 frame instead of 4 subframes to save memory.

The table row index for SPAR bitrate distribution table is computed as per the new IVAS bitrate and SBA analysis order. If N_{dmx} changes, the SPAR decoder is re-configured. In this step, the necessary number of TD decorrelators is set up, the SPAR MD decoder is reconfigured, the buffer for the upmix matrix is allocated with the correct dimensions, and the necessary number of JBM transport-channel buffers is allocated.

The renderer is selected, configured, and initialized according to the current bitrate.

The number DirAC parameters band and DoAs are re-computed and the buffers allocated accordingly. The flag for the higher-order mode is activated is the current bitrate is 384 kbps or 512 kbps. The sector decoding paths are configured according to clause 6.4.6.5.5.

The core-coder is re-configured to take into account the updated number of channels in the first set of transport channels, N_{dmx}, and the bitrate itself.

If the number of transport channels differs between the current and the previous bitrate, then HP20 filter buffers are allocated or deallocated to match the new number of transport channels. If the granularity or N_{dmx} changes, the JBM parameters are adjusted accordingly. The renderer buffers are flushed.

The JBM re-configuration upon a change of the IVAS bitrate is described in clause 6.2.7.5.

6.4.10 SBA output format conversion

The decoded audio scene can be rendered to different formats directly by the DirAC decoder. Hence a format conversion is not required. The output is generated using the methods according to the clauses listed in Table 6.4-7.

Table 6.4-7: Rendering method from SBA format to different output formats

Output format	clause
FOA	6.4.6.4
HOA (high -order mode)	6.4.6.5.3
HOA (low-order mode)	6.4.6.5.4
BINAURAL	6.4.6.5.6
BINAURAL_ROOM	6.4.6.5.6
Multi-channel	6.4.6.5.7
Stereo	6.4.6.5.8
Mono	6.4.6.5.8.5

6.4.11 SBA decoding with TSM

For mono and stereo output the all processing steps to produce the final output are done before the TSM in the first processing step. The transport channel buffer is a simple output buffer.

For all other output formats the following processing steps are done before the TSM:

- SPAR and DirAC parameter (meta data) and transport channel decoding
- Application of AGC/PCA on the transport channels
- In case of rendering to binaural with the parametric renderer or the parametric room renderer
    a. Calculation of the SPAR upmix matrix
    b. Application of the gain for binaural rendering on the transport channels

After the TSM and the transport channel buffer management according to clause 6.2.7.2, the local subframes are calculated according to 6.2.7.4.3.1, the meta data mapping $m_{ts,SPAR}$ for the SPAR upmix parameters is determined according to clause 6.2.7.4.2.1 Eqs. (6.2-91) and (6.2-93) with $L_{sf} = 1, m_{ts,SPAR} = m_{sf}$.

If the DirAC synthesis is done in the low-order operating mode (clause 6.4.6.5.4), the DirAC meta data mapping $m_{ts,DirAC}$ is determined according to clause 6.2.7.4.2.1 Eqs. (6.2-91) and (6.2-92) with $L_{sf} = 4, n_{offset}$ being the current read offset into the DirAC meta data buffer, and L_{md}=18, $m_{ts,DirAC} = m_{sf}$.

If the DirAC synthesis is done in the high-order operation mode or if the rendering to binaural is done with the parametric renderer or the parametric room renderer, the DirAC meta data mapping is first determined according to clause 6.2.7.4.2.1 Eqs. (6.2-91) and (6.2-93) with $L_{sf} = 4$, $m_{ts,DirAC} = m_{sf}$. The final DirAC meta data map $m_{sf,DirAC}$ is determined according to the following pseudo code:

$idx_{ts,abs} = 0$
for $idx_{sf} = 0, \dots, n_{SF} - 1$:
    $tmp = 0$
    For $idx_{ts} = 0, \dots, n_{slots,idx_{sf}} - 1$:
        $tmp = tmp + m_{ts,DirAC}(idx_{ts,abs})$
        $idx_{ts,abs} = idx_{ts,abs} + 1$
    $m_{sf,DirAC}(idx_{sf}) = \left\lfloor \frac{tmp}{n_{slots,idx_{sf}}} \right\rfloor$

The remaining processing steps are done with the adapted sub frames and using the meta data maps. The renderer granularity is always 1.25 ms.

6.5 Metadata-assisted spatial audio (MASA) decoding

6.5.1 Overview

The MASA format decoder decodes from the received IVAS encoded bitstream the MASA spatial metadata parameters and the associated transport audio signals. The MASA metadata decoding process is performed with the following general pattern.

First, MASA signalling parameters are decoded from the encoded bitstream. In practice, this is the inverse operation of what is described in clause 5.5.3.2. The determined signalling bits indicate the following states: number of transport channels in MASA format (1 or 2), number of concurrent directional parameters sets in the MASA spatial metadata (1 or 2), is the underlying format OMASA format, and the subframe structure of the metadata parameters ("1sf" or "4sf") as configuration parameter "joinedSubframes".

This is followed by the MASA codec configuration step (see clause 6.5.2) mirroring the encoder configuration step (see clause 5.5.3.2). This configuration setups the MASA decoder into correct state to be able to decode the encoded bitstream correctly based on the signalled information and the codec total bitrate.

If the codec configuration indicates low enough bit budget for the metadata coding, then low-bitrate signalling bit is also read as shown in clause 6.5.3.7.

Next, the MASA spatial metadata parameters (direction [as azimuth, elevation, and direction index], direct-to-total energy ratios associated with the directions, spread coherence, and surrounding coherence) are retrieved from the IVAS bitstream for each time-frequency tile of the configured coding time-frequency resolution (1 or 4 temporal subframes and 1-24 coding subbands) by the MASA metadata decoding steps in clauses 6.5.3 and 6.2.4.

The decoded metadata allows updating the state variable for indicating if there are significant coherence parameters present in the decoded spatial metadata or not. This information is used in rendering functions to optimize processing. Additionally, an encoding metric for the quality of encoded spatial metadata representation is formed (as detailed in clause 6.5.3.8).

Metadata decoding is followed by two possible replications steps. First, if the retrieved spatial metadata parameters were in the low-bitrate reduced time-frequency resolution with low-bitrate mode indication from signalling, then replication of spatial parameters is done from the low-bitrate reduced TF-resolution to 4 temporal subframes and 5 frequency bands TF-resolution. Second, if low-bitrate mode was not used but the indicated subframe structure is "1sf", then replication through time is done to obtain data for all 4 time subframes.

As the final part of the MASA metadata decoding, the coded TF-resolution metadata is replicated into full MASA format TF-resolution of 4 temporal subframes and 24 frequency bands and the non-transmitted spatial metadata parameters are reconstructed. This is detailed in clause 6.5.3.9.

The full format MASA metadata is then passed to the renderer in use (see clauses 6.5.7 and 7.2.2.3 for details) or to output for external processing (see clause 6.5.7.4).

After decoding the MASA metadata, similar to the encoder process in clause 5.5.3.1, the remaining bits are passed to the bit budget for CPE or SCE (whichever is in use) for decoding the associated transport audio channels. The transport audio channels are then decoded using the methods detailed in clauses 6.2.3.3 (for CPE, i.e., stereo-MASA) and 6.2.3.2 (for SCE, i.e., mono-MASA) and the decoded transport audio channels are also passed to the renderer in use (see clauses 6.5.7 and 7.2.2.3 for details) or to output for external processing (see clause 6.5.7.4).

6.5.2 Decoding MASA configuration

Decoding the MASA configuration mirrors the encoder configuration steps where applicable. The input received for this configuration process comprises of the IVAS format, codec total bitrate, and the signalled parameters presented in clause 6.5.1.

First, the decoder core coding elements are set similarly as in clause 5.5.3.2.2 followed by setting the common MASA metadata codec configuration (see clause 5.5.3.2.4) using the received configuration parameters. This sets the decoder into a same base state as the encoder was when encoding the bitstream. Similarly to the encoder, the number of coding bands B_{coding} and the number of bands with two directional parameter sets B_{2dir} are adjusted for transport audio signal bandwidth as presented in clause 5.5.3.2.11. The band mapping is adjusted as well. Further adjustments are performed in clause 6.5.3 when decoding the number of sub bands with no signal content when the information is available.

Next, a 16-bit spherical grid is created if codec total bitrate is 512 kbps.

This is followed by setting the configuration parameters in the metadata decoder structure. This includes metadata bit budget "max_metadata_bits", band mapping from coding bands to MASA frequency bands "band_mapping", number of transport channels "nchan_transport", number of bands with two directional parameters sets "numTwoDirBands", number of coding bands "nbands", number of time subframes "nblocks", and configuration parameter for selecting the method of compression used to compress the spatial metadata "mc_ls_setup". The parameter "nblocks" is set to 1 time subframe if "joinedSubframes" is true, otherwise it is in the default value of 4 time subframes. As detailed in clause

5.5.3.2.2, the configuration parameter “mc_ls_setup” provides for the encoder (clause 5.5.3.3) and decoder (clause 6.5.3) information for selecting the method based on the source format of the spatial audio content. In decoder, this indicative information of the method used to compress the spatial metadata associated with the spatial audio content is received by decoding from the bitstream (while receiving and decoding the spatial audio content and the associated spatial metadata) the IVAS format and possible MC-format channel layout and setting the value to “MC_LS_SETUP_INVALID” for MASA format or the channel layout for MC-format.

As part of the previous process, memory for the metadata decoder structure is also reserved based on the need.

6.5.3 MASA metadata decoding

6.5.3.1 Overview

The decoding flow for the MASA metadata is given by the following:

1. If the IVAS bitrate is larger or equal to 384 kbps the number of higher sub bands with no audio content is read with a Golomb Rice code of order 1
2. If coherence may be present (from the configurations) one bit is read to tell if it was all zero or not
3. If there are two directions and the bitrate is lower than 512 kbps
    a. Read which sub bands have the second direction by using a Golomb Rice decoder of order 0. It decodes the integer values representing differences minus one unit between two consecutive sub bands with two directions. The differences are read until the number of sub bands that should have the second direction for the given IVAS total bitrate is reached.
4. End If
5. Decode energy ratios of first direction
6. If there are 2 directions
    a. Decode the energy ratios of the second direction
7. End if
8. Calculate the total number of TF tiles for all directions
9. If coherence is non null and the remaining number of bits allocated for the metadata after decoding the energy ratios and the signalling so far – 4.3* number of TF tiles is larger than a threshold (12)
    a. Decode the surround coherence
10. Else
    a. Surround coherence is all zero
11. End if
12. Calculate remaining bits allocated for the reading the metadata
13. For each direction
    a. If coherence is present read bits for spread coherence
    b. Decode directions (elevation and azimuth)
    c. Decode the spread direction
    d. If first direction out of two directions update number of remaining bits for decoding
14. End for

6.5.3.2 Energy ratio decoding

6.5.3.2.1 Energy ratio decoding for bitrates up to 256 kbps

The energy ratio parameters for bitrates up to 256 kbps have been quantized with 3 bits. A single value per sub band, has been assigned at the encoder to all the sub frames of the same frame. The energy ratio values for the TF tiles of one direction or of the first direction are decoded according to the procedure described in clause 6.2.4.2 and one quantized value and one quantization index is obtained for each sub band. The decoding of the energy ratios for the second direction, when present in the encoded data in conformity with the common encoding-decoding configuration presented in table 5.5-3, is presented in clause 6.5.3.6.

6.5.3.2.2 Energy ratio decoding for 384 kbps and 512 kbps

For the higher bitrates, 384 kbps and 512 kbps the energy ratios have different values across the sub frames. They are decoded using the procedure from clause 6.2.4.2 for the 4-bit codebook case and decoding a separate value for each TF tile. The decoding of the energy ratios at 384 kbps for the second direction, when present, are decoded using the procedure described in clause 6.5.3.6. When the IVAS codec is operating at 512 kbps, the energy ratios of the second direction are independently decoded.

6.5.3.3 Direction decoding

6.5.3.3.1 Direction decoding for bitrates up to 256 kbps

The direction decoding consists of decoding the azimuth and elevation values for each TF tile, for each direction if data for more than 1 direction has been encoded. The decoded quantization indexes of the energy ratios are used to determine the initial bit allocation for the directional information of each TF tile according to table 5.2-32. The decoding flow is presented in the following:

1. Read 1 bit for 2D signalling for directional metadata
2. Read 1 bit, *ec_flag*, signalling entropy coding method (EC1 or other)
3. If EC1 (*ec_flag* = 0)
    a. Decode azimuth and elevation values using EC1 decoder from clause 6.2.4.3.2
4. Else
    a. If nblocks > 1
        i. Read EC2/EC3 (ec_flag2)
        ii. If EC2
            1. Decode azimuth and elevation values using EC2 decoder from clause 6.2.4.3.3
        iii. Else
            1. Decode azimuth and elevation values using EC3 decoder from clause 6.2.4.3.4
        iv. End if
    b. Else
        i. Decode azimuth and elevation values using EC3 decoder from clause 6.2.4.3.4
    c. End if
5. End if

6.5.3.3.2 Direction decoding for 384 kbps and 512 kbps

The decoding of the directions is using the raw decoding method presented in 6.2.4.1.3 for 11 bits and 16 bits at 384 kbps and 512 kbps respectively.

6.5.3.4 Spread coherence decoding

6.5.3.4.1 Spread coherence decoding for bitrates up to 256 kbps

There are *nblocks*×*nbands* spread coherence parameters for MASA format. The number of sub bands *nbands* and number of subframes *nblocks* are given in table 5.5-3. In MASA format the spread coherence is encoded for bitrates starting with 48 kbps. For the lower bitrates it is set to zero. When present, the spread coherence parameters are decoded according to the procedure presented in clause 6.2.4.4.1.1. If 2 directions are present the spread coherence parameters for the two directions are independently decoded.

6.5.3.4.2 Spread coherence decoding for 384 kbps and 512 kbps

The quantized spread coherence values were quantized using 3 and 4 bits at 384 kbps and 512 kbps respectively. Except for the number of bits, the decoding of the spread coherence parameters for these two bitrates follows the same flow and it is presented in clause 6.2.4.4.1.2.

6.5.3.5 Surround coherence decoding

6.5.3.5.1 Surround coherence decoding for bitrates up to 256 kbps

The surround coherence decoding for these bitrates is described in clause 6.2.4.4.2.1.

6.5.3.5.2 Surround coherence decoding for 384 kbps and 512 kbps

The surround coherence decoding for these bitrates is described in clause 6.2.4.4.2.2.

6.5.3.6 Decoding of the second direction parameters

6.5.3.6.1 Joint decoding of parameters associated with first and second sound source directions

The joint decoding of spatial parameters of a TF tile associated with a first and second sound source direction is performed for the azimuth direction and the direct-to-total energy ratio only. For both parameters, the decoding is performed when the second direction is signalled in the bitstream, and when the coding configurations have allowed the encoding of these parameters.

The azimuth direction parameters associated with the first direction are first decoded, followed by the decoding of one corresponding value for each TF tile where the second direction parameters were encoded. These second direction values are then subtracted from the first direction corresponding decoded azimuth values. The resulting differences are next transformed to the opposite direction by rotating them with 180 degrees and the transformed values are assigned as azimuth parameters of the second direction for the corresponding TF tile.

The direct-to-total energy ratio for each of the TF tile having second direction parameters are decoded according to the method presented in clause 6.2.4.2.2.

6.5.3.6.2 Independent decoding of parameters associated with first and second sound source directions

For the cases where the parameters associated with the first and second directions were independently considered for encoding, they are decoded following the one direction case decoding for the first direction, followed by the one direction decoding of the parameters corresponding to the second direction.

6.5.3.7 Reconstruction of MASA metadata from low-rate reduction

When the low-rate reduction of MASA metadata parameter values has been performed in the encoder (see clause 5.5.3.2.8), that is, the MASA metadata parameter values have been reduced from a time-frequency resolution of 5 frequency bands and 4 time subframes to either 1 frequency band and 4 time subframes or 5 frequency bands and 1 time subframe, then the original resolution is separated out from the metadata parameter values that have been decoded from the bitstream (with methods shown in clause 6.5.3 and 6.2.4).

First, the known configuration and read status bits are inspected to identify if a previous merging of MASA metadata parameter values has been done in the encoder to further reduce the number of MASA metadata parameter values in the encoded bitstream. This follows the practice in clause 5.5.3.2.8, that is, low-rate reductions are only done when the read subframe bit shows that the subframe mode is "4sf" and the configured metadata bit budget is less than a predefined value of MINIMUM_BIT_BUDGET_NORMAL_META bits. When this is true, an indicator bit is read from the encoded metadata bitstream. This indicator bit is associated with the previous merging and is based on the selected merging approach. This indicator bit value is stored into variable low_bitrate_mode.

With value of 1 in low_bitrate_mode, the indication is that merging has been done through time. In this case, the separation of parameter values is done by duplication of the single time subframe data for each metadata parameter value on each frequency band to all time subframes. This is done with equations:

$$\theta(b,m) = \theta(b,0)\ \forall m \in [0,3]$$

$$\phi(b,m) = \phi(b,0)\ \forall m \in [0,3]$$

$$r_{dir}(b,m) = r_{dir}(b,0)\ \forall m \in [0,3]$$

$$\zeta(b,m) = \zeta(b,0)\ \forall m \in [0,3]$$

$$\gamma(b,m) = \gamma(b,0)\ \forall m \in [0,3]$$

Where $\theta, \phi, r_{dir}, \zeta$, and γ are the MASA metadata parameter values azimuth, elevation, direct-to-total energy ratio, spread coherence, and surround coherence respectively.

With value of 0 in low_bitrate_mode, the indication is that merging has been done through frequency. In this case, the separation of parameter values is done by replication of the single frequency band data for each metadata parameter value on each time subframe to all frequency bands. This is done with equations:

$$\theta(b,m) = \theta(0,m)\ \forall b \in [0,4]$$

$$\phi(b,m) = \phi(0,m)\ \forall b \in [0,4]$$

$$r_{dir}(b,m) = r_{dir}(0,m)\ \forall b \in [0,4]$$

$$\zeta(b,m) = \zeta(0,m)\ \forall b \in [0,4]$$

$$\gamma(b,m) = \gamma(0,m)\ \forall b \in [0,4]$$

After this, the metadata parameter values are given for normal decoding and rendering process (see clauses 6.5.3 and 6.5.7) to produce output.

6.5.3.8 Determining encoding metric based on the quality of the spatial metadata representation in encoding

As part of the MASA spatial metadata decoding described in clauses 6.5.3 and 6.2.4, an encoding metric ν is determined. This encoding metric measures the quality of the entropy encoded spatial metadata when the EC3 direction encoding mode is in use (see clause 5.2.4.5.3) compared to the spatial metadata with initially allocated quantization accuracy. If EC1 or EC2 direction encoding mode is used, then $\nu = 1$. In case of EC3, the encoding metric is determined with the following procedure from the direction index spatial parameter decoding.

First, a parameter $bits_{alloc}$ is determined to indicate the number of bits allocated for the encoding of the direction index spatial parameter values for the frame. In practice, the bit allocation for each direction index spatial parameter value TF-tile is determined as detailed in clause 5.2.4.3.2.1 (and the corresponding decoding clause 6.2.4.1.2.1) and then all the bit allocation values for TF-tiles are summed together to determine the parameter $bits_{alloc}$. Note that this bit allocation value per TF-tile is also used to control the quantization of the full resolution direction parameter values into the direction indices by setting the quantization resolution of the spatial metadata.

Then, the direction index spatial parameter values are decoded from the bitstream according to the decoded signalling bits (as shown in clause 6.2.4.3). When the decoding is directed to using EC3, a parameter $bits_{used}$ is determined as the number of bits used for encoding all the direction index spatial parameters for the frame. This value is obtained as part of the decoding process of EC3. As EC3 reduces the quantization resolution of the spatial metadata as part of its coding process, this parameter value $bits_{used}$ also indicates the reduction of the quality of representation of the spatial metadata.

With the parameters $bits_{alloc}$ and $bits_{used}$ known, the encoding metric is determined as a ratio of the parameters, that is,

$$\nu = \frac{bits_{used}}{bits_{alloc}}$$

The encoding metric ν is then passed to the rendering tools (see clauses 6.5.7.2.8 and 7.2.2.3.3) to allow the control of spatial audio signal rendering process based on the encoding metric.

6.5.3.9 Reconstruction of full MASA format metadata

As part of the MASA format decoding (i.e., retrieving transport audio signals and associated metadata, see clause 6.5.1), a reconstruction of the full TF-resolution MASA spatial metadata is done. The input for this process is received from the MASA spatial metadata decoding (see clauses 6.5.3 and 6.2.4) as the decoded MASA spatial metadata parameters (directions, direct-to-total ratios associated with the directions, spread coherence, and surrounding coherence) for each time subframe and frequency band of the coded TF-resolution. As a preliminary step, the time resolution has been reconstructed into 4 time subframes regardless of the coded resolution (see clause 6.5.1). The number of frequency bands is 5-24.

This input metadata is reconstructed into a full format MASA spatial metadata with all spatial metadata parameters present on a full TF-resolution intended either for parametric rendering or output to external processing. When the target is parametric rendering, the TF resolution is 4 time subframes and 60, 40, or 20 frequency bins depending on the output sampling rate (48 kHz, 32 kHz, or 16 kHz respectively). When the target is output to external processing, then the MASA format native TF-resolution of 4 temporal subframes and 24 frequency bands is used. The targeted frequency resolution is reconstructed by replicating the coded frequency bands for each time subframe to full resolution frequency bands (or bins). For MASA spatial metadata parameter, this is done with the equation:

$$\rho'(k_b, m) = \rho(b, m), \qquad k_b \in [K_{map}(b), K_{map}(b+1) - 1]$$

where ρ is replaced with any MASA spatial metadata parameter, b is the coding band index, k_b is the frequency band (or bin) index for the full resolution metadata, $K_{map}(b)$ is the band mapping function for the start target band based on the codec band.

In addition, reconstruction of a full set of energy ratios is done to fulfill the MASA format specification. The full set of energy ratios contain 1 or 2 direct-to-total ratios (associated with the direction parameters) r_{dir1} and r_{dir2}, diffuse-to-total ratio r_{diff}, and remainder-to-total ratio r_{rem}. As described in clause 5.5.3.2.5, the energy ratios are associated with each other with the sum relation of

$$r_{dir1} + r_{dir2} + r_{diff} + r_{rem} = 1$$

To save bitrate, a smaller selection of energy ratios for each time subframe and frequency band is encoded into the metadata bitstream. As mentioned earlier this smaller selection received from the metadata decoder contains 1 or 2 direct-to-total ratios. Thus, the further energy ratios that were not transmitted (namely, diffuse-to-total energy ratio and remainder-to-total energy ratio) are obtained based on the received smaller selection of energy ratios for each time subframe and frequency band of the target TF-resolution. This is done with

$$r_{diff} = \begin{cases} 1 - r_{dir1}, & or \\ 1 - (r_{dir1} + r_{dir2}) \end{cases}$$
$$r_{rem} = 0$$

where the chosen equation for r_{diff} depends on the number of directional metadata parameter sets present in the encoded MASA metadata bitstream. In practice, r_{diff} is always the sum of received smaller selection of energy ratios (that is, 1-2 direct-to-total energy ratios) subtracted from one.

If the selected output is not for external processing, then all the above reconstructed spatial metadata parameters are provided to the rendering functions (see clauses 6.5.7 and 7.2.2.3 for details) for rendering the output audio signals using the reconstructed metadata (at the full TF-resolution and including the complete set of energy ratios) together with the associated transport audio signals. As part of this providing of the reconstructed spatial metadata to the selected renderer, the spatial metadata parameters are also delayed by two subframes (10 ms in time) to restore correct synchronization between the received and decoded transport audio signals and spatial metadata parameters.

If the selected output is for external processing, then the process in clause 6.5.8 is followed to create correct descriptive metadata and full MASA format output.

6.5.4 MASA DTX operation

The MASA DTX decoding operation is presented in clause 6.2.4.5.

6.5.5 MASA PLC

In case of a packet loss, the MASA metadata from the previous frame is used. The packet loss concealment operations for the transport audio signals are based on the methods for SCE and CPE for which the decoder operations are described in clauses 6.2.3.2 and 6.2.3.3, respectively.

6.5.6 MASA bitrate switching

The MASA operation under bitrate switching is inherently robust as described in clause 5.5.7.

6.5.7 MASA rendering

6.5.7.1 Binaural and stereo rendering

Binaural (with and without a room effect) and stereo rendering are performed using the parametric binauralizer and stereo renderer presented in clause 7.2.2.3.

6.5.7.2 Multi-channel loudspeaker and Ambisonics rendering

6.5.7.2.1 Overview

The multi-channel loudspeaker and Ambisonics (SBA) renderer can render multi-channel loudspeaker and Ambisonics output signals from decoded MASA transport audio signal(s) and spatial metadata.

The rendering is performed in subframes, where m denotes the subframe index. A subframe contains N_{slots} CLDFB slots (in non-JBM operation, $N_{slots} = 4$, in JBM operation $N_{slots} = 1 \ldots 7$). The data determined at previous calls (i.e., subframes m-1 and earlier) affects the rendering of the present subframe m due to temporal averaging and interpolation.

As an input, the renderer obtains (or receives) a spatial audio signal containing one or two transport audio signals and associated spatial metadata. The number of transport audio signals (i.e., one or two) is determined by decoding it from the received IVAS bitstream (see clause 6.1), and the renderer receives this information (i.e., the number of transport audio signals). The spatial metadata obtained (or received) by the renderer contains the following parameters associated with the audio signals: azimuth $\theta(k, m, i)$, elevation $\phi(k, m, i)$, direct-to-total energy ratio $r_{dir}(k, m, i)$, spread coherence $\zeta(k, m, i)$, and surround coherence $\gamma(k, m)$.

The fetching of the temporally correct spatial metadata parameter values for the current subframe m so that they are in sync with the audio signals is handled in clause 6.2.7. It operates differently for the JBM and non-JBM use. Fetching the correct spatial metadata values is not discussed in the following, it is assumed that it has already been correctly performed, as described in the aforementioned clause.

First, direct and diffuse power factors and surround coherence ratios are computed using the methods described in clause 6.5.7.2.2, based on the direct-to-total energy ratio $r_{dir}(k, m, i)$ and the surround coherence $\gamma(k, m)$.

Then, directional responses are computed using the methods described in clause 6.5.7.2.3 (clause 6.5.7.2.3.1 for Ambisonics output and clause 6.5.7.2.3.2 multi-channel loudspeaker output) based on the MASA spatial metadata.

Then, diffuse responses are computed using the methods described in clause 6.5.7.2.4.

Then, the transport audio signals are transformed to the time-frequency domain with a 60-bin (with the sampling rate of 48 kHz) complex low-delay filter-bank (CLDFB) (see clause 6.2.5 for details), resulting in $S(k, n, i)$, where k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index.

Then, prototype audio signals (direct and diffuse prototype audio signals) are determined based on the transport audio signals (and spatial metadata with Ambisonic output) using the methods presented in clause 6.5.7.2.5 (multi-channel loudspeaker output) or clause 6.5.7.2.6 (Ambisonics output).

Then, the determined diffuse prototype audio signals decorrelated using the methods presented in clause 6.5.7.2.7.

Then, the spatial audio output signals (multi-channel loudspeaker or Ambisonic signals) are synthesized using the methods presented in clause 6.5.7.2.8, yielding $S_{out}(k,n,j)$.

Finally, the time-frequency domain signals are converted to the time domain via the inverse CLDFB (see clause 6.2.5 for details), yielding $s_{out}(n,j)$, which are the synthesized time domain multi-channel loudspeaker or Ambisonic signals.

6.5.7.2.2 Determining direct and diffuse power factors and surround coherence ratio

The direct power factor is determined using the direct-to-total energy ratios $r_{dir}(k,m,i)$ by

$$f'_{dir}(k,m) = \sum_{i=0}^{D-1} r_{dir}(k,m,i), \quad if\ k < K_{maxdec}$$

$$f'_{dir}(k,m) = \left(\sum_{i=0}^{D-1} r_{dir}(k,m,i)\right)^2, \quad if\ k \geq K_{maxdec}$$

where k is the frequency bin index, m the subframe index, i the direction index, D the number of directions, and K_{maxdec} is the largest frequency bin where the decorrelation is applied.

The diffuse power factor is determined using the direct-to-total energy ratios $r_{dir}(k,m,i)$ by

$$f'_{diff}(k,m) = 1 - \sum_{i=0}^{D-1} r_{dir}(k,m,i), \quad if\ k < K_{maxdec}$$

$$f'_{diff}(k,m) = \left(1 - \sum_{i=0}^{D-1} r_{dir}(k,m,i)\right)^2, \quad if\ k \geq K_{maxdec}$$

Then, surround coherent energy factor is determined by

$$f_{surrcoh}(k,m) = f'_{diff}(k,m)\gamma(k,m)$$

The power ratios are adjusted using it by

$$f_{dir}(k,m) = f'_{dir}(k,m) + f_{surrcoh}(k,m)$$

$$f_{diff}(k,m) = f'_{diff}(k,m) - f_{surrcoh}(k,m)$$

Then, surround coherence ratio is determined by

$$r_{surrcoh}(k,m) = \frac{f_{surrcoh}(k,m)}{f_{surrcoh}(k,m) + f'_{dir}(k,m)}$$

6.5.7.2.3 Determining directional responses

6.5.7.2.3.1 Determining directional responses with HOA renderer

In this clause, the directional responses are determined for Ambisonics (SBA) output.

First, a directional response $g_{dir}(k,m,j)$ is determined for each spherical harmonic j using the method described in clause 6.5.7.2.3.3 based on the azimuth $\theta(k,m,0)$ and elevation $\phi(k,m,0)$ angles and the spread coherence $\zeta(k,m,0)$.

If there are two directions, a directional response $g_{dir,0}(k,m,j)$, $g_{dir,1}(k,m,j)$ is determined using the same method based on the azimuth $\theta(k,m,i)$ and elevation $\phi(k,m,i)$ angles and the spread coherence $\zeta(k,m,i)$ for the two directions i. Then, direct ratio parameters are determined based on direct-to-total energy ratios $r_{dir}(k,m,i)$ by

$$r_d(k,m,i) = \frac{r_{dir}(k,m,i)}{\sum_{i=0}^{1} r_{dir}(k,m,i)}$$

Using them, the combined directional response is determined by

$$g_{dir}(k,m,j) = r_d(k,m,0)g_{dir,0}(k,m,j) + r_d(k,m,1)g_{dir,1}(k,m,j)$$

Then, in case there are ISM directions (i.e., when operating in the OMASA mode), a directional response $g_{ISM,i}(k,m,j)$ is determined for each object i. This is performed by determining spherical harmonic gain vectors $\mathbf{g}_{SH,\theta,\varphi}(m,i)$, which correspond to the azimuth $\theta_{ISM}(m,i)$ and elevation $\phi_{ISM}(m,i)$ angles, using the ACN channel order (see clause 6.4.6.5.1) and the SN3D normalization, and setting the determined spherical harmonic gain vectors as the directional response ($g_{ISM,i}(k,m,j) = \mathbf{g}_{SH,\theta,\varphi}(m,i)$). The ISM directional responses are combined by

$$g_{ISM}(k,m,j) = \sum_{i=0}^{N_{obj}-1} g_{ISM,i}(k,m,j) r_{ISM}(k,m,i)$$

where $r_{ISM}(k,m,i)$ is the ISM energy ratio.

The directional responses from the MASA part and the ISM part are combined by determining direct ratio parameters

$$r_{d,MASA}(k,m) = \frac{\sum_{i=0}^{N_{dir}-1} r_{dir}(k,m,i)}{\sum_{i=0}^{N_{dir}-1} r_{dir}(k,m,i) + \sum_{i=0}^{N_{obj}-1} r_{ISM}(k,m,i)}$$

$$r_{d,ISM}(k,m) = \frac{\sum_{i=0}^{N_{obj}-1} r_{ISM}(k,m,i)}{\sum_{i=0}^{N_{dir}-1} r_{dir}(k,m,i) + \sum_{i=0}^{N_{obj}-1} r_{ISM}(k,m,i)}$$

and combining the responses by

$$g'_{dir}(k,m,j) = r_{d,MASA}(k,m)g_{dir}(k,m,j) + r_{d,ISM}(k,m)g_{ISM}(k,m,j)$$

If there are no ISM directions, $g'_{dir}(k,m,j) = g_{dir}(k,m,j)$.

Then, the surround coherence is rendered by

$$g''_{dir}(k,m,j) = g'_{dir}(k,m,j)\sqrt{1 - r_{surrcoh}(k,m)}, \quad if\ j \geq 1$$

where $r_{surrcoh}(k,m)$ is the surround coherence ratio determined in clause 6.5.7.2.2.

Then, squared direct responses are determined by

$$g_{dir,sq}(k,m,j) = \left(g''_{dir}(k,m,j)\right)^2$$

Then, modified direct responses are determined. For the first spherical harmonic, the modified direct response is set to 1

$$g'''_{dir}(k,m,0) = 1$$

If there is one transport audio signal, the remaining spherical harmonics are left unmodified

$$g'''_{dir}(k,m,j) = g''_{dir}(k,m,j), \quad if\ j \geq 1$$

If there are two transport audio signals, and the transport audio signal type is "$spaced$" (see clause 6.5.7.2.6.2 for the types), the modified direct response is set to 1 for the dipole frequency range for the second spherical harmonic

$$g'''_{dir}(k,m,1) = 1, \quad if\ 1 \leq k < 3$$

and left unmodified for the remaining frequency bins (i.e., $g'''_{dir}(k,m,1) = g''_{dir}(k,m,1)$).

If there are two transport audio signals and the transport audio signal type is not "$spaced$", the modified direct response is set to 1 for all frequency bins for the second spherical harmonic

$$g'''_{dir}(k,m,1) = 1$$

The remaining spherical harmonics are left unmodified

$$g'''_{dir}(k,m,j) = g''_{dir}(k,m,j), \quad if\ j \geq 2$$

6.5.7.2.3.2 Determining directional responses with VBAP renderer

In this clause, the directional responses are determined for multi-channel loudspeaker output. The directional responses are determined using VBAP. VBAP is initialized using the methods described clause 7.2.1.2.1. This can be done once during the initialization of the IVAS decoder.

A directional response $g_{dir}(k,m,j)$ is determined for each loudspeaker channel j using the method described in clause 6.5.7.2.3.4 based on the azimuth $\theta(k,m,0)$ and elevation $\phi(k,m,0)$ angles and the spread coherence $\zeta(k,m,0)$. The determined direct response gains $g_{dir}(k,m,j)$ are normalized using the methods presented in clause 6.5.7.2.3.5.

If there are two directions, a directional response $g_{dir,0}(k,m,j)$, $g_{dir,1}(k,m,j)$ is determined using the same method based on the azimuth $\theta(k,m,i)$ and elevation $\phi(k,m,i)$ angles and the spread coherence $\zeta(k,m,i)$ for the two directions i. The determined direct response gains $g_{dir,i}(k,m,j)$ are normalized using the methods presented in clause 6.5.7.2.3.5.

Then, direct ratio parameters are determined based on direct-to-total energy ratios $r_{dir}(k,m,i)$ by

$$r_d(k,m,i) = \frac{r_{dir}(k,m,i)}{\sum_{i=0}^{1} r_{dir}(k,m,i)}$$

Using them, the combined directional response is determined by

$$g_{dir}(k,m,j) = r_d(k,m,0)g_{dir,0}(k,m,j) + r_d(k,m,1)g_{dir,1}(k,m,j)$$

The determined direct response gains $g_{dir}(k,m,j)$ are normalized using the methods presented in clause 6.5.7.2.3.5.

Then, in case there are ISM directions (i.e., when operating in the OMASA mode), a directional response $g_{ISM,i}(k,m,j)$ is determined for each object using the method described in clause 7.2.1.2.2 based on the ISM directions (azimuth $\theta_{ISM}(m,i)$ and elevation $\phi_{ISM}(m,i)$). The ISM directional responses are combined by

$$g_{ISM}(k,m,j) = \sum_{i=0}^{N_{obj}-1} g_{ISM,i}(k,m,j) r_{ISM}(k,m,i)$$

where $r_{ISM}(k,m,i)$ is the ISM energy ratio. The determined direct response gains $g_{ISM}(k,m,j)$ are normalized using the methods presented in clause 6.5.7.2.3.5.

The directional responses from the MASA part and the ISM part are combined by determining direct ratio parameters

$$r_{d,MASA}(k,m) = \frac{\sum_{i=0}^{N_{dir}-1} r_{dir}(k,m,i)}{\sum_{i=0}^{N_{dir}-1} r_{dir}(k,m,i) + \sum_{i=0}^{N_{obj}-1} r_{ISM}(k,m,i)}$$

$$r_{d,ISM}(k,m) = \frac{\sum_{i=0}^{N_{obj}-1} r_{ISM}(k,m,i)}{\sum_{i=0}^{N_{dir}-1} r_{dir}(k,m,i) + \sum_{i=0}^{N_{obj}-1} r_{ISM}(k,m,i)}$$

and combining the responses by

$$g'_{dir}(k,m,j) = r_{d,MASA}(k,m)g_{dir}(k,m,j) + r_{d,ISM}(k,m)g_{ISM}(k,m,j)$$

The determined direct response gains $g'_{dir,i}(k,m,j)$ are normalized using the methods presented in clause 6.5.7.2.3.5. If there are no ISM directions, $g'_{dir}(k,m,j) = g_{dir}(k,m,j)$.

Then, the surround coherence is rendered by

$$g''_{dir}(k,m,j) = g'_{dir}(k,m,j)\sqrt{1 - r_{surrcoh}(k,m)} + \sqrt{\frac{r_{surrcoh}(k,m)}{N_{diff}}}$$

where N_{diff} is the number of diffuse rendering channels and $r_{surrcoh}(k,m)$ is the surround coherence ratio determined in clause 6.5.7.2.2. The determined direct response gains $g''_{dir}(k,m,j)$ are normalized using the methods presented in clause 6.5.7.2.3.5, yielding the output of the method, denoted as $g_{dir}(k,m,j)$ in the following.

In addition, squared direct responses are determined by

$$g_{dir,sq}(k,m,j) = (g_{dir}(k,m,j))^2$$

6.5.7.2.3.3 Spread coherence HOA panning

First, a spherical harmonic gain vector $\mathbf{g}_{SH,\theta,\varphi}$, which corresponds to the azimuth θ and elevation ϕ angles, is determined using the ACN channel order (see clause 6.4.6.5.1) and the SN3D normalization. This spherical harmonic gain vector is set as the directional response $g_{dir,centre}(j) = \mathbf{g}_{SH,\theta,\varphi}$, where j is the spherical harmonic index. Then, two additional direct responses $g_{dir,left}(j)$ and $g_{dir,right}(j)$ are determined using the same method based on the modified azimuth angles of $\theta + 30$ and $\theta - 30$ (elevation is kept as ϕ).

Then, centre and side gains are determined by

$$if\ \zeta < 0.5\ \begin{cases} g_{centre} = (1 - 2\,\zeta) + \dfrac{2\,\zeta}{3} \\ g_{side} = \dfrac{2\,\zeta}{3} \end{cases}$$

$$if\ \zeta \geq 0.5\ \begin{cases} g_{side} = \dfrac{1}{4 - 2\zeta} \\ g_{centre} = (2 - 2\,\zeta) g_{side} \end{cases}$$

Then, the direct response is determined by

$$g_{dir}(j) = g_{dir,centre}(j) g_{centre} + \left(g_{dir,left}(j) + g_{dir,right}(j)\right) g_{side}$$

6.5.7.2.3.4 Spread coherence VBAP panning

First, a directional response $g_{dir,centre}(j)$ is determined for each loudspeaker channel j using the method described in clause 7.2.1.2.2 based on the azimuth θ and elevation ϕ angles. Then, two additional direct responses $g_{dir,left}(j)$ and $g_{dir,right}(j)$ are determined using the same method based on the modified azimuth angles of $\theta + 30$ and $\theta - 30$ (elevation is kept as ϕ).

Then, centre and side gains are determined by

$$if\ \zeta < 0.5\ \begin{cases} g_{centre} = (1 - 2\,\zeta) + \dfrac{2\,\zeta}{\sqrt{3}} \\ g_{side} = \dfrac{2\,\zeta}{\sqrt{3}} \end{cases}$$

$$if\ \zeta \geq 0.5\ \begin{cases} g_{side} = 1 \\ g_{centre} = (2 - 2\,\zeta) \end{cases}$$

Then, the direct response is determined by

$$g_{dir}(j) = g_{dir,centre}(j) g_{centre} + \left(g_{dir,left}(j) + g_{dir,right}(j)\right) g_{side}$$

6.5.7.2.3.5 Panning gain normalization

First, the energy sum is computed of the direct response gains

$$E_{sum} = \sum_j g_{dir}(j)^2$$

Then, a normalization value is determined by

$$g_{norm} = \sqrt{\frac{1}{E_{sum}}}$$

Then, the panning gains are normalized by

$$g_{dir,norm}(j) = g_{norm} g_{dir}(j)$$

6.5.7.2.4 Determining diffuse responses

The diffuse response $g_{diff}(k, m, j)$ is determined based on the output format. In case of mono rendering (see clause 6.5.7.3), two values are set as:

$$g_{diff}(k, m, 0) = 1.0$$
$$g_{diff}(k, m, 1) = \frac{1}{\sqrt{3}}$$

In case of Ambisonics rendering, values are set up for each Ambisonics channel with following equations depending on the order of the Ambisonics:

$$g_{diff}(k, m, j) = \begin{cases} 1, when\ j = 0 \\ \frac{1}{\sqrt{3}}, when\ j \in [1,3] \\ \frac{1}{\sqrt{5}}, when\ j \in [4,8] \end{cases}$$

The channel index $j \in [0, J-1]$ where number of channels $J = (l+1)^2$ which depends on the order of the Ambisonics variable l.

In case of loudspeaker rendering, if the channel layout of the loudspeaker output format is 5.1, 5.1+2 or 5.1+4, then the diffuse response is obtained from table 6.5-1.

Table 6.5-1: Diffuse responses for channel layouts 5.1, 5.1+2, and 5.1+4

	j = 0	1	2	3	4	5	6	7	8
5.1	0.4219	0.4219	0.3704	0.5034	0.5034				
5.1+2	0.3817	0.3817	0.2819	0.5399	0.5399	0.1519	0.1519		
5.1+4	0.3456	0.3456	0.3035	0.4124	0.4124	0.2702	0.2702	0.3023	0.3023

For other channel layouts, the diffuse response equal for each channel and determined as

$$g_{diff}(k, m, j) = \frac{1}{\sqrt{N}}$$

where J is the number of channels in the loudspeaker setup.

6.5.7.2.5 Determining prototype audio signals for multi-channel loudspeaker output

In case of one transport audio signal, it is directly used as the prototype audio signal for both the direct and diffuse rendering

$$S_{proto,dir}(k, n, i) = S(k, n, i)$$

$$S_{proto,diff}(k, n, i) = S(k, n, i)$$

In case of two transport audio signals, three prototype audio signals are determined for the direct and the diffuse rendering

$$S_{proto,dir}(k, n, 0) = S(k, n, 0) + S(k, n, 1)$$

$$S_{proto,dir}(k, n, 1) = S(k, n, 0)$$

$$S_{proto,dir}(k,n,2) = S(k,n,1)$$

and they are the same for the direct and the diffuse rendering

$$S_{proto,diff}(k,n,i) = S_{proto,dir}(k,n,i)$$

6.5.7.2.6 Determining prototype audio signals for Ambisonics output

6.5.7.2.6.1 Overview

In case of one transport audio signal, it is directly used as the prototype audio signal for both the direct and diffuse rendering

$$S_{proto,dir}(k,n,i) = S(k,n,i)$$

$$S_{proto,diff}(k,n,i) = S(k,n,i)$$

In case of two transport audio signals, the stereo type of the MASA transport audio signals is determined using the method described in clause 6.5.7.2.6.2. Based on the determined transport audio signal type, the prototype audio signals are determined using the methods described in clause 6.5.7.2.6.3.

6.5.7.2.6.2 MASA transport signal stereo type detection

Two transport audio signals $S_i(k,n)$ (in the time-frequency domain, where k is the frequency bin and n is the slot index) are obtained as an input. Parameters associated with the transport audio signals are obtained by analysing the transport audio signals, and based on the obtained parameters, the type of the transport audio signals is determined. The transport audio signals are then processed based on the determined transport audio signal type, as detailed in clause 6.5.7.2.6.3, and the processed transport audio signals (i.e., the prototype audio signals) are rendered to spatial audio signals (Ambisonic audio signals), see clause 6.5.7.2.8. I.e., the transport audio signals are converted to Ambisonic audio signal representation based on the transport audio signal type.

First, the left and the right channel broadband energies are determined

$$E_{i,bb}(n) = \sum_{k=0}^{K-1} |S_i(k,n)|^2$$

where K is the number of frequency bins. And, the total broadband energy is determined

$$E_{total,bb}(n) = \sum_{i=0}^{1} E_{i,bb}(n)$$

These energies are smoothed over time by

$$E'_{i,bb}(n) = a_1 E_{i,bb}(n) + b_1 E'_{i,bb}(n-1)$$

$$E'_{total,bb}(n) = a_1 E_{total,bb}(n) + b_1 E'_{total,bb}(n-1)$$

$$a_1 = 0.01$$

$$b_1 = 1 - a_1$$

where $n-1$ refers to the last slot of the previous frame when $n = 0$.

Next, the smaller from the left and the right energies is selected and multiplied by 2

$$E'_{lr,bb}(n) = 2 \min_{\mathrm{i}} \left(E'_{i,bb}(n) \right)$$

The broadband LR-to-total ratio is computed using these energies as

$$\beta_{bb}(n) = 10 \log_{10} \frac{E'_{lr,bb}(n)}{E'_{total,bb}(n)}$$

Then, the left and the right channel high-frequency energies are determined

$$E_{i,hi}(n) = \sum_{k=15}^{K-1} |S_i(k,n)|^2$$

where K is the number of frequency bins. And, the total high-frequency energy is determined

$$E_{total,hi}(n) = \sum_{i=0}^{1} E_{i,hi}(n)$$

These energies are smoothed over time by

$$E'_{i,hi}(n) = a_2 E_{i,bb}(n) + b_2 E'_{i,hi}(n-1)$$

$$E'_{total,hi}(n) = a_2 E_{total,hi}(n) + b_2 E'_{total,hi}(n-1)$$

$$a_2 = 0.1$$

$$b_2 = 1 - a_2$$

Next, the smaller from the left and the right energies is selected and multiplied by 2

$$E'_{lr,hi}(n) = 2 \min_{\mathrm{i}} \left(E'_{i,hi}(n) \right)$$

The high-frequency LR-to-total ratio is computed using these energies as

$$\beta_{hi}(n) = 10 \log_{10} \frac{E'_{lr,hi}(n)}{E'_{total,hi}(n)}$$

Then, sum and total energies are determined

$$E_{sum}(k,n) = \left| \sum_{i=0}^{1} S_i(k,n) \right|^2$$

$$E_{total}(k,n) = \sum_{i=0}^{1} |S_i(k,n)|^2$$

These energies are smoothed over time by

$$E'_{sum}(k,n) = a_1 E_{sum}(k,n) + b_1 E'_{sum}(k,n-1)$$

$$E'_{total}(k,n) = a_1 E_{total}(k,n) + b_1 E'_{total}(k,n-1)$$

The minimum sum to total ratio is computed using them by

$$\chi(n) = 10 \log_{10} \min_{\mathrm{k}} \frac{E'_{sum}(k,n)}{E'_{total}(k,n)}, \quad 0 \le 0 \le 24$$

Then, subtract energy is computed by

$$E_{sub}(n) = |S_0(0,n) - S_1(0,n)|^2$$

The target Y energy $E_{target,y}(n)$ is determined in clause 6.5.7.2.8, and it is used in the following.

The energies are smoothed over time by

$$E'_{sub}(n) = a_3 E_{sub}(n) + b_3 E'_{sub}(n-1)$$

$$E'_{target}(n) = a_3 E_{target}(n) + b_3 E'_{target}(n-1)$$

$$a_3 = 0.0004$$

$$b_3 = 1 - a_3$$

The subtract to target ratio is computed using them by

$$\nu(n) = 10\log_{10}\frac{E'_{sub}(n)}{E'_{target}(n)}$$

Using the determined parameters, metrics are determined. First, a change to spaced metric is determined by

$$\Xi_s(n) = \frac{-\nu(n) - 3}{3} + \frac{\max(-\chi(n), 0)}{6} + \frac{\beta_{bb}(n)}{6}, \quad if\ \nu(n) < -3, else\ \Xi_s(n) = 0$$

Then, two change to downmix metrics are determined by

$$\Xi_{d1}(n) = \frac{\nu(n)}{3} + \frac{\chi(n) + 1}{6} + \frac{-\beta_{bb}(n)}{6}, \quad if\ \nu(n) > 0, else\ \Xi_{d1}(n) = 0$$

$$\Xi_{d2}(n) = \frac{\nu(n) + 4}{3} + \frac{\chi(n)}{6} + \frac{-\beta_{hi}(n) - 12}{3}, \quad if\ \beta_{hi}(n) < -12, else\ \Xi_{d2}(n) = 0$$

Using these metrics, the transport audio signal type is determined by

$$if\ \Xi_s(n) > 1, \qquad T(n) = "spaced"$$

$$else\ if\ \Xi_{d1}(n) > 1 \quad OR \quad \Xi_{d2}(n) > 1, \qquad T(n) = "downmix"$$

$$else, \qquad T(n) = T(n-1)$$

The transport audio signal type is changed only after the first 400 slots (i.e., 0.5 seconds). For the first 400 slots, the default value of “downmix” is used.

6.5.7.2.6.3 Prototype signal determination based on the detected stereo type

The prototype audio signals are determined based on the determined transport audio signal type (i.e., the stereo type) $T(n)$ (determined in clause 6.5.7.2.6.2).

If the transport audio signal type $T(n)$ is “$spaced$”, the prototype audio signals are determined as follows.

The first prototype audio signal for direct rendering is determined by (W prototype)

$$S_{proto,dir}(k, n, 0) = S(k, n, 0), \quad if\ k < 7$$

$$S_{proto,dir}(k, n, 0) = 0.5(S(k, n, 0) + S(k, n, 1)), \quad if\ k \geq 7$$

The second prototype audio signal for direct rendering is determined by (Y prototype)

$$S_{proto,dir}(k, n, 1) = S_{proto,dir}(k, n, 0), \quad if \quad k = 0$$

$$S_{proto,dir}(k, n, 1) = -\mathrm{i}\,(S(k, n, 0) - S(k, n, 1)), \quad if \quad 0 < k < 3$$

$$S_{proto,dir}(k, n, 1) = S_{proto,dir}(k, n, 0), \quad if \quad k \geq 3$$

The prototype audio signals for the diffuse rendering are determined by

$$S_{proto,diff}(k, n, 0) = S(k, n, 0) + S(k, n, 1)$$

$$S_{proto,diff}(k, n, 1) = S(k, n, 0)$$

$$S_{proto,diff}(k, n, 2) = S(k, n, 1)$$

If the transport audio signal type $T(n)$ is “$downmix$”, the prototype audio signals are determined as follows.

The first prototype audio signal for direct rendering is determined by (W prototype)

$$S_{proto,dir}(k, n, 0) = S(k, n, 0) + S(k, n, 1)$$

The second prototype audio signal for direct rendering is determined by (Y prototype)

$$S_{proto,dir}(k,n,1) = S_{proto,dir}(k,n,0) - S_{proto,dir}(k,n,1)$$

The prototype audio signals for the diffuse rendering are determined by

$$S_{proto,diff}(k,n,0) = S(k,n,0) + S(k,n,1)$$

$$S_{proto,diff}(k,n,1) = S(k,n,0)$$

$$S_{proto,diff}(k,n,2) = S(k,n,1)$$

6.5.7.2.7 Decorrelation

The decorrelated audio signals $S_{dec}(k,n,i)$ are determined by decorrelating the determined diffuse prototype audio signals $S_{proto,diff}(k,n,i)$.

Before the decorrelation, the determined prototype audio signals are mapped to the output channels, yielding $S_{proto,diff,mapped}(k,n,i)$. For the multi-channel loudspeaker output, in case of a single transport audio signal, the determined single diffuse prototype audio signal is set to all output channels.

In case of two transport audio signals, three prototype audio signals were determined. The first prototype audio signal $S_{proto,diff}(k,n,0)$ is used for output channels having the azimuth angle of 0. The second prototype audio signal $S_{proto,diff}(k,n,1)$ is used for output channels having the azimuth angle larger than 0. The third prototype audio signal $S_{proto,diff}(k,n,1)$ is used for output channels having the azimuth angle smaller than 0.

For the Ambisonics output, the diffuse rendering is performed via a virtual loudspeaker rendering. The virtual loudspeaker setup contains 8 loudspeakers at ±45 and ±135 degrees of azimuth with the elevations of 0 and 35 degrees. The mapped diffuse prototype audio signals $S_{proto,diff,mapped}(k,n,i)$ are determined using this virtual loudspeaker setup as for the multi-channel loudspeaker output.

Then, the mapped diffuse prototype audio signals $S_{proto,diff,mapped}(k,n,i)$ are decorrelated using the method described in clause 6.2.6, yielding $S_{dec}(k,n,i)$.

In case of multi-channel loudspeaker output, these are the output of the decorrelation.

In case of Ambisonic output, the decorrelated virtual-loudspeaker setup audio signals are converted to Ambisonic signals using a Ambisonic encoding matrix. The Ambisonic encoding matrix is determined by first determining a spherical harmonic gain vector $\mathbf{g}_{SH,\theta,\varphi}$ for each virtual loudspeaker direction, so that the spherical harmonic gain vector corresponds to the azimuth θ and elevation ϕ angles of the virtual loudspeaker, using the ACN channel order (see clause 6.4.6.5.1) and the SN3D normalization. The determined spherical harmonic gain vectors are combined to form the Ambisonic encoding matrix, which is then applied to the decorrelated virtual-loudspeaker setup audio signals to generate the decorrelated Ambisonic signals.

6.5.7.2.8 Spatial synthesis

The spatial audio signals, i.e., the multi-channel loudspeaker or Ambisonic signals, are determined (i.e., synthesized) in this clause.

First, the target direct energies are determined by

$$E_{target,dir}(k,m,j) = f_{dir}(k,m)\, g_{dir,sq}(k,m,j)\, E(k,m)$$

where $f_{dir}(k,m)$ is the direct power factor, $g_{dir,sq}(k,m,j)$ the squared direct responses, $E(k,m)$ the transport signal energy, k the frequency bin index, m the subframe index, and j the output channel index. The transport signal energy is determined by

$$E(k,m) = \sum_{i=0}^{N_{trans}-1} \left(\sum_{n=n_{first}(m)}^{n_{last}(m)} |\mathrm{S}(k,n,i)|^2 \right)$$

where $n_{first}(m)$ and $n_{last}(m)$ are the first and last temporal slots of the current subframe m, i the transport audio signal channel index, and the N_{trans} the number of transport audio signals.

The target direct amplitudes are determined by

$$A_{target,dir}(k,m,j) = f_{dir}(k,m)g_{dir}(k,m,j)E(k,m)$$

where $g_{dir}(k,m,j)$ is the direct responses.

The target diffuse energies are determined by

$$E_{target,diff}(k,m,j) = f_{diff}(k,m)g_{diff,sq}(k,m,j)E(k,m)$$

where $f_{diff}(k,m)$ is the diffuse power factor and $g_{diff,sq}(k,m,j)$ the squared diffuse responses.

If the output is Ambisonics, the target Y energy is determined to be used for the stereo type detection in clause 6.5.7.2.6.2 by

$$E_{target,y}(m) = E_{target,dir}(0,m,1) + E_{target,diff}(0,m,1)$$

The rendering gains (for example, amplitude panning gains) are smoothed over time to mitigate artefacts due to time-frequency processing. In the following, it is described how the temporal smoothing parameters are adaptively determined based on the spatial metadata parameters (the energy ratio parameter) and the quality of the encoding of the spatial metadata parameters.

First, an instantaneous direction smoothness parameter is determined

$$\xi(k,m) = \left(1 - r_{diff}(k,n)\right)^p$$

$$r_{diff} = 1 - \sum_{i=0}^{1} r_{dir}(k,n,i)$$

where $r_{dir}(k,n,i)$ is the direct-to-total energy ratio for the direction i and p is a tuning factor.

The direction smoothness parameter is averaged over time using energy weighting by

$$\xi_{sm}(k,m) = \frac{0.01E(k,m)\xi(k,m) + 0.99E_{sm}(k,m-1)\xi_{sm}(k,m-1)}{0.01E(k,m) + 0.99E_{sm}(k,m-1)}$$

where $E(k,m)$ is the energy for the subframe m, and $E_{sm}(k,m)$ is a temporally smoothed version of the energy

$$E_{sm}(k,m) = 0.01E(k,m) + 0.99E_{sm}(k,m-1)$$

Then, an adaptive smoothing parameter is determined using the averaged direction smoothness parameter

$$\alpha'(k,m) = \xi_{sm}(k,m)\alpha_{fast}(k) + (1 - \xi_{sm}(k,m)\alpha_{slow}(k)$$

where $\alpha_{slow}(k)$ and $\alpha_{fast}(k)$ are determined as follows. Denoting $t_\alpha = 20$ and $\alpha_{max} = 0.1$, $\alpha_{slow}(k)$ is determined using

$$\alpha_{slow}(k) = \min\left(\alpha_{max}, \frac{(\max(k,1) + 0.5)}{2t_\alpha}\right)$$

$\alpha_{fast}(k)$ is determined using the same equation as $\alpha_{slow}(k)$, but with $t_\alpha = 10$ and $\alpha_{max} = 0.12$.

Then, the encoding quality based smoothing factor (i.e., a smoothing control factor) is determined by

$$\mu = \nu^2$$

where ν is an encoding metric indicating the quality of encoding (determined in clause 6.5.3.8). Then, the adaptive smoothing parameter is modified using it by

$$\alpha(k,m) = \mu\alpha'(k,m) + (1 - \mu)\alpha_{qb}$$

where $\alpha_{qb} = 0.02$.

The rendering gains are then to be temporally smoothed using the adaptive smoothing parameter $\alpha(k,m)$, and the smoothed rendering gains to be applied to the audio signals to position the audio signals. Thus, the spatial audio signals are generated from the transport audio signals based on the spatial metadata, the encoding metric, and the direction smoothness parameter.

The prototype signal energies are determined by

$$E_{proto}(k,m,j) = \sum_{n=n_{first}(m)}^{n_{last}(m)} \left|S_{proto,dir}(k,n,j)\right|^2$$

The decorrelated signal energies are determined by

$$E_{dec}(k,m,j) = \sum_{n=n_{first}(m)}^{n_{last}(m)} |S_{dec}(k,n,j)|^2$$

The target direct energies, target direct amplitudes, target diffuse energies, prototype signal energies, and decorrelated signal energies are temporally smoothed using the adaptive smoothing parameter $\alpha(k,m)$

$$E_{target,dir,sm}(k,m,j) = \alpha(k,m)E_{target,dir}(k,m,j) + \big(1-\alpha(k,m)\big)E_{target,dir,sm}(k,m-1,j)$$

$$A_{target,dir,sm}(k,m,j) = \alpha(k,m)A_{target,dir}(k,m,j) + \big(1-\alpha(k,m)\big)A_{target,dir,sm}(k,m-1,j)$$

$$E_{target,diff,sm}(k,m,j) = \alpha(k,m)E_{target,diff}(k,m,j) + \big(1-\alpha(k,m)\big)E_{target,diff,sm}(k,m-1,j)$$

$$E_{proto,sm}(k,m,j) = \alpha(k,m)E_{proto}(k,m,j) + \big(1-\alpha(k,m)\big)E_{proto,sm}(k,m-1,j)$$

$$E_{dec,sm}(k,m,j) = \alpha(k,m)E_{dec}(k,m,j) + \big(1-\alpha(k,m)\big)E_{dec,sm}(k,m-1,j)$$

Processing gains are next determined using the temporally smoothed energies, i.e., these processing gains have been temporally smoothed using the determined adaptive smoothing parameter $\alpha(k,m)$. The direct and diffuse processing gains, h_{dir} and h_{diff} respectively, are determined by

$$h_{dir}(k,m,j) = \sqrt{\frac{E_{target,dir,sm}(k,m,j)}{E_{proto,sm}(k,m,j)}}$$

$$h_{diff}(k,m,j) = \sqrt{\frac{E_{target,diff,sm}(k,m,j)}{E_{dec,sm}(k,m,j)}}, \quad if\ k < K_{maxdec}$$

$$h_{diff}(k,m,j) = \sqrt{\frac{E_{target,diff,sm}(k,m,j)}{E_{proto,sm}(k,m,j)}}, \quad if\ k \geq K_{maxdec}$$

In case of Ambisonics output, the direct processing gains are adjusted by

$$h_{dir}(k,m,j) := -1h_{dir}(k,m,j), \ \ if\ A_{target,dir,sm}(k,m,j) < 0$$

The spatial audio output signals (multi-channel loudspeaker or Ambisonic signals) are synthesized by

$$S_{out}(k,n,j) = h_{dir}(k,m,j)S_{proto,dir}(k,n,j) + h_{diff}(k,m,j)S_{dec}(k,n,j), \ \ if\ k < K_{maxdec}$$

$$S_{out}(k,n,j) = h_{dir}(k,m,j)S_{proto,dir}(k,n,j) + h_{diff}(k,m,j)S_{proto,dir}(k,n,j), \ \ if\ k \geq K_{maxdec}$$

The processing gains h_{dir} and h_{diff} can be interpolated between the values computed for the previous subframe and the current subframe so that the new values are not taken into use instantly, but, instead, slot by slot via interpolation.

6.5.7.3 Mono rendering

The mono signal is rendered as the Ambisonics (SBA) rendering presented in clause 6.5.7.2, but only the omnidirectional component W of Ambisonics is rendered, which is the output mono signal.

6.5.7.4 Output for external processing

When the IVAS decoder (see clause 6.1) is configured for external processing (EXT) output, then the MASA decoder is configured to produce fully compatible MASA format frames (as specified in Annex A of [12]) directly from the audio and metadata decoder outputs. This output is intended for further processing (e.g., retransmission, mixing, or external rendering).

The output is provided with a similar structure interface (of type MASA_DECODER_EXT_OUT_META) as the input was obtained in clause 5.5.2.1. Spatial metadata is provided by the metadata reconstruction step in clause 6.5.3.9. The descriptive metadata is constructed from the received signalling information as follows:

- formatDescriptor: set to “IVASMASA”
- numberOfDirections: equal to received number of directions in metadata
- numberOfChannels: set to number of transport channels (nchan_transport) – 1
- sourceFormat: set to 0
- transportDefinition: set to 0
- channelAngle: set to 0
- channelDistance: set to 0
- channelLayout: set to 0

The combined full MASA format of transport audio channels, spatial metadata, and descriptive metadata is then provided for output. Due to the inherent differences in the delay properties of the audio codec and the metadata codec, the provided MASA format output is out-of-sync with transport audio channels being 12 ms delayed compared to the metadata frames. This delay should be compensated for further processing where synchronization is important (e.g., rendering, mixing, etc). A recommended option for synchronization is to delay the metadata by 2 time subframes (10 ms).

The EXT output of MASA has an algorithmic delay of 32 ms, and therefore mono/stereo output is available through EXT output by stripping off the metadata.

6.5.8 MASA decoding with TSM

MASA decoding with time scale modification follows procedures explained in clause 6.2.7 (and specifically in subclause 6.2.7.4.2.2 for the metadata mapping) in addition to procedures in clause 6.5.

6.6 Object-based audio (ISM) decoding

6.6.1 Discrete ISM decoding mode

Figure 6.6-1 is a schematic block diagram illustrating the ISM decoder in the DiscISM mode.

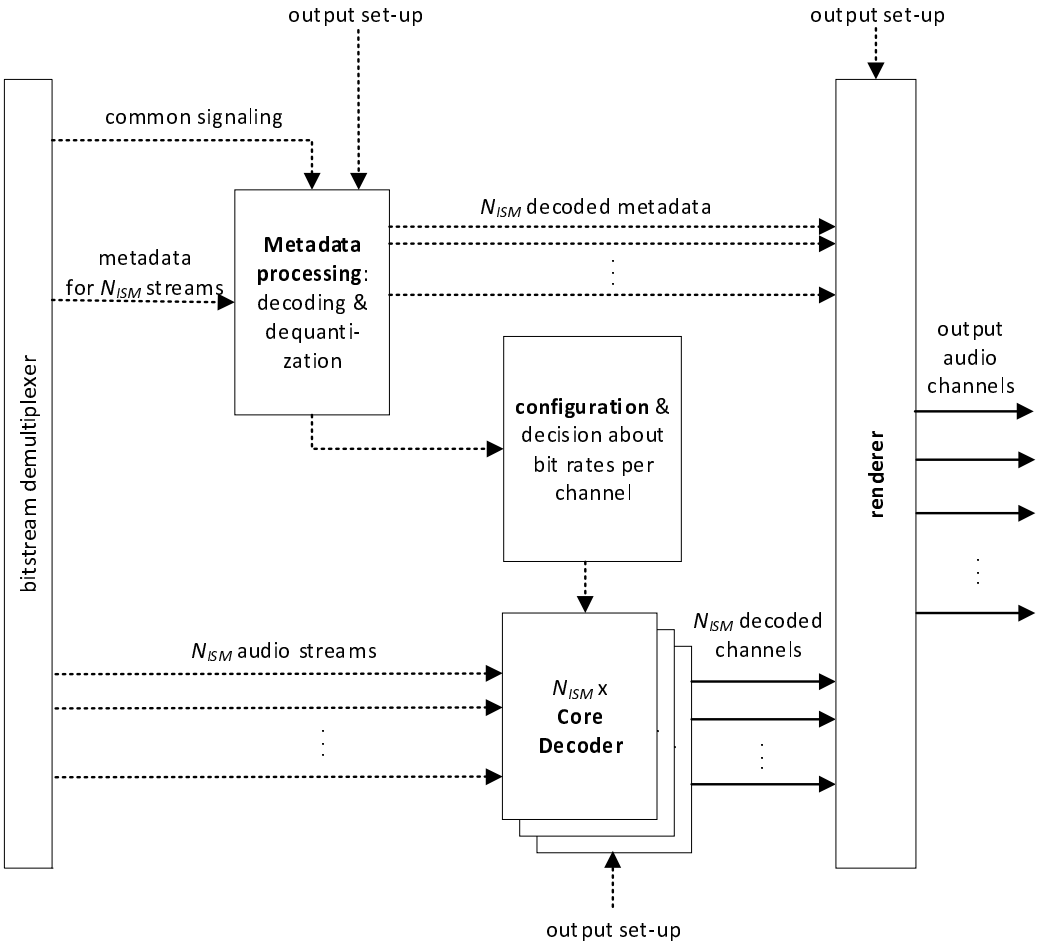


Figure 6.6-1: Block diagram of the DiscISM decoder

The bitstream demultiplexer receives a bitstream which is in the structure from Figure 5.6-4. When the IVAS format corresponds to the ISM format, the following is read from the bitstream in a sequential order: a) ISM common signaling incl. the number of audio streams, N_{ISM}, ISM importance classes, $class_{ISM}[n]$, and metadata presence flags, $flag_{meta}[n]$, $n = 0, \ldots, N_{ISM} - 1$, b) the coded metadata for N_{ISM} streams, c) core-coder payloads for N_{ISM} streams. It is noted that the ISM mode is not part of the bitstream in active frames but it is derived from the number of coded streams N_{ISM} and the *ism_total_brate* parameter.

One the metadata are decoded, the information about respective bit-budgets and ISM classes per stream are supplied from the metadata processing module to the configuration module which comprises the bit-budget allocator. The bit-budget allocator at the decoder uses the same procedure as in the bit-budget allocator at the encoder to determine the core-decoder bitrates (see clause 5.6.2.3).

Next, the N_{ISM} transport channels from the bitstream demultiplexer are sequentially decoded using N_{ISM} fluctuating bitrate SCE core-decoders (clause 6.2.3.2). These core-coder channels (corresponding to the transport channels) are finally supplied to the renderer.

It is noted that Figure 6.6-1 contains arrows indicating “output set-up” parameters. These parameters are e.g. output audio configuration, output sapling rate, etc. and they are used for simplifying some steps during the decoding process.

6.6.2 Parametric ISM decoding mode

6.6.2.1 General

In ISM format, if the bitrate was configured as 24.4 kbps or 32 kbps and there were 3 or 4 input ISMs, the input audio objects were encoded in Parametric ISM (ParamISM) mode. Per each time frame, the ParamISM decoder decodes an encoded audio signal comprising two transport channels and metadata, including direction information for three or four audio objects as well as parameter data for each audio object. The parameter data is used along with the transport

channels to render the original audio objects to a target output format. To that end, the two transport channels are transformed into a spectral representation and then rendered into a number of audio channels using the direction information (azimuth and elevation), contained in the transmitted metadata. The direction information is based on the two most relevant objects of each parameter band, where a parameter band comprises a number of time-frequency tiles. The parameter data includes object indices identifying the two most relevant objects per parameter band as well as a power ratio between the two relevant objects. Figure 6.6-2 shows an overview of the ParamISM decoder.

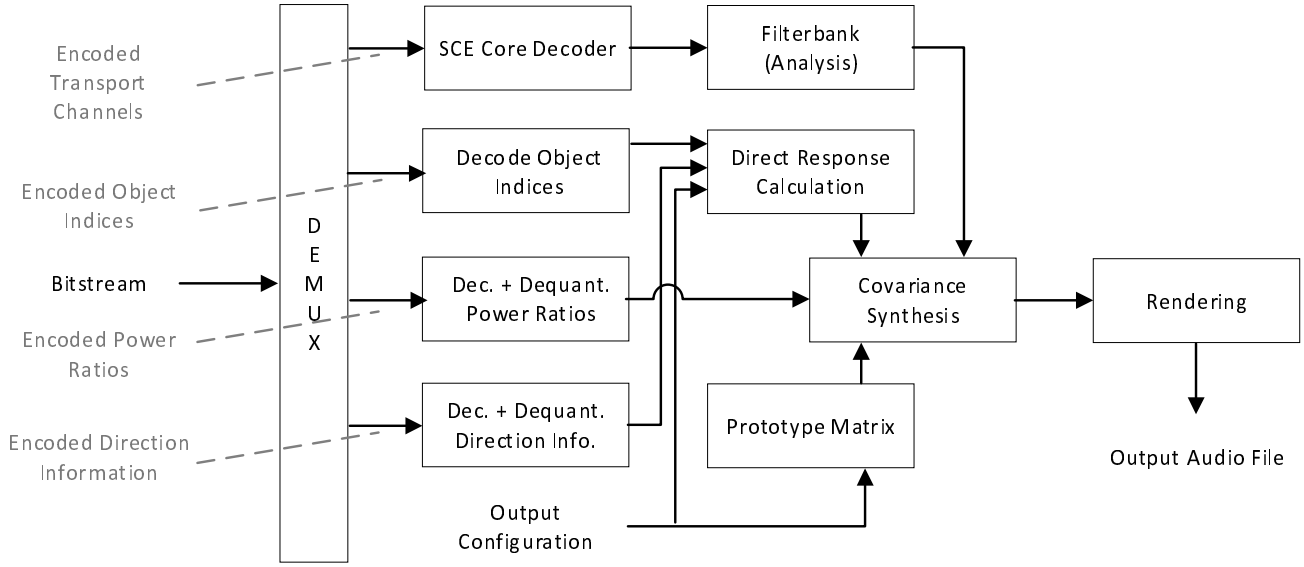


Figure 6.6-2: ParamISM decoder overview

6.6.2.2 Parameters

The parameter data is decoded from the bitstream for each frame. The decoded azimuth and elevation indices are mapped to their corresponding quantized azimuth and elevation values as described in clause 6.6.3.

The decoded power ratio index of each parameter band is mapped to its corresponding quantized power ratio value according to

$$\hat{r}_1(l) = r_{idx}(l)/(2(2^3 - 1)) + 0.5, \quad (6.6\text{-}1)$$

where l is the parameter band index. The second, not transmitted, power ratio belonging to the second most relevant object is obtained by:

$$\hat{r}_2(l) = 1 - \hat{r}_1(l) \quad (6.6\text{-}2)$$

Since $idx \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, the resulting values of $\hat{r}_1$ always range between 0.5 and 1 and the values of $\hat{r}_2$ always range between 0 and 0.5.

After also decoding the transmitted object indices, the obtained parametric data then comprises, per each time frame: azimuth and elevation information for each original input object, as well as, per parameter band, the object indices indicating the two most relevant objects and the power ratios indicating the contribution of each of the two relevant objects in terms of signal power. During rendering, the object indices and power ratios are expanded from the parameter band resolution (1 time slot by 9 or 11 frequency bands according to the input sampling rate) to the CLDFB filterbank resolution (see clause 6.2.5) by means of repetition. Furthermore, the noisy speech flag is decoded for each frame, indicating whether or not the encoded audio content was classified as noisy speech by the encoder.

6.6.2.3 Downmix

The two transport channels are decoded from the bitstream via the SCE decoder described in clause 6.2.3.2.

If the output format is set to loudspeaker, Ambisonics, or external (EXT), the following two processing steps – CLDFB analysis and energy compensation – are conducted in a frame-wise manner:

The decoded time-domain transport channels are transformed into a spectral representation by means of a CLDFB filterbank as defined in clause 6.2.5.1, resulting in a time-frequency resolution of, e.g., 16 time slots and 60 frequency bins per transport channel in the case of 48 kHz input. In the case of an input sampling rate of 32 or 16 kHz, the number of frequency bins reduces to 40 or 20, respectively.

The energy compensation conducted at the encoder (see clause 5.6.3.4) is reversed in the decoder:

$$DMX'_L = gain_{DMX}\, DMX_L \tag{6.6-3}$$

$$DMX'_R = gain_{DMX}\, DMX_R \tag{6.6-4}$$

$$gain_{DMX} = \sqrt{\frac{E_{DMX}}{E_{sum}+\epsilon}} \tag{6.6-5}$$

$$E_{sum} = \sum_j \left(DMX_L(j) + DMX_R(j)\right)^2 \tag{6.6-6}$$

$$E_{DMX} = \sum_j DMX_L{}^2(j) + DMX_R{}^2(j) \tag{6.6-7}$$

From the second frame onwards, a smoothing between frames is again employed, requiring $gain_{DMX}$ to be stored as $gain_{DMX}^{prev}$ for the next frame to be processed. The smoothed gain value is then given as:

$$gain_{DMX}^{smooth} = 0.75\, gain_{DMX} + 0.25\, gain_{DMX}^{prev} \tag{6.6-8}$$

The compensated transport channels are obtained by:

$$g_{nrg} = \frac{gain_{DMX}^{smooth} - gain_{DMX}^{prev}}{framesize/2} \tag{6.6-9}$$

$$DMX'_L(j) = \begin{cases} (gain_{DMX}^{prev} + j g_{nrg})\, DMX_L(j) & \forall\, j < framesize/2 \\ gain_{DMX}^{smooth}\, DMX_L(j) & \forall\, j \geq framesize/2 \end{cases} \tag{6.6-10}$$

$$DMX'_R(j) = \begin{cases} (gain_{DMX}^{prev} + j g_{nrg})\, DMX_R(j) & \forall\, j < framesize/2 \\ gain_{DMX}^{smooth}\, DMX_R(j) & \forall\, j \geq framesize/2 \end{cases} \tag{6.6-11}$$

6.6.2.4 ParamISM Decoding and Rendering

6.6.2.4.1 Output Formats

The decoded transport channels are rendered to the target output format by means of different rendering strategies that depend on the target output format. A general overview of the output formats and the respective rendering tools is found in clause 6.6.7 and Table 6.6-2.

6.6.2.4.2 Mono, stereo, and binaural output

For mono output, a mono downmix is generated as described in clause 6.6.7.2.

For stereo output, the decoded transport channels are output.

For the binaural output configurations, the parametric binaural renderer described in clause 7.2.2.3 is used. Prior to rendering, a loudness correction by a factor of 1.3657 is employed. The ParamISM parameters can be directly used as MASA parameters. Since ParamISM does not use any spread or surrounding coherence parameters, these are set to 0 for all frequency bins.

6.6.2.4.3 Loudspeaker and Ambisonics output

To render the decoded transport channels to a loudspeaker or Ambisonics configuration, EFAP described in clause 7.2.1.3 is used in a first step to calculate a direct response information for each object. Based on the transmitted and dequantized direction information (azimuth and elevation) of the original input objects as well as information on the target loudspeaker layout, the EFAP direct response values are determined in the form of direct response vectors $\boldsymbol{dr}_i$, with i the object index, resulting in one vector per object with each vector containing one value per each output channel (excluding any LFE loudspeaker channels). These values are specific to the target layout, i.e., the number and location

of the loudspeakers and serve as the panning gains. The direct response vectors are calculated once per frame and object.

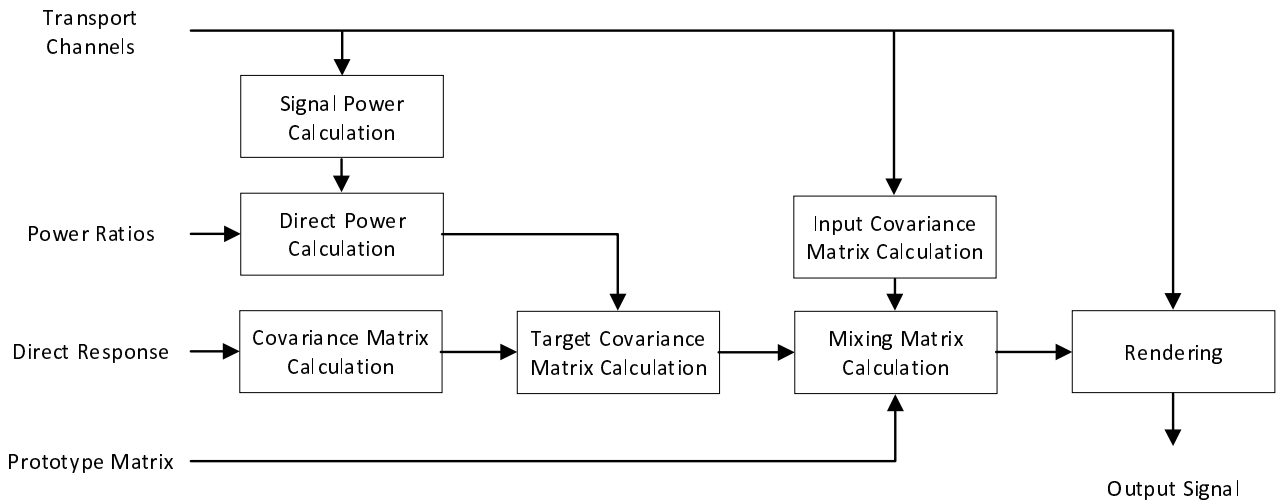


Figure 6.6-3: Overview of covariance synthesis

For rendering, covariance synthesis is employed using the direct response information as well as the information on the number of audio channels to acquire the output audio channels. Figure 6.6-3 shows an overview of the covariance synthesis. Specifically, a mixing matrix $\mathbf{M}_{sl}^{smooth}$ is calculated for each time slot and frequency bin of the current frame, and multiplied to the two decoded transport channels $\boldsymbol{x} = \boldsymbol{x}(k,n) = [\boldsymbol{X}_1, \boldsymbol{X}_2]^{\mathrm{T}} = [X_1(k,n), X_2(k,n)]^{\mathrm{T}}$, resulting in the output audio channels $\boldsymbol{y} = \boldsymbol{y}(k,n) = [Y_1(k,n), Y_2(k,n), Y_3(k,n), \dots]^{\mathrm{T}}$:

$$\boldsymbol{y} = \mathbf{M}_{sl}^{smooth}\boldsymbol{x} \tag{6.6-12}$$

The mixing matrix is calculated from the direct response values, the prototype matrix, the input covariance matrix $\mathbf{C}_x$ of the transport channels, the reference power of the transport channels, and the target covariance matrix $\mathbf{C}_y$ of the output channels.

The high pass filtered downmix signals s_{dmx}^{hp20} are transformed into the CLDFB domain according to clause 6.2.5.1, resulting in the CLDFB domain downmix signals $\boldsymbol{X}$.

The input covariance matrix for one CDLFB slot and one CLDFB band is given by the diagonal matrix:

$$\mathbf{C}_x(b,n) = \begin{bmatrix} X_1(b,n)X_1(b,n)^* & 0 \\ 0 & X_2(b,n)X_2(b,n)^* \end{bmatrix} \tag{6.6-13}$$

As only the main diagonal elements are used, all other matrix elements are explicitly set to zero.

The covariance matrix estimates are accumulated per CLDFB band for all CLDFB timeslots to create the sum of the covariances

$$\mathbf{C}_x(b) = \sum_{n=0}^{ts_{end}} \mathbf{C}_x(b,n) \tag{6.6-14}$$

All following steps are carried out for each band b.

The prototype matrix $\mathbf{Q}$ is defined as:

$$Q(j,i) = \begin{cases} 1 & \text{if } \left(i = 0 \wedge \theta_j > 0\right) \vee \left(i = 1 \wedge \theta_j < 0\right) \\ 0 & \text{if } \left(i = 0 \wedge \theta_j < 0\right) \vee \left(i = 1 \wedge \theta_j > 0\right) \\ 0.5 & \text{if } \theta_j = 0 \end{cases} \tag{6.6-15}$$

where $i \in \{0, 1\}$ denotes the transport channel index, $j \in \{0, 1, \dots, N\}$ denotes the output channel index, with N the number of output channels, and θ_j denotes the azimuth value of loudspeaker j.

The reference power per transport channel is obtained by:

$$P_i\ (k,n) = |X_i(k,n)|^2, \tag{6.6-16}$$

where X_i denotes the i-th transport channel in the spectral domain, k denotes the frequency bin index, and n denotes the time slot index. Since the rendering is done using the entire frame (as opposed to subframes), the signal powers of all time slots are accumulated. Furthermore, both transport channels are added to obtain the total signal power of the downmix:

$$P_{\mathrm{DMX}}(k) = \sum_{i=0}^{1}\sum_{n=0}^{15} P_i(k,n) \tag{6.6-17}$$

To calculate the target covariance matrix $\mathbf{C}_y$ for one frequency bin, the transmitted object indices and power ratios are required. For each of the 11 parameter bands, the two corresponding object indices, indicating the two relevant objects of each parameter band, are used to determine which direct response vectors to include in the direct response matrix.

The target covariance matrix for the frequency bin k is obtained as:

$$\mathbf{C}_y = \mathbf{R}\mathbf{E}\mathbf{R}^H, \tag{6.6-18}$$

with the direct response matrix $\mathbf{R} = [\boldsymbol{dr}_1\ \boldsymbol{dr}_2]$ that contains the previously obtained direct response vectors of the two relevant objects (as indicated by the transmitted object indices) and a diagonal matrix $\mathbf{E}$, with $e_{i,i} = E_i$ and

$$E_i = DP_i(k) \tag{6.6-19}$$

that contains the direct powers

$$DP_i(k) = \hat{r}_i * P_{\mathrm{DMX}}(k),\ \ i \in \{0,1\} \tag{6.6-20}$$

Here, i denotes the relevant object index. $\hat{r}_i$ is the quantized power ratio value of the i-th relevant object and is valid for all frequency bins that are contained in the parameter band that $\hat{r}_i$ was calculated for.

Up to this point, the noisy speech flag determined in the encoder was of no relevance. When calculating the direct power values, however, this flag is evaluated. If it is 0, the direct power values are obtained according to Eq. (6.6-20); if the flag is 1, instead of calculating two direct power values for two relevant object, the number of original input objects N_{ISM} is used so that three or four direct power values are calculated:

$$DP_i(k) = \frac{1}{N_{ISM}} * P_{\mathrm{DMX}}(k),\ \ i \in \{0,1,\dots,N_{ISM}-1\} \tag{6.6-21}$$

Consequently, the matrix $\mathbf{R}$ contains all N_{ISM} direct responses instead of only those of the two relevant objects and $\mathbf{E}$ is expanded from a 2×2 matrix to a $N_{ISM} \times N_{ISM}$ matrix. The dimensions of $\mathbf{C}_y$ remain unaffected. As the noisy speech flag is determined on a frame level, the decision how to calculate the direct power values and how to set up the direct response matrix is accordingly made on a per-frame basis.

Finally, the mixing matrix is calculated from the covariance matrices and the prototype matrix. $\mathbf{C}_y$ is decomposed according to

$$\mathbf{K}_y = \mathbf{R}\mathbf{E}^{\circ 1/2} \tag{6.6-22}$$

and $\mathbf{C}_x$ is decomposed according to

$$\mathbf{K}_x = \mathbf{C}_x{}^{\circ 1/2} \tag{6.6-23}$$

Here, the $^\circ$ operator denotes an element-wise operation. The matrix $\mathbf{K'}_x^{-1}$ is formulated as a regularized $\mathbf{K}_x$:

$$\mathbf{S}_x = \mathbf{K}_x \tag{6.6-24}$$

$$s_{x_{max}} = \max(\mathrm{diag}(\mathbf{S}_x)) \tag{6.6-25}$$

$$s'_{x_{i,i}} = \max\left(s_{x_{i,i'}}, \alpha s_{x_{max}}\right),\ \ i = 1,\dots,m \tag{6.6-26}$$

with a regularization factor $\alpha = 0.2$, and finally

$$\mathbf{K'}_x^{-1} = \mathbf{S'}_x^{\circ -1} \tag{6.6-27}$$

A normalization matrix $\mathbf{G}_{\hat{y}}$ is formulated according to

$$\mathbf{C}_{\hat{y}} = \mathbf{Q}\mathbf{C}_x\mathbf{Q}^H \tag{6.6-28}$$

$$c_{\hat{y}_{max}} = \max(\mathrm{diag}(\mathbf{C}_{\hat{y}})) \tag{6.6-29}$$

$$c'_{\hat{y}_{i,i}} = \max\left(c_{\hat{y}_{i,i}}, \beta c_{\hat{y}_{max}}\right),\ i = 1, \ldots, n \tag{6.6-30}$$

with a regularizaton factor $\beta = 0.001$. Given $\mathbf{C}_y = \mathbf{R}\mathbf{E}\mathbf{R}^H$, the elements of the normalization matrix are then defined as

$$g_{\hat{y}_{i,i}} = \sqrt{\frac{c_{y_{i,i}}}{c'_{\hat{y}_{i,i}}}},\ i = 1, \ldots, n \tag{6.6-31}$$

An optimal mixing matrix $\mathbf{M}_{opt}$ is then formulated with

$$\mathbf{U}, \mathbf{S}, \mathbf{V} = \mathrm{SVD}(\mathbf{K}_x^H\mathbf{Q}^H\mathbf{G}_{\hat{y}}^H\mathbf{K}_y) \tag{6.6-32}$$

$$\mathbf{P} = \mathbf{V}\mathbf{U}^H \tag{6.6-33}$$

$$\mathbf{M}_{opt} = \mathbf{K}_y\mathbf{P}\mathbf{K}'^{-1}_x \tag{6.6-34}$$

where $\mathrm{SVD}(.)$ denotes a singular value decomposition. As a last step, an energy compensation using a diagonal compensation matrix $\mathbf{G}$ is applied to finally yield the mixing matrix $\mathbf{M}$:

$$\mathbf{C}_{\hat{y}} = \mathbf{M}_{opt}\mathbf{C}_x{\mathbf{M}_{opt}}^H \tag{6.6-35}$$

$$g_{i,i} = \sqrt{\frac{c_{y_{i,i}}}{c_{\hat{y}_{i,i}}}},\ i = 1, \ldots, n \tag{6.6-36}$$

$$\mathbf{M} = \mathbf{G}\mathbf{M}_{opt} \tag{6.6-37}$$

For each time slot sl in the frame a smoothed mixing matrix

$$\mathbf{M}_{sl}^{smooth} = g(sl)\,\mathbf{M} + (1 - g(sl))\,\mathbf{M}^{prev} \tag{6.6-38}$$

is obtained from the current and previous mixing matrices and applied on each CLDFB band according to Eq. (6.6-12). The employed interpolator g_{sl} is a frame interpolator according to clause 6.2.7.4.3.2.

After applying the mixing matrix to all slots, the resulting output channels are converted into the time domain by means of a CLDFB synthesis filterbank as defined in clause 6.2.5.2 to acquire the output audio channels in the time domain. The mixing matrix $\mathbf{M}$ (not the smoothed mixing matrix!) is stored as $\mathbf{M}^{prev}$ for use in the next frame. Furthermore, the azimuth and elevation values transmitted as parametric data for each input object are stored as output ISM metadata.

The steps described so far apply to both loudspeaker and Ambisonics output configurations. For the latter, an additional conversion step is required to convert the multichannel format to the desired Ambisonics (FOA, HOA2 or HOA3) format. For all Ambisonics output configurations, the 7.1+4 multi-channel layout is used as an intermediate format. The conversion is then done according to clause 7.5.2.2.

6.6.2.4.4 EXT output

For EXT output, no rendering to a specific output format is conducted. Instead, the encoded objects are output again as ISMs so that the number of input channels matches the number of output channels. Furthermore, the decoded original associated metadata (if present) is output in quantized form according to clause 6.6.3.

The direct response values in this case are not obtained via EFAP, but explicitly set to:

$$R(j,i) = \begin{cases} 1 & if\ i = j \\ 0 & if\ i \neq j \end{cases} \tag{6.6-39}$$

where i denotes the object index and j denotes the output channel index.

The prototype matrix is defined as:

$$Q(j,i) = \begin{cases} 1 & \text{if } \left(i = 0 \ \wedge \ \theta_j > 0\right) \vee \left(i = 1 \wedge \theta_j < 0\right) \\ 0 & \text{if } \left(i = 0 \wedge \theta_j < 0\right) \vee \left(i = 1 \wedge \theta_j > 0\right) \\ 0.5 & \text{if } \theta_j = 0 \end{cases} \quad (6.6\text{-}40)$$

where $i \in \{0, 1\}$ denotes the transport channel index, $j \in \{0, 1, \dots, N\}$ denotes the output channel index, with N the number of input objects, and θ_j denotes the azimuth value of object j. In the case of the external output configuration, the number of input objects/channels is equal to the number of output objects/channels.

The remaining rendering steps correspond to those of loudspeaker rendering described in clause 6.6.2.4.3.

6.6.3 ISM metadata decoding

The metadata decoding and dequantization process is the same regardless of the ISM mode being DiscISM or ParamISM and starts with decoding the ISM common signalling including the number of audio streams, N_{ISM}, and metadata presence flags $flag_{meta}[n], n = 0, \dots, N_{ISM} - 1$.

The metadata are then decoded and processed in the metadata processing module (see Figure 6.6-1) where the metadata are decoded and dequantized for audio streams with active content. The decoding and dequantization performed by the metadata processing module at the decoder are the inverse of the quantization and coding performed by the metadata processing block at the encoder from Figure 5.6-1 and clause 5.6.4. The produced decoded and dequantized metadata for the N_{ISM} audio streams are then supplied to the renderer or a metadata writer in case of an external rendering.

6.6.4 DTX operation decoding

6.6.4.1 Overview

This clause describes a method for decoding audio objects (ISMs) during DTX operation where each audio object includes an audio stream and metadata (MD) parameters. It consists of decoding the metadata comprising adjusting their values to lower differences in the MD parameters between frames (see details in clause 6.6.4.3) and decoding the audio streams (see details in clause 6.6.4.4).

In DTX operation mode, the object or transport channel signals are only decoded directly from the bitstream if the signal exhibits voice activity (determined at the encoder). If the signal does not exhibit voice activity instead the decoder - and the renderer in case of ParamISM mode – generates comfort noise signals using a multi-channel signal generator depending on the information on the background noise that was determined from the bitstream in the most recent received SID frame.

The ISM format decoder in the DTX operation follows the structure of the ISM decoder in active frames from Figure 6.6-1 though there are a few differences when SID or NO_DATA frames are decoded. In case of the SID frame, which has a structure from Figure 5.6-6, the bitstream demultiplexer extracts a) the number of streams N_{ISM}, b) spatial information including the ISM mode information, c) ISM common signalling, d) metadata, and e) CNG core-coder parameters which encode information on a background noise of one of the transport channels and control parameters for CNG. More details are provided in following clauses.

6.6.4.2 CNG control parameter decoding

For generating spatial comfort noise, coherence values between the dominant object/transport channel signal and the other object/transport channel signals are transmitted in every SID frame if the number of objects is bigger than one. Decoding of the coherence values is done analogously to the encoding process. First, the SCE_ID is read from the bitstream. It indicates for which of the object/transport channel signals (the “dominant” signal) the information on the background noise was encoded in the SID frame. Then, the respective coherence values are read for all object/transport channel signals in sequential order. For the dominant signal, i.e. the one with index SCE_ID, no coherence value is read from the bitstream, but instead it is set to one. The dequantized coherence values for each object/transport channel signal are then

$$\hat{C}_{SCE\_ID,j} = \begin{cases} 1, if\ j = SCE\_ID \\ \frac{I_{C_{SCE\_ID,j}}}{2^4 - 1}, otherwise \end{cases} \quad (6.6\text{-}41)$$

Where $I_{C_{SCE\_ID,j}}$ is the respective index value read from the bitstream for signal j.

6.6.4.3 Metadata Dequantization and Decoding

The metadata processing module is supplied with the coded metadata of the transmitted N_{ISM} audio streams and the ISM common signaling and dequantize the metadata for the audio streams. At the decoder, an inverse processing to the encoder processing from clause 5.6.6.4 is used while a certain adjustment to the decoded metadata values is used.

At the ISM decoder, the SID frames are received at a certain rate (8 frames by default), resulting in a possibility that received MD parameter values are changed between SID frames with a large step. In order to avoid subjective artifacts from this, the metadata processing module adjusts the MD parameter values at the ISM decoder such that the MD parameter value differences are lowered between frames. Specifically, an interpolation between the true decoded and dequantized current frame MD parameter value and the previous frame MD parameter value is applied in certain frames following the SID frame. This results in the MD parameter values evolving more smoothly while the smoothing is applied in several CNG frames, or several active frames, or several CNG and active frames following an SID frame.

The adjustment, or smoothing, is applied on each MD parameter such that the maximum difference (step) of a MD parameter between two adjacent frames is not more than a given threshold. Let's suppose the current decoded and dequantized MD parameter value of azimuth being θ_{dec} for one stream and the maximum smoothing step for azimuth difference being Δ_θ. Note that the stream index *n* in the metadata parameter is omitted in this clause for simplification while it is supposed that the same adjustment is done for every metadata parameter (i.e. azimuth and elevation included in the SID payload) of every stream. Then:

$$\theta_{dec} = \begin{cases} \theta_{last} + \mathrm{sgn}(\theta_{true} - \theta_{last}) \cdot \Delta_\theta & \text{if } |\theta_{true} - \theta_{last}| > \Delta_\theta \\ \theta_{true} & \text{otherwise} \end{cases} \quad (6.6\text{-}42)$$

where θ_{true} is the transmitted quantized azimuth (a quantized MD parameter value in general) in the SID frame and θ_{last} is the azimuth (a MD parameter value in general) in the preceding frame and updated at the end of each frame decoding as $\theta_{last} = \theta_{dec}$. The azimuth smoothing step value is set to $\Delta_\theta = 5$ and the elevation smoothing step value to $\Delta_\varphi = 5$. Further, the operation $\mathrm{sgn}(x)$ in Equation (6.6-42) is a mathematical expression

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (6.6\text{-}43)$$

Further, the maximum number of frames to apply the smoothing step is set to a given threshold value D_{maxx}. Specifically, it is not limited in inactive segments but limited in active segments to $D_{max} = 5$ frames while the same value of parameter D_{max} is used for all metadata parameters.

Moreover, the smoothing step is skipped in active frames when the absolute value of the difference between the current frame quantized MD parameter value θ_{true} and the previous frame decoded MD parameter value θ_{last} is higher than the smoothing step value Δ_θ multiplied by the threshold value D_{max}. For example, in case of the azimuth MD parameter, it means that:

$$\text{if } |\theta_{true} - \theta_{last}| > D_{max} \cdot \Delta_\theta \text{ then skip the smoothing} \quad (6.6\text{-}44)$$

The relation (6.6-44) also applies to the elevation MD parameter and it is used to prevent wrong smoothed MD parameter estimation when the true value of the MD parameter is changing significantly from frame to frame and the number of frames between the last SID frame and the active frame is low (in general lower than D_{max}).

6.6.4.4 ISM DTX core decoding and CNG

Comfort noise generation for ISM DTX employs a multi-channel signal generator similar to the one described in clause 6.3.5.2.3. In case of ParamISM, it generates two channels of comfort noise using a random generator running with different seeds, controlled by the coherence values encoded in the SID frame and shaping the random noise using FD-CNG as in EVS. In case of DiscISM, a comfort noise signal for each object is generated.

In case of SID frame, the core-decoder SID bitrate of 2.4 kbps is assigned to the SCE core-decoder corresponding to the dominant object/transport channel signal. For that SCE core-decoder, a mono SID frame is read and decoded from the IVAS SID bitstream using FD-CNG SID decoding as described in clause 6.7.3 of [3]. The remaining SCE core-decoders are assigned a bitrate of 0 kbps and no data is read from the SID bitstream for these core-decoders and no CNG parameters are updated for them. Thus, comfort noise generation in the non-dominant SCE core-decoders relies on the background noise data that was estimated in the decoder, see clause 6.7.3.2 of [3]. In case of a NO_DATA frame, all core-decoders are assigned 0 kbps and no bitstream data is read at all. As the dominant object/transport channel was determined in the encoder by choosing the signal with the highest long-term energy, the dominant signal usually has the highest energy in the respective frame. However, as the selection criterion is long-term energy and not current-frame

energy, this assumption does not always hold. Also, as the assignment of dominant object/transport channel can vary over time and background noise parameters estimated at the decoder are used in CNG for the non-dominant objects/transport channels, it is possible that the generated comfort noise for a non-dominant SCE core-decoder has higher energy than the comfort noise that is generated for the dominant object/transport channel's SCE. This can be problematic if the background noise level changes over time while at the same time decoder-side noise estimation is not or not often enough triggered. To avoid too loud comfort noise levels in the non-dominant SCEs, the energy of their respective FD-CNG noise level array is limited to the energy of the noise shape vector of the dominant object/transport channel. The dominant SCE core-decoder is always run first. After decoding is done for the dominant SCE, the energy in the noise level array is calculated for all objects as

$$E_{noise,\ i} = \sum_{j=0}^{L_{SID}-1} (N_{i,FD-CNG}^{[CNG]})^2\,(j). \tag{6.6-45}$$

Here, $N_{i,FD\text{-}CNG}^{[CNG]}$ denotes the noise level array used in FD-CNG for SCE *i*, as defined clause 6.7.3.3 of [3] (there denoted without the *i* subscript). If any noise level energy for an SCE is higher than the noise level energy of the dominant SCE, the corresponding noise level array is scaled so that the energy of the scaled array is equal to that of the dominant SCE's noise level energy:

$$N_{i,FD-CNG}^{[CNG]}(j) := \begin{cases} \sqrt{\frac{E_{noise,SCE_{ID}}}{E_{noise,i}}}\, N_{i,FD-CNG}^{[CNG]}(j), if\ E_{noise,i} > E_{noise,SCE_{ID}} \\ N_{i,FD-CNG}^{[CNG]}(j), otherwise \end{cases}. \tag{6.6-46}$$

Afterwards, the remaining SCE core-decoders are run in sequential order (skipping the one containing the dominant object transport channel signal as it was already decoded).

The CNG mechanism in the ISM DTX decoder aims to both recreate the spatial and the spectral characteristics of the background noise in the encoded signal. The spatial impression is approximated by synthesizing comfort noise in each SCE core-decoder with the same coherence between the respective object/transport channel signal and the dominant object/transport channel signal as estimated in the encoder. Intermediate noise signals are generated by a multi-channel generator using the information on the background noise that is encoded in the SID frame or the decoder-side estimated noise shape (for the non-dominant object/transport channel signals) and the respective broadband coherence value as a control parameter. Spectral shaping of the intermediate noise signals is then done using FD-CNG from EVS to generate the final decoded comfort noise signals.

The multi-channel generator operates similarly to the stereo CNG mechanism used in MDCT-based stereo DTX decoding. It generates an intermediate noise signal for each object or transport channel by weighted mixing of different noise signals with the mixing being controlled by the transmitted coherence values decoded from the SID frame. In case of ParamISM, always two channels of noise need to be generated and the same approach as described in clause 6.3.5.2.3 is applied. Thus, the intermediate noise signal for the first transport channel is generated according to equation (6.3-183) and the second transport channel is generated according to equation (6.3-184), with $coh := \hat{C}_{SCE\_ID,j}, j \neq SCE\_ID$. In case of DiscISM mode a similar approach is used that also addresses the case of more than two noise signals to be generated. For each object, the intermediate noise signal is generated by mixing two noise signals, one generated with the same noise seed as was used for the dominant object and one generated with an object-individual noise seed. The mixing process is controlled by the respective decoded coherence value. CNG is applied in the frequency domain by generating random Gaussian noise separately in the real and imaginary parts of the FFT coefficients as described in clause 6.7.3.3.2 of [3], but instead of just filling the real and imaginary parts of each FFT coefficient with one value generated by a random gaussian noise source, two noise sources are mixed according to:

$$N_{j,mixed}(i) = \sqrt{1 - \hat{C}_{SCE\_ID,j}}\ N_j(i) + \sqrt{\hat{C}_{SCE\_ID,j}}\ N_{SCE\_ID}(i). \tag{6.6-47}$$

Here, *j* is the object index and N_j denotes the random noise source that is individual to the respective object's SCE. Note that $\hat{C}_{SCE\_ID,j} = 1$ for *j = SCE_ID*. That means that for the dominant object, only one noise signal is used. For all other objects, this same noise signal is mixed with an individual one. The weighting of the noise signals, controlled by the decoded coherence values, allows for the coherence between the dominant and the non-dominant objects for the generated comfort noise to be similar to the coherence measured on the encoder side.

The intermediate noise signals for each object/transport channel are then synthesized and shaped using FD-CNG as used in EVS by applying the steps described in clause 6.7.3.3.3 to 6.7.3.3.4 of [3] with the following addition. For all non-dominant SCE core-decoders, FD-CNG synthesis uses the decoder-side estimates of the spectral shape of the background noise. This estimation works the same as in described in clause 6.7.3.2 of [3]. This decoder-side noise estimation is done only on FFT bands and not on the higher CLDFB bands. For the dominant SCE, the CLDFB band estimates are read from the SID bitstream, but for the other SCEs, the noise levels in these bands can either be not

available (if this SCE has not received a noise shape update in an SID yet) or be out of date (if this SCE has received a noise shape update from an SID before, but not recently). To avoid using non-meaningful data for these upper bands in CNG, values for these upper bands are extrapolated from the lower band values. This is simply done by assigning the value of the highest decoder-side estimate band to all upper bands according to:

$$N_{i,FD-CNG}^{[CNG]}(k) = N_{i,FD-CNG}^{[CNG]}\left(j_{min}^{[SID]}\left(L_{SID}^{[FFT]}\right)-1\right), j_{min}^{[SID]}\left(L_{SID}^{[FFT]}\right) \le k < j_{max}^{[SID]}(L_{SID}-1) \qquad (6.6\text{-}48)$$

where $N_{i,FD-CNG}^{[CNG]}, j_{min}^{[SID]}, L_{SID}^{[FFT]}, j_{max}^{[SID]}, L_{SID}$ are defined clause 6.7.3.2 of [3]. This step is applied before shaping the respective intermediate noise signal, i.e. before the processing described in clause 6.7.3.3.2 of [3].

If the first active frame after a CNG period was coded in the MDCT domain (i.e. not using LP-based coding), the same smoothing approach as described in 6.3.5.2.4 is used.

6.6.5 ISM PLC

ISM PLC consists of a metadata and an audio transport-channel part. The metadata PLC processing saves the object positions/orientations/radiuses/panning gains from the last good frame and holds them until the next good frame is received. The transport-channel processing for PLC is performed inside every core-decoder according to clause 6.2.2.

6.6.6 ISM bitrate switching

6.6.6.1 Metadata handling in bitrate switching

The direction metadata is encoded independent of the bitrate conditions. However, gain metadata coding operation is only active for bitrates higher than 64 kbps. Therefore, a mechanism is introduced to ensure smooth transition between the detail level of metadata in case of bitrate switching. Two detail levels are described for this operation.

In the case of level 1 detail of metadata, only the common metadata (azimuth and elevation) decoder is active. If the second or extended level of metadata detail is used, the radius r and the orientation angles yaw and $pitch$ are also decoded with extended metadata decoder. Note that the first level of detail, azimuth and elevation is a subset of the extended level of detail. The level of detail for metadata to be decoded is indicated by the parameter received via bitstream, bt_{extmd}. If bt_{extmd} is set, the encoded metadata includes the extended metadata, level 2 detail.

A variable to track the state of the level of metadata is introduced as ac_{extmd}. Active state variable, ac_{extmd}, determines the detail level of the metadata to be used for rendering. At the beginning of the decoding process, ac_{extmd} is set to -1 to indicate the decoding process of the first frame.

A counter cnt_{extmd} is introduced to keep track of the number of received frames from different levels of detail. The level 1 detail of metadata is always encoded and decoded independent of the transmission conditions. However, some parts of the level 2 detail are only encoded for high bitrates (≥ 64 kbps). In the event of the changes in the bitrate conditions, the metadata change counter, cnt_{extmd} keeps track of the duration for such changes before updating the active state, ac_{extmd}. The counter, cnt_{extmd} controls the status of the metadata detail state and allows the updates in the metadata memory accordingly via the variable ac_{extmd}.

Before the decoding process, all metadata parameters (level 1 and level 2) are set to their default values and stored in the metadata memory.

For the first received frame, indicated by $ac_{extmd} = -1$, cnt_{extmd} is set to 'zero' and the state ac_{extmd} is updated with the bitstream level, bt_{extmd}:

$$cnt_{extmd} := 0 \qquad (6.6\text{-}49)$$

$$ac_{extmd} := bt_{extmd} \qquad (6.6\text{-}50)$$

If bt_{extmd} indicates level 2 detail, the metadata (both level 1 and 2) is decoded according to the level encoded, their values are used for rendering, and stored in the memory.

In the case of bt_{extmd} indicating level 1 detail, the level 2 metadata parameters (yaw, pitch, and radius) are reset to their default values to be used in the rendering process. The metadata memory is updated with the default values of level 2 metadata in this case. The level 1 metadata parameters are updated according to the bitstream in the memory and those values are used for rendering.

For the static bitrate condition where the metadata detail state currently being used is the same with the bitstream metadata detail level, i.e., $ac_{extmd} == bt_{extmd}$, the operations follow the logic in the previous clause with Equations (6.6-49) and (6.6-50).

If the received metadata detail level is different from the current state, $ac_{extmd} \neq bt_{extmd}$ when $ac_{extmd} \neq -1$, it corresponds to the change in the bitstream conditions. In this case, cnt_{extmd} is incremented by one:

$$cnt_{extmd} := cnt_{extmd} + 1. \quad (6.6\text{-}51)$$

Following the incrementation, cnt_{extmd} is subjected to a threshold of 5 to see if the change is persistent over a period (5 frames in this case). If counter $cnt_{extmd} < 5$, meaning the change has been effective for less than 5 frames, the metadata values from the memory are used for rendering. The update on the level 2 metadata parameters is withheld for a short time window. When the counter cnt_{extmd} reaches the threshold, meaning $cnt_{extmd} = 5$, Equations (6.6-49) and (6.6-50) are applied followed by the same operations in the first frame and the renderer uses the received and decoded metadata. In case level 1 metadata is received, the default values are used for the extended metadata in level 2. The renderer uses the received and decoded values of level 2 detail metadata when bt_{extmd} indicates level 2 detail. The metadata memory is updated with the metadata values used for rendering.

6.6.6.2 Codec reconfiguration in bitrate switching

Bitrate switching in ISM format mainly cause reconfiguration of the underlying SCE core-decoders as the bitrate might influence selection of the core-decoder tools to be used. If a bitrate switch causes the ISM mode to change from DiscISM to ParamISM, or the other way around, more reconfiguration is needed as the number of audio channels – and thus also the number of SCE core-decoder instances – changes. Note that this can only happen if the number of objects is 3 or 4, namely when switching from a bitrate that is lower than 48 kbps to a bitrate of 48 kbps or higher (or the other way around). The performed reconfigurations are:

- Re-setting the core-decoder bitrates according to clause 5.6.2.3.1 step 1 and 2. This results in an initial equal bitrate distribution between the SCEs. The remaining steps are skipped.
- Reconfiguration of the SCE core-decoders. This can include deallocation of old SCE core-decoders and/or setting up new ones in case the bitrate switch causes the ISM mode to change. This also includes reconfiguration and initializations of the HP20 output filters. For 3 or 4 objects, the number of transport channels can change when switching from a DiscISM bitrate to a ParamISM bitrate or vice versa. Memories of the high pass filters for persisting channels are not filled with zeros but carried over for signal continuity.
- Selecting the renderer if the ISM mode changes based on [7], clause 5.1. Depending on the output config, changing the renderer can result in reconfiguration of HRTF data or the respective reverb module.
- Reconfiguring the JBM. This is dependent on whether the renderer changes or not, see 6.6.8.
- Initializing or deallocating buffers and structures needed especially for ParamISM if the ISM mode changes. This includes initializing the ParamISM renderer structures and initializing DirAC-related structures if switching to a ParamISM bitrate. Then, also the prototype matrix and the interpolator states are initialized, see clause.
- Reconfiguration of the CLDFB instances to match the current number of transport channels and the core-decoder setup.

6.6.7 ISM output format conversion

6.6.7.1 Overview

The ISM format supports the following output audio configurations: mono, stereo, multichannel (5.1, 7.1, 5.1+2, 5.1+4, 7.1+4), SBA (FOA, HOA2, HOA3), binaural, stereo panning, and external. Several different rendering tools depending on the ISM mode and output audio configuration are used while details about particular rendering tools can be found in clause 7.2.

Rendering of decoded ISM output audio in DiscISM mode is realized using rendering tools listed in Table 6.6-1.

Table 6.6-1: Rendering tools in DiscISM mode

Output configuration	Rendering tool	Described in clause
Mono	Passive mono downmix	6.6.7.2
Stereo	ISM renderer	7.5.5.1
MC	ISM renderer	7.5.5.1
SBA	ISM to SBA renderer	7.5.2.1
Binaural	TD renderer	7.2.2.2
Binaural + BRIR	ISM renderer → CRend renderer	7.2.2.5
Binaural + room	TD renderer	7.2.2.2
EXT	direct output of TCs	

Rendering of decoded ISM output audio in ParamISM mode is realized using rendering tools listed in Table 6.6-2.

Table 6.6-2: Rendering tools in ParamISM mode

Output configuration	Rendering tool	Described in
Mono	Passive mono downmix	6.6.7.2
Stereo	direct output of TCs	6.6.2.4.2
MC	ParamISM renderer	6.6.2.4.3
SBA	ParamISM renderer	6.6.2.4.3
Binaural	Parametric renderer + loudness correction	7.2.2.3
Binaural + BRIR	Parametric renderer + loudness correction	7.2.2.3
Binaural + room	Parametric renderer + loudness correction	7.2.2.3
EXT	Rendering to EXT	6.6.2.4.4

Finally, a rendering to an external audio configuration is supported which means that there are outputted separately ISM audio streams and ISM metadata associated to every audio stream.

The loudness correction in ParamISM applies a gain of 1.3657 to the decoded transport channels prior to binaural rendering (see also clause 6.6.2.4.2 for some additional remarks).

6.6.7.2 Mono downmix

To generate mono output for one time frame, a proto signal is first computed from the transport channels, i.e., the transmitted SCEs:

$$y_{proto}(n) = \sum_{i=1}^{N_{TC}} x_i(n) \tag{6.6-52}$$

Here, $n = 1, \dots, N$ is the time sample index, N is the number of time samples per frame, N_{TC} denotes the number of transport channels, i is the transport channel index, and $x_i(n)$ describes the i-th transport channel.

Furthermore, an input energy E_{in} and a proto energy E_{proto} are calculated with the help of local energies E'_{in} and E'_{proto} according to:

$$E'_{in} = \sum_{n=1}^{N} \sum_{i=1}^{N_{TC}} x_i^2(n) \tag{6.6-53}$$

$$E'_{proto} = \sum_{n=1}^{N} y_{proto}^2(n) \tag{6.6-54}$$

$$E_{in} = \alpha_{DMX} E_{in}^{prev} + (1 - \alpha_{DMX}) E'_{in} \tag{6.6-55}$$

$$E_{proto} = \alpha_{DMX} E_{proto}^{prev} + (1 - \alpha_{DMX}) E'_{proto} \tag{6.6-56}$$

The smoothing coefficient is defined as $\alpha_{DMX} = 0.95$ and both E_{in}^{prev} and E_{proto}^{prev} are initialized to 0 for the first fame to be processed. Afterward, the input and proto energies E_{in} and E_{proto} of the current frame are retained and stored as E_{in}^{prev} and E_{proto}^{prev} for use in the next frame to be processed.

With the maximum allowed downmix gain $g_{DMX,max} = 4$ and $\epsilon = 10^{-15}$, an equalization factor is determined from the input and proto energies:

$$g_{eq} = \min(g_{DMX,max}, \sqrt{E_{in}/(E_{proto} + \epsilon)}) \tag{6.6-57}$$

The mono downmix is finally obtained by equalizing the proto signal according to:

$$y_{mono}(n) = g_{eq} y_{proto}(n) \quad (6.6\text{-}58)$$

6.6.8 ISM decoding with TSM

The render granularity for ParamISM is always 1.25 ms. The render granularity for discrete ISM depends on the output format and the used renderer. In general it is also 1.25 ms, if the output configuration is any binaural one, the renderer granularity is 5 ms.

The decoding steps done before the TSM are:

- meta data and transport channel decoding
- if the output is mono, the passive mono downmix
- if the frame is coded with ParamISM and the renderer is binaural parametric or binaural parametric room, the loudness correction

All remaining processing steps are carried out after the TSM.

If a bit rate switch is occurring and the render granularity changes from 5 ms to 1.25 ms a render flush according to 6.2.7.5.1 is done.

The transport channel buffer is reconfigured on rate switch if the number of channels to be processed by the TSM or the render granularity changes.

If a rate switch is occurring and the render granularity changes from 1.25 ms to 5 ms, a rendering with discarded samples according to clause 6.2.7.5.3 take place.

The decoder shall keep track of the already rendered samples, time slots, and subframes.

6.7 Multi-channel audio (MC) decoding

6.7.1 LFE channel decoding

6.7.1.1 Overview

Unless a received frame is marked with the BFI flag as lost, MDCT LFE decoding takes place as described in clauses 6.7.1.2-6.7.1.6. Even in that case, buffering of the previously synthesized LFE channel (prior to LP filtering) takes place to prepare for a potential future frame loss. If the current frame is marked with the BFI flag, PLC processing takes place as described in clause 6.7.1.7.

6.7.1.2 Bitstream decoding

MDCT based LFE decoding essentially applies the inverse of the encoder side operations in reverse order.

The LFE decoding starts with parsing the bit fields used for coded LFE frames as defined in clause 5.7.2.2, Table 5.7-4. The first step is the parsing of the first bit to identify potential usage of silence frame coding. If this bit is found to be set, two sets of all-zero MDCT coefficients $\hat{X}_1(k)$ and $\hat{X}_2(k)$, as defined in eqs. (6.7-11) and (6.7-12), are passed to further processing as described under clause 6.7.1.4.

Otherwise, these MDCT coefficients are unequal zero and decoded as follows. The next bits represent the applied quantization strategy ϵ and the coded shift value s. The decoded shift value is obtained by offsetting the (unsigned) integer value corresponding to the 5-bit code word representing s by 4 for fine quantization strategy ($\epsilon = 0$). For coarse quantization strategy ($\epsilon = 1$), the integer value is taken as such without offset as decoded shift value.

Next is to decode the MDCT coefficient signs. For fine quantization strategy ($\epsilon = 0$), 16 sign values are extracted, for coarse quantization strategy ($\epsilon = 1$), 12 sign values are extracted.

Next is to parse the bit representing the applied coding strategy and subsequently to parse and decode the MDCT coefficient magnitudes. In case the coding strategy bit is set, the MDCT coefficient magnitudes are base-2 decoded. For each MDCT coefficient, the decoded magnitude value is obtained by parsing the number of bits from the bitstream corresponding to the subband group and used quantization strategy, according to Table 5.7-3, and taking the corresponding unsigned integer value. Otherwise, the MDCT coefficient magnitudes are obtained by arithmetic decoding.

6.7.1.3 MDCT coefficient reconstruction

With the decoded MDCT coefficient magnitudes, signs and shift value available, next the MDCT coefficients are reconstructed. Firstly, the signs are restored. In case the sign bit of a given MDCT coefficient is set, the corresponding magnitude value is negated followed by decrementing it by one. Otherwise, the magnitude value is taken as is. This step results in normalized MDCT coefficients $\hat{X}_{q,i}$ for the relevant subband groups. Next, the normalized MDCT coefficients are scaled up to the reconstructed MDCT coefficients using the negated shift value as follows:

$$\hat{X}_i = \hat{X}_{q,i} 2^{-s/4} \text{ with } i = \begin{cases} 0 \dots 15 \text{ for } \epsilon = 0 \\ 0 \dots .11 \text{ for } \epsilon = 1 \end{cases} .$$

Finally, the reconstructed MDCT coefficients are rearranged to two sets of MDCT coefficients $\hat{X}_1(k)$ and $\hat{X}_2(k)$ including zero values for the uncoded MDCT coefficients of frequencies above the relevant subband groups. This yields the following sets of MDCT coefficients for the first and second encoded LFE subframes:

$$\hat{X}_1(k) = \begin{cases} \{\hat{X}_0, \hat{X}_1, \hat{X}_4, \hat{X}_5, \hat{X}_8, \hat{X}_9, \hat{X}_{12}, \hat{X}_{13}, 0, \dots 0\} & \text{for } \epsilon = 0 \\ \{\hat{X}_0, \hat{X}_1, \hat{X}_4, \hat{X}_5, \hat{X}_8, \hat{X}_9, 0, \dots 0\} & \text{for } \epsilon = 1 \end{cases} \text{ and} \quad (6.7\text{-}1)$$

$$\hat{X}_2(k) = \begin{cases} \{\hat{X}_2, \hat{X}_3, \hat{X}_6, \hat{X}_7, \hat{X}_{10}, \hat{X}_{11}, \hat{X}_{14}, \hat{X}_{15}, 0, \dots 0\} & \text{for } \epsilon = 0 \\ \{\hat{X}_2, \hat{X}_3, \hat{X}_6, \hat{X}_7, \hat{X}_{10}, \hat{X}_{11}, 0, \dots 0\} & \text{for } \epsilon = 1 \end{cases} . \quad (6.7\text{-}2)$$

6.7.1.4 Inverse MDCT

The two decoded sets of MDCT coefficients $\hat{X}_1(k)$ and $\hat{X}_2(k)$ representing two LFE subframes are subsequently transformed to time domain according to the following inverse MDCT procedure.

In general, the IMDCT transforms the vector $X(k), k = 0 \dots L/2 - 1$ with $X(k)$ either being $\hat{X}_1(k)$ or $\hat{X}_2(k)$ back to a time domain vector $x(n), n = 0 \dots L - 1$ according to the following equation (except for an IMDCT gain scaling factor that is treated separately):

$$x(n) = \sum_{k=0}^{L/2-1} \mathrm{X}(k) \cos\left[\frac{2\pi}{L}\left(n + \frac{1}{2} + \frac{L}{4}\right)\left(k + \frac{1}{2}\right)\right], \; n = 0 \dots L - 1 \; . \quad (6.7\text{-}3)$$

As in the forward MDCT case described in clause 5.7.2.2, the IMDCT is efficiently computed by means of inverse FFT. This requires the following steps, described for a generic vector $X(k)$ of $L/2$ MDCT coefficients, where L equals 960, 64 or 320 for an output audio sampling frequency of, respectively, 48000 Hz, 32000 Hz or 16000 Hz.

Firstly, pre-rotation operation is carried out with the twiddling factors also used in the forward MDCT calculation. This operation converts the MDCT coefficients to a buffer of $L/4$ complex DFT coefficients as follows:

$$\tilde{X}(k) = X(L - 2k - 1)\mathrm{e}^{\mathrm{j}\frac{2\pi}{L}\left(k+\frac{1}{8}\right)} + jX(2k)\mathrm{e}^{\mathrm{j}\frac{2\pi}{L}\left(k+\frac{1}{8}\right)} \; , \; k = 0 \dots \frac{L}{4} - 1 \; . \quad (6.7\text{-}4)$$

After this preparation, the complex DFT coefficients $\tilde{X}(k)$ rescaled with an IMDCT gain scaling factor and then transformed to time domain using a complex valued FFT, yielding a vector of $L/4$ complex samples $\tilde{x}(n)$.

After that, post-rotation operation is carried out with the same twiddling factors as used prior to the inverse FFT. In addition, a buffer of $L/2$ real-valued samples $\tilde{\tilde{x}}(n)$ is created by interlacing real and imaginary parts of the twiddled samples with negated real parts and the imaginary parts in reverse order:

$$\begin{aligned} \tilde{\tilde{x}}(2n) &= -\Re\mathrm{e}\left(\tilde{x}(n)e^{j\frac{2\pi}{L}\left(n+\frac{1}{8}\right)}\right) \\ \tilde{\tilde{x}}\left(\frac{L}{2} - 1 - 2n\right) &= \Im\mathrm{m}\left(\tilde{x}(n)e^{j\frac{2\pi}{L}\left(n+\frac{1}{8}\right)}\right) \end{aligned} \; , \; n = 0 \dots \frac{L}{4} - 1 \; . \quad (6.7\text{-}5)$$

The final step of the IMDCT computation is to generate a block $x(n)$ of L output samples by sign flipping and/or time reversing the first and second $L/4$ portions of $\tilde{\tilde{x}}(n)$:

$$x(n) = -\tilde{\tilde{x}}\left(\frac{L}{4} - 1 - n\right) \ , \tag{6.7-6}$$

$$x\left(n + \frac{L}{4}\right) = \tilde{\tilde{x}}(n) \ , \tag{6.7-7}$$

$$x\left(n + \frac{L}{2}\right) = \tilde{\tilde{x}}\left(n + \frac{L}{4}\right) \ , \tag{6.7-8}$$

$$x\left(n + \frac{3L}{4}\right) = \tilde{\tilde{x}}\left(\frac{L}{2} - 1 - n\right) \ , \tag{6.7-9}$$

with index $n = 0 \ldots \frac{L}{4} - 1$.

6.7.1.5 Windowing and overlap-add

The two IMDCT transformed subframe vectors $\hat{x}_1(n)$ and $\hat{x}_2(n)$ are subsequently windowed with the KBD window also used by the MDCT encoder, see clause 5.7.2.2. This yields the windowed subframe vectors $\hat{x}_{w,1}(n)$ and $\hat{x}_{w,2}(n)$ where it is notable that the portions corresponding to the zero-pad portions (1 ms at the start and the end of the window) of the KBD window are zero and do not need to be computed.

Subsequent overlap-add operation generates the reconstructed frame as follows:

$$\hat{x}_{rec}(n) = \begin{cases} \hat{x}'_{w,2}(n + L - l_0 - l_{fad}) + \hat{x}_{w,1}(n + l_0), & n = 0 \ldots l_{fad} - 1 \\ \hat{x}_{w,1}(n + l_0), & n = l_{fad} \ldots l_{fad} + 2l_0 - 1 \\ \hat{x}_{w,1}(n + l_0) + \hat{x}_{w,2}(n - l_0 - l_{fad}), & n = l_{fad} + 2l_0 \ldots 2l_{fad} + 2l_0 - 1 \\ \hat{x}_{w,2}(n - l_0 - l_{fad}), & n = 2l_{fad} + 2l_0 \ldots L - 1 \end{cases} \tag{6.7-10}$$

wherein, $l_0 = L/20$ denotes the size of the zero-pad portion, $l_{fad} = 8L/20$ denotes the size of the cross-over portion and $\hat{x}'_{w,2}$ denotes the second subframe of the preceding frame.

6.7.1.6 Output LPF and delay adjustment

The LFE output buffer is finally generated by low-pass filtering the reconstructed frame using the same 4th order Butterworth filter as in the encoder, see clause 5.7.2.1 and by delaying the LP filtered LFE signal to make it time aligned with the other channels of the decoded multi-channel signal.

6.7.1.7 LFE PLC

6.7.1.7.1 Overview

LFE frames marked by the BFI flag as lost or unusable are not decoded using the regular LFE decoding scheme. Instead, a packet loss concealment mechanism for the LFE channel is applied following a linear predictive approach in time domain. The general concept is to reconstruct the samples of a substitution frame by operating an all-pole filter that is derived through bandwidth sharpening from an LPC synthesis filter. The filter is tuned in on a portion of the previously reconstructed signal and is then operated as a resonator by keeping it ringing for the samples of the substitution frame. To reduce complexity of the computations and memory needs, operations are done in down-sampled domain at a sampling frequency of 1600 Hz. For the case of burst errors with multiple successive frame losses, an attenuation/muting mechanism is provided.

6.7.1.7.2 Burst loss determination

In case a frame is marked with the BFI flag, an LFE burst loss counter is incremented. If, after the increment, the counter value is less than a threshold of LFE_PLC_MUTE_THR=10, then PLC operation without muting behavior takes place. Otherwise, a slightly modified PLC procedure with muting behavior is applied with an attenuation of approximately 3dB per frame effective starting from the 10th successive frame marked with the BFI flag.

Frames not marked with the BFI flag cause the LFE burst loss counter to be set to zero. Otherwise, if the LFE burst loss counter exceeds the threshold of 10, it is set to 10.

6.7.1.7.3 LFE substitution frame processing

6.7.1.7.3.1 Buffering of previously synthesized LFE channel

PLC for the LFE channel relies on a buffer of 240 samples of the previously synthesized LFE signal in down-sampled domain. To maintain that buffer in a prepared state for the case the next frame is marked as lost or unusable, each synthesized LFE frame, either obtained through regular LFE decoding or LFE PLC, is firstly decimated to the sampling frequency of 1600 Hz and then pushed from right-hand side into the buffer while discarding the corresponding number of 32 oldest samples. As the LFE channel contents can assumed to be sufficiently band limited, the decimation merely takes every n^{th} sample of the LFE signal at original sampling rate, starting from the zeroth sample. The decimation factor n equals the original sampling rate (48000, 32000 or 16000 Hz) divided by 1600 Hz.

6.7.1.7.3.2 Resonator filter calculation

Upon raised BFI flag, the substitution frame processing begins with the calculation of an initial set of LPC synthesis filter coefficients. To that end, the 240 samples of the previously synthesized and down-sampled LFE signal are first windowed with a 240-point symmetric Hamming window. Next, the autocorrelations of the windowed signal $s_w(n)$ are computed by

$$r_c(k) = \sum_{n=k}^{239} s_w(n)s_w(n-k)\,, \quad k = 0 \ldots 20\,. \tag{6.7-11}$$

If $r_c(0)$ is below a threshold of 0.0000024, then a buffer prepared to store the substitution frame is zeroed and the procedure continues as described in clause <td alias signal calculation>.

Otherwise, the procedure continues with solving the following equation system using the Levinson-Durbin recursion:

$$\sum_{k=1}^{20} a_k r(|i-k|) = -r(i), \quad i = 1 \ldots 20\,. \tag{6.7-12}$$

The Levinson-Durbin algorithm is described in clause 5.1.9.4 of [3].

The resulting LPC synthesis filter is subsequently modified by bandwidth sharpening. To that end and in case no attenuation/muting takes place, the filter coefficients are modified using a bandwidth sharpening factor γ according to

$$a_{\gamma,k} = a_k \gamma^k, k = 1 \ldots 20\,. \tag{6.7-13}$$

6.7.1.7.3.3 Calculation of bandwidth sharpening factor

The bandwidth sharpening is controlled by the sharpening factor $\gamma = 1 + \delta$, where δ is used as a helper variable. δ is successively iterated using a nested interval technique such that the resulting modified filter is as close as possible at the stability limit but still stable.

6.7.1.7.3.4 Stability test of modified filter

Part of the procedure is the determination if the all-pole filter with modified coefficients is stable. The corresponding test relies on determining if the all-pole filter with coefficients modified according to equation (6.7-13)) and a specific value of δ_{test} is stable. To that end, the modified filter coefficients are firstly calculated according to

$$a_{test,k} = a_k(1 + \delta_{test})^k, k = 1 \ldots 20\,. \tag{6.7-14}$$

Subsequently, stability is checked by first converting the linear prediction coefficients $a_{test,k}$ to reflection coefficients $refl_{test,k}$ using a backwards Levinson Durbin recursion as described in clause 5.2.6.1.4.1 of [3]. If the absolute value of any of the reflection coefficients is equal or greater than 1, the stability test declares the filter instable, otherwise it is declared stable.

In the following description, the stability test for a set of filter coefficients $a_k, k = 1 \ldots 20$ and a sharpening parameter δ is shortly referred to as check_stab($\boldsymbol{a}, \delta$). It returns the value True in case the filter is declared stable. Otherwise, it returns the value False.

6.7.1.7.3.5 Nested interval search

The nested interval search to find the maximum δ for which the modified filter is still stable is done using the following iteration.

In preparation of the iteration, firstly, the initial interval size for the nested interval search is determined.

Starting with an initial value of $\epsilon = 0.01$, it is checked using check_stab($\boldsymbol{a}, \delta_{test}$) with $\delta_{test} = \epsilon$ if the corresponding all-pole filter is stable. While this is true, ϵ is doubled and the check is repeated. As this loop ends with an ϵ value resulting in an instable filter, the latest doubling is subsequently reverted. To catch the case where already the initial value of $\epsilon = 0.01$ leads to an instable filter, for that case, ϵ is successively halved followed by a stability check until the resulting filter is stable. The result of this preparation procedure is an iteration start value of $\delta_0 = \epsilon$ for which a stable filter is ensured and for which a modified filter with $\delta = 2\epsilon$ is unstable.

The nested interval search is then carried out according to the following pseudo-code with the iteration start value $\delta_{test} = \delta_0$ and initial interval size of $\epsilon = \delta_0/2$. The iteration stops with $\delta = \delta_{test}$ if the latest sharpening parameter δ_{test} provides a stable filter and when the interval size is below an end value of $\epsilon_{stop} = 10^{-5}$.

```
While ( True )
    {
        δ_test = δ_test + ϵ
        stable = check_stab( a, δ_test )
        if ( !stable )
        {
            if ( |ϵ| > ϵ_stop )
            {
                ϵ = −|ϵ|/2
            }
            else
            {
                ϵ = −|ϵ|
            }
        }
        else
        {
            if (|ϵ| < ϵ_stop)
            {
                break
            }
            ϵ = |ϵ|/2
        }
    }
δ = δ_test
```

6.7.1.7.3.6 Modification of bandwidth sharpening factor in case of attenuation/muting

In case attenuation/muting takes place, the filter coefficients are modified using a modified bandwidth sharpening factor that additionally imprints the attenuation behavior. While the previously calculated bandwidth sharpening factor γ essentially results in that the modified LPC synthesis filter produces a sustained substitution signal, for attenuation/muting, γ is additionally scaled to perform attenuation. Scaling γ has the effect that the poles of the modified synthesis filter are moved by the scaling factor inwards the unit circles and that the filter response decays accordingly. To achieve a target attenuation of 3dB per frame (0.02s), given the sampling rate of 1600, the bandwidth sharpening parameter γ is modified with the following factor:

$$\alpha_{mute} = \sqrt[0.02 \cdot 1600]{10^{-3[dB]/20}} = 0.9892647\ . \qquad (6.7\text{-}15)$$

Accordingly, the resulting sharpening factor to be used replaces the previously calculated sharpening factor according to $\gamma = \alpha_{mute}\gamma$.

It is notable that α_{mute} remains unchanged for any subsequent substitution frames of a given frame loss burst. Further attenuation and ultimately muting is accomplished since a new substitution frame is generated based on the already attenuated previous substitution signal.

6.7.1.7.3.7 Generation of substitution signal

With the coefficients of the all-pole resonator filter available, the samples of the substitution frame in subsampled time domain are generated by IIR filtering according to the following filter recursion:

$$\hat{s}(n) = \sum_{k=1}^{20} a_k \gamma^k \hat{s}(n-k)\ ,\ n = 0 \ldots L-1\quad . \qquad (6.7\text{-}16)$$

Note that prior to the filtering, the filter memories are initialized with the most recent samples of the buffered previously synthesized LFE channel signal in down-sampled domain:

$$\hat{s}(n) = s(240 - 20 + n),\ n = -20 \ldots -1 \quad . \tag{6.7-17}$$

The filtering is carried out for L=58 samples in down sampled domain, which accommodates for 32 samples of a frame, 16 samples needed for overlap-add processing in the LFE MDCT decoding/reconstruction framework and 10 samples delay of the lowpass filter used in up-sampling to a sampling rate of 48000 kHz.

After generation of the substitution signal in subsampled domain, it is resampled to a sampling frequency of 48000. This is done in the numerically efficient polyphase implementation using an up-sampling factor of $30 = 48000/1600$. The interpolation lowpass filter is designed using a Kaiser window FIR filter design technique with a cutoff frequency of 800 Hz at the sampling frequency of 48000 Hz. The filter delay is 300 samples at 48000 Hz, corresponding to 10 samples in 1600 Hz down sampled domain. Note, to properly warm up the interpolation filter state memory, it is needed to start the up-sampling 30 samples prior to the end of the previously synthesized and down-sampled LFE signal. In case the output audio sampling frequency is other than 48000 Hz, the substitution signal is subsequently further decimated to that respective sampling frequency.

Subsequently, the substitution signal is windowed and folded as described in clause 5.7.2.2. The two resulting signal buffers then undergo the inverse operation, windowing, and overlap-add with the previously synthesized frame as described in clauses 6.7.1.4 and 6.7.1.5 followed by LP filtering and delay adjustment, as regularly decoded frames. The synthesized LFE channel signal prior to LP filtering is buffered for the event of a future frame loss, as described above in this LFE PLC clause.

6.7.2 Multi-channel MASA (McMASA) decoding mode

6.7.2.1 McMASA decoding mode overview

McMASA decoder receives encoded data comprising one or two transport audio signals, McMASA metadata, and a separated channel audio signal (in case of the separate channel mode). The McMASA decoder decodes the transport audio signals and the separated channel audio signal (as described in clause 6.7.2.3) and the McMASA metadata (as described in clause 6.7.2.2). Then, spatial audio is synthesized using the decoded separated channel audio signal, transport audio signals, and McMASA metadata, as described in 6.7.2.4.

6.7.2.2 McMASA metadata decoding

6.7.2.2.1 Decoding McMASA configuration

McMASA metadata decoder is configured similarly to the MASA metadata decoder detailed in clause 6.5.2 with the adjustments as detailed in the McMASA encoder clause 5.7.3.6.1.

6.7.2.2.2 Spatial metadata decoding

The spatial metadata decoding follows the flow of operations presented for MASA metadata in clause 6.5.3.1 with the following modifications:

- The directional metadata dequantization for 5.1 and 7.1 multichannel input format is performed in 2D.
- The elevation alphabet used at the entropy decoding methods used in EC1 and EC3 count only the positive elevation values
- The EC1 decoding of the azimuth uses the adaptive average, mirroring the adaptation from clause 5.2.4.5.1.4
- The azimuth dequantization uses the variant designed for McMASA case presented in clause 6.2.4.1.2.2

The low bitrate azimuth quantization and encoding of EC3 uses the codebook and method mentioned in clause 6.2.4.3.4.

6.7.2.2.3 LFE-to-total energy ratio decoding

McMASA low frequency effect to total energy ratio is encoded for active frames with 1 bit for 13.2 kbps, with 4 bits for 16.4 kbps and, adaptively using up to 8 bits for bitrates 24.4 kbps and higher. Inactive LFE frames are for all bitrates encoded with 1 bit. See table 5.7-6 in clause 5.7.3.6.3.

LFE-to-total energy ratio for 13.2 kbps is decoded based on the received bit (0/1). If the received bit is (1), the previous LFE-to-total energy ratio $\hat{\Xi}_{prev}$ value is increased with predetermined value (MCMASA_LFE_BETA=0.09). If previous frame's LFE gain bit was also set to 1 (two consecutive active frames), the value is further multiplied with (MCMASA_LFE_THETA=1.3). Thus, LFE-to-total energy ratio for current frame $\hat{\Xi}$ can be obtained as follows if received index is (1):

$$\hat{\Xi} = \begin{cases} 0.09 + \hat{\Xi}_{prev}, if\ prevind == 0 \\ 1.3 * 0.09 + \hat{\Xi}_{prev}, if\ prevind == 1 \end{cases}$$

If received index is (0), the previous quantized LFE gain parameter is multiplied by a damping factor (MCMASA_LFE_ALPHA=0.67)

$$\hat{\Xi} = 0.67 * \hat{\Xi}_{prev}$$

Finally, if $\hat{\Xi} > 1.0$ it is set to maximum of 1.0 and copied to all subframes $\hat{\Xi}(m)$. In practice, one bit is used to bump up or dampen the LFE-to-total energy ratio frame-by-frame basis.

LFE-to-total energy ratio quantizer for 16.4 kbps is using 1 (activity) + 3 bits with scalar quantizer in log_2 domain. The $\hat{\Xi}_{log_2}$ values are within range $[-6.5, \ldots, 0.5]$ with step of 1.0 (see table 5.7-7 in clause 5.7.3.6.3 for codebook). Decoded LFE-to-total energy ratio $\hat{\Xi}_{log_2}$ is converted to linear domain and copied to all subframes $\hat{\Xi}$(m).

Bitrates 24.4 kbps and higher use bitrate adaptive VQ subframe residual coding. The amount of subframe residual bits depends on the decoded scalar LFE-to-total energy ratio. See table 5.7-6 for subframe VQ bit allocation. `McMASA_LFEGain_vectors[4x16]` contains the codebook. The final subframe decoded LFE-to-total energy ratio values are obtained with

$$\hat{\Xi}(m) = 2^{(\hat{\Xi}_{log_2} + CB(4*i+m))}$$

Finally, the linear LFE-to-total energy ratio values $\hat{\Xi}(m)$ for all subframes are limited within range $[0.0\ 1.0]$.

6.7.2.3 McMASA transport audio signal decoding

McMASA transport audio signals are decoded mirrored to the encoding process described in clause 5.7.3.7 including the selection of transmission scheme and bitrate division based on the total codec bitrate. This provides further McMASA decoding steps with the McMASA transport audio signals $s_{trans}(n, i)$ and the separated channel signal $s_{sep}(n)$ accordingly.

6.7.2.4 McMASA rendering

6.7.2.4.1 Binaural rendering

Binaural (with and without a room effect) rendering is performed using the parametric binauralizer and stereo renderer presented in clause 7.2.2.3 (i.e., the spatial audio is synthesized using the methods described therein).

6.7.2.4.2 Multi-channel loudspeaker rendering

6.7.2.4.2.1 Overview

The multi-channel loudspeaker renderer can render multi-channel loudspeaker output signals from decoded McMASA transport audio signal(s) and spatial metadata (i.e., spatial audio is synthesized). In this case, the input to the encoder has been multi-channel loudspeaker signals, and they have been encoded using the McMASA methods (see clause 5.7.3 for details), resulting in the transport audio signals and the spatial metadata (containing spatial audio and coherence parameters). The renderer described in this clause reproduces the sound scene conveyed by the original multi-channel loudspeaker signals using the transport audio signals and the spatial metadata.

The rendering is performed in subframes, where m denotes the subframe index. A subframe contains N_{slots} CLDFB slots (in non-JBM operation, $N_{slots} = 4$, in JBM operation $N_{slots} = 1 \ldots 7$). The data determined at previous calls (i.e., subframes m-1 and earlier) affects the rendering of the present subframe m due to temporal averaging and interpolation.

The fetching of the temporally correct spatial metadata parameter values for the current subframe m so that they are in sync with the audio signals is handled in clause 6.2.7. It operates differently for the JBM and non-JBM use. Fetching the correct spatial metadata values is not discussed in the following, it is assumed that it has already been correctly performed, as described in the aforementioned clause.

There are two operation sub-modes in McMASA, "normal" and "separate-channel" sub-mode. The separate-channel sub-mode is used with bitrates equal to or larger than 64 kbps. The normal sub-mode is used with bitrates smaller than 64 kbps. The rendering in the normal sub-mode and in the separate-channel sub-mode are described in following clauses.

6.7.2.4.2.2 "Normal" sub-mode rendering

As an input, the renderer obtains (or receives) a spatial audio signal containing one or two transport audio signals and associated spatial metadata. The number of transport audio signals (i.e., one or two) is determined by decoding it from the received IVAS bitstream (see clause 6.1), and the renderer receives this information (i.e., the number of transport audio signals). The spatial metadata obtained (or received) by the renderer contains the following parameters in frequency bands b: direction (as azimuth $\theta(b, m, i)$ and elevation $\phi(b, m, i)$ angles), direct-to-total energy ratio $r_{dir}(b, m, i)$, spread coherence $\zeta(b, m, i)$, and surround coherence $\gamma(b, m)$. In addition, lower frequency effect (LFE) information called the LFE-to-total energy ratio parameter is received.

First, direct and diffuse power factors and surround coherence ratios are computed using the methods described in clause 6.5.7.2.2, based on the direct-to-total energy ratio $r_{dir}(k, m, i)$ and the surround coherence $\gamma(k, m)$.

Then, directional responses are computed using the methods described in clause 6.5.7.2.3 based on the MASA spatial metadata.

Then, diffuse responses are computed using the methods described in clause 6.5.7.2.4.

Then, the transport audio signals are transformed to the time-frequency domain with a 60-bin (with the sampling rate of 48 kHz) complex low-delay filter-bank (CLDFB) (see clause 6.2.5 for details), resulting in $S(k, n, i)$, where k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index.

Then, the LFE signal $S_{LFE}(k, n)$ is rendered using the methods described in clause 6.7.2.4.2.4. As a part of the rendering, the transport audio signals $S(k, n, i)$ are modified to mitigate the LFE signal in them.

Then, prototype audio signals (direct and diffuse prototype audio signals) are determined based on the (modified) transport audio signals using the methods presented in clause 6.5.7.2.5.

Then, the determined diffuse prototype audio signals are decorrelated using the methods presented in clause 6.5.7.2.7.

Then, the spatial audio signals are synthesized using the methods presented in clause 6.5.7.2.8, yielding $S_{out}(k, n, j)$. The LFE signal $S_{LFE}(k, n)$ is combined with the synthesized spatial audio signals $S_{out}(k, n, j)$, resulting in the output multi-channel loudspeaker signals.

Finally, the time-frequency domain signals are converted to the time domain via the inverse CLDFB (see clause 6.2.5 for details), yielding $s_{out}(n, j)$, which are the synthesized time domain multi-channel loudspeaker signals.

6.7.2.4.2.3 "Separate-channel" sub-mode rendering

As an input, the renderer obtains (or receives) a spatial audio signal containing two transport audio signals $s_{trans}(n, i)$ and associated spatial metadata. The spatial metadata obtained (or received) by the renderer contains the following parameters in frequency bands b: direction (as azimuth $\theta(b, m, i)$ and elevation $\phi(b, m, i)$ angles), direct-to-total energy ratio $r_{dir}(b, m, i)$, spread coherence $\zeta(b, m, i)$, and surround coherence $\gamma(b, m)$. In addition, the renderer obtains (or receives) a separated channel signal $s_{sep}(n)$. In addition, lower frequency effect (LFE) information called the LFE-to-total energy ratio parameter is received.

First, a set of audio signals is synthesized using the decoded transport audio signals and the decoded spatial metadata. This synthesis (or in other words, rendering) is performed using a loudspeaker setup from which the channel associated with the separated channel signal has been omitted.

In the rendering of the set of audio signals, direct and diffuse power factors and surround coherence ratios are first computed using the methods described in clause 6.5.7.2.2, based on the direct-to-total energy ratio $r_{dir}(k, m, i)$ and the surround coherence $\gamma(k, m)$.

Then, directional responses are computed using the methods described in clause 6.5.7.2.3 based on the MASA spatial metadata.

Then, diffuse responses are computed using the methods described in clause 6.5.7.2.4.

Then, the transport audio signals are transformed to the time-frequency domain with a 60-bin (with the sampling rate of 48 kHz) complex low-delay filter-bank (CLDFB) (see clause 6.2.5 for details), resulting in $S(k, n, i)$, where k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index.

Then, prototype audio signals (direct and diffuse prototype audio signals) are determined based on the (modified) transport audio signals) using the methods presented in clause 6.5.7.2.5.

Then, the determined diffuse prototype audio signals are decorrelated using the methods presented in clause 6.5.7.2.7.

Then, the spatial audio signals are synthesized using the methods presented in clause 6.5.7.2.8, yielding $S_{out}(k, n, j)$.

Finally, the time-frequency domain signals are converted to the time domain via the inverse CLDFB (see clause 6.2.5 for details), yielding $s_{synth}(n, j)$, which are the synthesized time domain set of audio signals. These signals do not contain the channel associated with the separated channel signal nor the LFE channel.

Then, the LFE signal $s_{LFE}(n)$ is rendered using the methods described in clause 6.7.2.4.2.5. As a part of the rendering, the separated channel signal $s_{sep}(n)$ is modified to mitigate the LFE signal in it.

Finally, the multi-channel indices of the separated channel signal $s_{sep}(n)$ and the set of audio signals $s_{synth}(n, j)$ are identified. In addition, the multi-channel index of the LFE signal $s_{LFE}(n)$ is identified. Using the indices, the decoded separated channel audio signal $s_{sep}(n)$, the set of audio signals $s_{synth}(n, j)$, and the LFE signal $s_{LFE}(n)$ are combined to provide the multi-channel loudspeaker audio signals $s_{out}(n, j)$, which is the output of the processing.

6.7.2.4.2.4 Determination of the LFE signal using CLDFB

In the following, it is described how the low frequency effect (LFE) signal is rendered based on the transport audio signal(s) $S_i(k, n)$ and LFE information called the LFE-to-total energy ratio $\Xi(m)$, where k is the frequency bin, n the slot index, and m the subframe index. The audio signal is in the CLDFB domain.

First, prototype LFE signal is determined for the first frequency bin using the transport audio signals

$$S_{proto}(0, n) = \sum_{i=0}^{N_{trans}-1} S_i(0, n)$$

where N_{trans} is the number of transport audio signals.

The proto and the transport audio signal low-frequency energies are determined by

$$E_{proto}(n) = \left|S_{proto}(0, n)\right|^2$$

$$E_{trans}(n) = \sum_{i=0}^{N_{trans}-1} |S_i(0, n)|^2$$

Then, the target LFE and transport energies are determined by

$$E_{target,LFE}(n) = \Xi(m)\, E_{trans}(n)$$

$$E_{target,trans}(n) = (1 - \Xi(m))\, E_{trans}(n)$$

where $\Xi(m)$ is the LFE-to-total energy ratio (for the first frequency bin/band) and m is the subframe corresponding to the slot n.

The determined energies are smoothed over time by

$$E_{x,sm}(n) = E_x(n) + 0.95 E_{x,sm}(n - 1)$$

Then, the LFE gain is determined by

$$g_{LFE}(n) = \min\left(1, \sqrt{\frac{E_{target,LFE,sm}(n)}{E_{proto,sm}(n)}}\right)$$

and the LFE signal is generated by

$$S_{LFE}(0,n) = g_{LFE}(n) S_{proto}(0,n)$$

The LFE processing is performed on the first frequency bin only, and the rest of the bins are set to zero

$$S_{LFE}(k,n) = 0, \quad if\ k > 0$$

Then, the transport audio signal gain is determined by

$$g_{trans}(n) = \min\left(1, \sqrt{\frac{E_{target,trans,sm}(n)}{E_{trans,sm}(n)}}\right)$$

and the transport audio signals are processed to mitigate the LFE signal in them by

$$S_i(0,n) := g_{trans}(n) S_i(0,n)$$

The processing performed on the first frequency bin only, and the rest of the bins are not modified

$$S_i(k,n) = S_i(k,n), \quad if\ k > 0$$

The modified transport audio signals $S_i(k,n)$ are used in the subsequent processing.

6.7.2.4.2.5 Determination of the LFE signal using filters

In the following, it is described how the low frequency effect (LFE) signal is rendered based on the separate-channel signal $s_{sep}(n)$ and LFE information called the LFE-to-total energy ratio $\Xi(m)$. The audio signal is in the time domain.

The input to the processing is the separate-channel signal $s_{sep}(n)$. It is first delayed by 1.25 milliseconds. Then, the delayed version $s_{del}(n)$ is low-pass and high-pass filtered

$$s_{lp}(n) = s_{lp}(n-1) + c_{lp} s_{del}(n) - c_{lp} s_{del}(n - L_{lp})$$

$$s_{hp}(n) = s_{del}(n - L_{hp}) - s_{lp}(n)$$

where $s_{lp}(n)$ is the low-pass filtered part of the audio signal, $s_{hp}(n)$ the high-pass filtered part, L_{lp} corresponds to 5 milliseconds (e.g., 240 samples at 48 kHz), L_{hp} corresponds to 2.5 milliseconds (e.g., 120 samples at 48 kHz), and $c_{lp} = 1/L_{lp}$. The cross-over frequency of these filters is at 120 Hz.

Then, the LFE signal is synthesized in 1.25-millisecond slots (having L_{slot} samples, e.g., 60 samples at 48 kHz). The slot index is referred to by μ.

First, the transport audio signal energy is computed by

$$E_{trans}(\mu) = \sum_{n=n_1(\mu)}^{n_2(\mu)} s_{lp}(n)^2$$

where n_1 is the first and n_2 the last sample of the slot μ.

Then, the target LFE and transport energies are determined by

$$E_{target,LFE}(\mu) = \Xi(m)\, E_{trans}(\mu)$$

$$E_{target,trans}(\mu) = (1 - \Xi(m))\, E_{trans}(\mu)$$

where $\Xi(m)$ is the LFE-to-total energy ratio and m is the subframe corresponding to the slot μ.

The determined energies are smoothed over time by

$$E_{x,sm}(\mu) = E_x(\mu) + 0.95E_{x,sm}(\mu - 1)$$

Then, the LFE gain is determined by

$$g_{LFE}(\mu) = \min\left(1, \sqrt{\frac{E_{target,LFE,sm}(\mu)}{E_{trans,sm}(\mu)}}\right)$$

Furthermore, an interpolator is determined by

$$g_{interp}(n) = \frac{n}{L_{slot}}$$

The LFE signal is generated by

$$s_{LFE}(n + n_1(\mu)) = (g_{LFE}(\mu)g_{interp}(n) + g_{LFE}(\mu - 1)\left(1 - g_{interp}(n)\right))\, s_{lp}(n + n_1(\mu))$$

It is further low-pass filtered to mitigate further any high-frequency content in it by

$$s'_{LFE}(n) = s'_{LFE}(n-1) + c_{lp2}s_{LFE}(n) - c_{lp2}s_{LFE}(n - L_{lp2})$$

where L_{lp2} corresponds to 2.5 milliseconds (e.g., 120 samples at 48 kHz), and $c_{lp2} = \frac{1}{L_{lp2}}$, yielding the generated output LFE signal.

Then, the transport audio signal gain is determined by

$$g_{trans}(\mu) = \min\left(1, \sqrt{\frac{E_{target,trans,sm}(\mu)}{E_{trans,sm}(\mu)}}\right)$$

and the low-pass signal is processed using it and combined with the high-pass signal to generate a modified separate-channel signal $s'_{sep}(n)$ where the LFE signal has been mitigated

$$s'_{sep}(n + n_1(\mu)) = \left(g_{trans}(\mu)g_{interp}(n) + g_{trans}(\mu - 1)\left(1 - g_{interp}(n)\right)\right)s_{lp}(n + n_1(\mu)) + s_{hp}(n + n_1(\mu))$$

Then, the modified the separate-channel signal $s'_{sep}(n)$ is delayed by 1.25 milliseconds to sync it with the generated LFE signal.

6.7.2.4.2.6 Speaker-layout-based optimisation of loudspeaker rendering

The loudspeaker rendering of McMASA described in clause 6.7.2.4.2 is configured with two- or three-dimensional set of multiple speaker nodes to synthesize the output audio signals. With McMASA, the original input audio signal to McMASA analysis has a known defined speaker layout (representing the input multi-channel format configuration) in which the received and decoded downmixed transport audio signals are based in. This defined speaker layout is transmitted as a signalled parameter within the same bitstream with the transport audio signals and the related MASA spatial metadata. This signalling is a required part of the MC format signalling. The defined speaker layout and the output speaker layout are given to the loudspeaker renderer to configure the renderer.

As presented in clause 6.7.2.4.2, the synthesis of the output audio signals in the McMASA loudspeaker rendering is based on dividing the received transport audio signals into direct and diffuse parts based on the received MASA format spatial metadata and synthesizing them to direct and diffuse audio signals respectively. These direct and diffuse audio signals are combined to produce the output audio signals for loudspeaker reproduction.

The loudspeaker rendering of McMASA is optimized in two ways with the knowledge of the received defined speaker layout and the targeted output speaker layout to achieve optimized quality and complexity reduction.

The first approach is specific for when the defined speaker layout of the original multichannel signal is 5.1. In this case, the defined speaker layout is a direct subset of output speaker layouts 7.1, 5.1+2, and 5.1+4. Thus, the loudspeaker renderer is configured to use the defined speaker layout as the targeted speaker layout for output audio signal synthesis instead of the output speaker layout. This causes both the direct and the diffuse audio signals to be synthesized only for the 5.1 channels. The remaining unused output channels in the output speaker layout that are not present in the defined

speaker layout are put to zero already in time domain and no inverse CLDFB transforms are required. This saves complexity in rendering and also provides output synthesis which is closer to the original sound scene representation.

The second approach is used when the defined speaker layout of the original multichannel signal has loudspeakers only in the horizontal plane, that is, the defined speaker layout is either 5.1 or 7.1, and the output speaker layout contains enough loudspeakers which are elevated, i.e., the layout is three dimensional which corresponds to predefined layouts of 5.1+4 and 7.1+4, or a custom output speaker layout with more than two elevated loudspeakers. In this case, when there is such dimension mismatch between the defined speaker layout and the output speaker layout, the diffuse audio signal synthesis of the loudspeaker rendering is configured to use only the horizontal plane loudspeakers of the output speaker layout. This causes the diffuse audio signal path to synthesize a diffuse part perception that is closer to the diffuse part of the original sound scene. Thus, direct audio signals are synthesized using the output speaker layout and diffuse audio signals are synthesized with horizontal speakers of the output speaker layout due to the defined speaker layout being horizontal. The complexity of rendering is not reduced as it was in the first approach, but the output audio signal synthesis is improved. In practice, this supplements the diffuse response determination in clause 6.5.7.2.4. If the defined speaker layout is 5.1 and output speaker layout is 5.1+4, then the diffuse response is determined with the 5.1 entry from the table 6.5-1. In other cases of this approach, the diffuse response is determined with function

$$g_{diff}(k,m,j) = \frac{1}{\sqrt{N_{horiz}}}$$

where N_{horiz} is the number of close to horizontal plane loudspeakers.

6.7.2.4.3 Ambisonics rendering

The Ambisonic signals are rendered by rendering first 7.1.4 multi-channel loudspeaker signals using the methods described in clause 6.7.2.4.2 and then converting them to Ambisonic signals using an Ambisonic decoding matrix.

6.7.2.4.4 Stereo rendering

In of one transport audio signal, the stereo rendering is performed using the DFT-based stereo renderer described in clause 6.4.6.5.8.

In case of two transport audio signals, the transport audio signals are outputted as the stereo signals.

In case of two transport audio signals and the separated channel audio signal, the separated channel audio signal is first added to the transport audio signals after a multiplication with a gain of $\sqrt{0.5}$, and the resulting modified transport audio signals are outputted as the stereo signals.

6.7.2.4.5 Mono rendering

In of one transport audio signal, the transport audio signal is outputted as the mono signal.

In case of two transport audio signals, the transport audio signals are summed together after a multiplication with a gain of $\sqrt{0.5}$, and the resulting signal is outputted as the mono signal.

In case of two transport audio signals and the separated channel audio signal, the transport audio signals are first summed together after a multiplication with a gain of $\sqrt{0.5}$, and then the separated channel audio signal is added to the summed signal, and the resulting signal is outputted as the mono signal.

6.7.2.5 McMASA PLC

In case of a packet loss, the MASA metadata and the LFE-to-total energy ratio data from the previous frame are used. The packet loss concealment for the transport signal(s) follows the corresponding mono (SCE) or stereo (CPE) PLC operations.

6.7.2.6 McMASA decoding with TSM

McMASA decoding with time scale modification follows procedures explained in clause 6.2.7 (and specifically in subclause 6.2.7.4.2.2 for the metadata mapping) in addition to procedures in clause 6.7.2.

6.7.3 Parametric MC decoding mode

6.7.3.1 ParamMC Overview

The ParamMC decoder and synthesizer generates the multi-channel synthesized signal from the downmix signal using the received transport signal $s_{dmx}(n)$ (i.e. the downmix signal) with 2 or 3 channels and the received side information containing the channel level and correlation information of the original signal. The synthesis reconstructs a target version of the covariance information of the original signal based on an estimated version of the original covariance signal reported to the number of synthesis channels. The estimated version of the original covariance is acquired from the from the covariance information of the downmix signal by applying an estimating rule which is also a prototype rule for calculating prototype signals. Figure 6.7-1 shows the processing and signal flow in the ParamMC decoder.

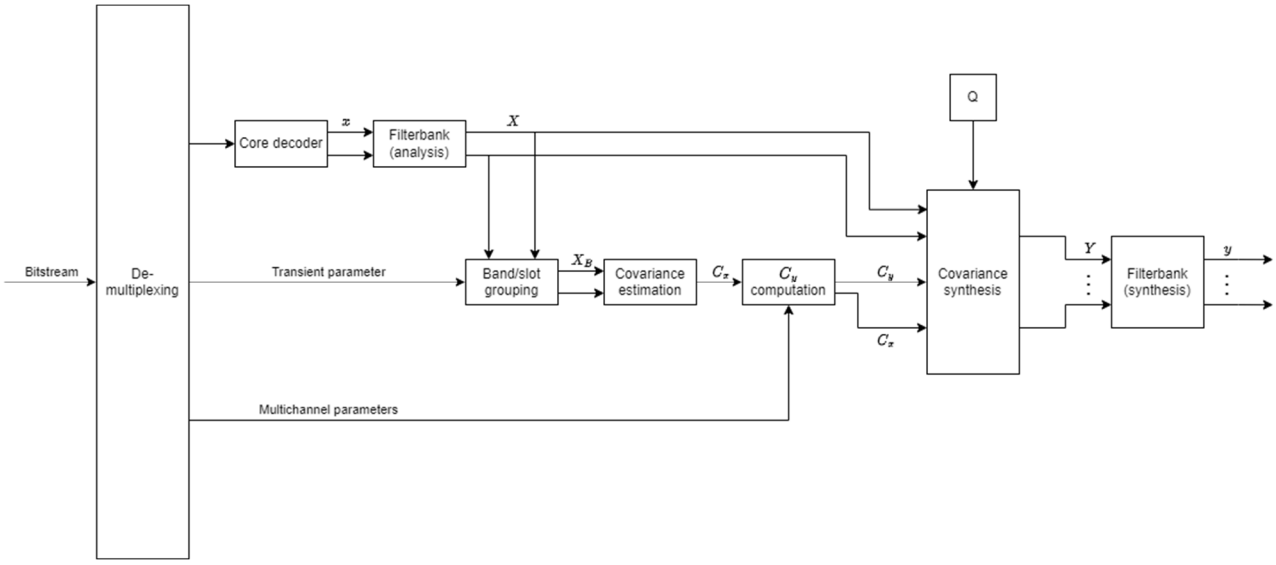


Figure 6.7-1: ParamMC Decoder

6.7.3.2 ParamMC configuration

Depending on the bit rate, the transport channel configuration the number of transport audio signals n_t, the parameter band grouping, and the parameter band mapping used to determine which parameter bands are sent in a certain frame based on i_F are chosen according to the Tables 5.7-8, 5.7-9, 5.7-10, 5.7-11, 5.7-12, 5.7-13, 5.7-14, and 5.7-16.

The maximum frequency for decorrelation is set to $f_{decorr,max} = 8000$ or $n_{b,decorr} = 20$ CLDFB bands, and the highest parameter band with residual mixing $n_{B,r}$ is the number of parameter bands n_b for WB from Table 5.7-15.

6.7.3.3 ParamMC Parameter Decoding

6.7.3.3.1 Common ParamMC parameter decoding

The common ParamMC parameters are decoding from the metadata as follows

- f_{LFE} with 1 bit
- the encoded band width with 2 bits
- the parameter frame indicator i_F with 1 bit
- the transient flag f_T with 1 bit, signalling the occurrence of a transient in the current frame
  - if the transient flag f_T is 1, the transient position i_T with 3 bits as integer, signalling in which segment the transient has occured

Depending on the signalled encoded band width and the chosen parameter band grouping the number of coded parameter bands n_B is taken from table 5.7-15 . If the transient flag f_t is set the parameter band step is set to $s_p = 2$, otherwise it is set to $s_p = 1$. If the transient flag is not set, the transient position is set to zero $i_T = 0$.

Depending on the encoded band width and the output sampling frequency the number of parameter bands used in the synthesis $n_{B,S}$ as the maximum of n_B and the number of parameter bands taken from table 5.7-15 using the output band width.

6.7.3.3.2 ICC and ICLD Parameter decoding

6.7.3.3.2.1 General parameter indices decoding

6.7.3.3.2.1.1 Overview

The quantized parameter indices decoded for all parameter bands belonging to the parameters bands to be decoded determined either by the coding band mapping and the current parameter frame indicator i_F or all bands if it is a transient frame, the number of bands n_{bands} having been encoded being (n_{b,I_F} see Table 5.7-24):

$$n_{bands} = \begin{cases} n_{B,I_F} & if \quad f_t = 0 \\ \frac{n_B}{s_p} + (n_B \bmod s_p) & else \end{cases} \quad (6.7\text{-}18)$$

Also the number of bands $n_{bands,LFE}$ with LFE parameters in the current frame is determined, $n_{bands,LFE} = 1$ if $f_{LFE} = 1$ and either the transient flag f_t=1 or band zero is in the transmitted band, i.e. $i_F(0)$ from the chosen parameter band mapping table is equal to the sent i_F and $n_{bands,LFE} = 0$ otherwise.

The total number of parameter indices encoded is

$$l_s = n_{bands} n_{NOLFE} + n_{bands,LFE}(n_{LFE} - n_{NOLFE}) \quad (6.7\text{-}19)$$

where n_{NOLFE} is the number of parameters in a band without LFE and n_{LFE} is the number of parameters with with LFE.

First the flag f_r indicating range coded parameters or uniformly coded parameters is read with 1 bit. Depending on this flag either range decoding or uniform decoding of the parameter indices is done.

6.7.3.3.2.1.2 ParamMC Uniform parameter decoding ($f_r = 0$)

The l_s quantized parameter indices are directly read as integers from the bitstream with a resolution of $n_{uniform}$ bits per index.

6.7.3.3.2.1.3 ParamMC parameter range decoding ($f_r = 1$)

If range decoding is chosen another flag is read with 1 bit from the bitstream indicating delta or absolute coding of the quantization indices. If the flag is one, l_s indices are decoded using the cumulative frequency tables cum_freq[] and symbol frequency tables sym_freq[] for delta coding and the range decoder. The delta coded indices are converted to absolute indices as described in the following pseudo code:

```
idx_prev = l_Q / 2 + l_Q % 2 - 1;
idx_offset = l_Q - 1;
for ( j = 0; j < l_s; j++ )
{
    idx[j] = idx_prev + delta_idx[j] - idx_offset;
    idx_prev = idx[j];
}
```

If the flag is zero, the quantized parameter indices are decoded with the range decoder cumulative frequency tables cum_freq[] and symbol frequency tables sym_freq[] for absolute coding, and the alphabet size defined as:

$$\text{sz\_alphabet} = \begin{cases} 2l_Q - 1, if\ delta\ coding \\ l_Q, else \end{cases} \quad (6.7\text{-}20)$$

The range decoder is based on the functions defined in clause 6.2.2.3.3 and is done according to the following pseudo code:

```
ivas_param_mc_range_decoder( )
{
    rc_uni_dec_init()
    for ( i=0; i < l_s ; i++ )
    {
        idx[i]=rc_uni_dec_read_symbol_fast(cum_freq,sym_freq,16,sz_alphabet);
        cur_bit_pos = rc_uni_dec_virtual_finish()
        if ( cur_bit_pos > max_allowable_bit_count )
        {
            bit_error_detected = 1;
            return 0;
        }
    }
}
```

6.7.3.3.2.1.4 Dequantization and rearranging

The sequence of quantization indices is used to get the sequence of quantized parameters $\hat{p}$ via the quantizer table t_k:

$$p(i) = t_k\big(idx(i)\big), i = 0, \dots, l_s - 1 \quad (6.7\text{-}21)$$

The sequence of decoded parameters is re-arranged into the correct position in the parameter band wise parameter buffer according to the following pseudo code:

```
for ( j = 0; j < n_NOLFE; ++j )
{
    coding_band = 0;
    k = 0;

    for ( i = 0; i < n_b; i += s_p )
    {
        if ( f_t == 1 || i_F == coding_band_mapping[i] )
        {
            i_q[i][j]=idx[k];
            if ( f_t == 1 && (i + 1) < n_b )
            {
                p_hat_out[i+1][j]=p_hat[k];
            }
            k++;
        }
    }
}
if ( n_bands_LFE )
{
    for ( j = 0 ; j < (n_LFE – n_NOLFE); j++)
    {
       p_hat_out [0][j + n_NOLFE] = p_hat[k];
       k++;
    }
}
else if ( i_F == coding_band_mapping[0] )
{
    for ( j = 0 ; j < (n_LFE – n_NOLFE); j++)
    {
       p_hat_out [0][j + n_NOLFE] = 0;
    }
}
```

6.7.3.3.2.2 ICLD parameter indices decoding

The quantized ICLD values are read from the bitstream and decoded and written to the parameter band wise ICLD buffer $\hat{\boldsymbol{\chi}}$ using the range coder code books specified in Tables 5.7-25 and 5.7-26, and a uniform bit demand $n_{uniform} = 4$, $l_Q = l_{Q,ICLD}$, $n_{NOLFE} = n_{ICLD,NOLFE}$, $n_{LFE} = n_{ICLD,LFE}$, and the ICLD quantization table 5.7-21 $t_k = \hat{\chi}_k, \hat{p}_{out} = \hat{\chi}$

6.7.3.3.2.3 ICC parameter indices decoding

The quantized ICC values are read from the bitstream and decoded and written to the parameter band wise ICC value buffer $\hat{\boldsymbol{\xi}}$ using the range coder code books specified in Tables 5.7-27 and 5.7-28, and a uniform bit demand $n_{uniform} = 3$, $l_Q = l_{Q,ICC}$, $n_{NOLFE} = n_{ICC,NOLFE}$, $n_{LFE} = n_{ICC,LFE}$, and the ICC quantization table 5.7-23 $t_k = \hat{\boldsymbol{\xi}}_k, \hat{p}_{out} = \hat{\boldsymbol{\xi}}$.

6.7.3.4 ParamMC Transport Audio Signal Decoding

The downmix audio signals s_{dmx} are decoded using either the Stereo MDCT decoder (for 2 transport channels) or the MCT decoder (for 3 transport channels). The decoded downmix audio signal are high pass filtered according to 6.2.1.1, resulting in the high pass filtered downmix signals s_{dmx}^{hp20}.

6.7.3.5 ParamMC Synthesis

6.7.3.5.1 ParamMC Synthesis Overview

The ParamMC synthesis generates the synthesis signal from the downmix signal, i. e. the transport time signals, the downmix signal having 2 or 3 channels. The synthesiser hast two paths, the first path is for synthesising the direct signal and using a mixing matrix that is calculated from the (target) covariance matrix of the synthesis signal and the covariance of the downmix signal. The second path is for synthesising a residual component using a prototype signal block configured for upmixing the the downmix signal from the number of downmix channels to the number of synthesis channels, a decorrelator for decorrelating the upmixed prototype signal and a mixing matrix block for synthesizing the residual signal using a (second) residual mixing matrix. The residual mixing matrix is obtained from the residual covariance provided by the first mixing matrix block and an estimate of the covariance matrix of the decorrelated prototype signals acquired from the covariance matrix of the downmix signal. The synthesizer adds up the direct (first) component of the synthesis signal with the residual (second) component of the synthesis signal. Figure 6.7-2 shows the processing blocks and signal flow of the ParamMC synthesis.

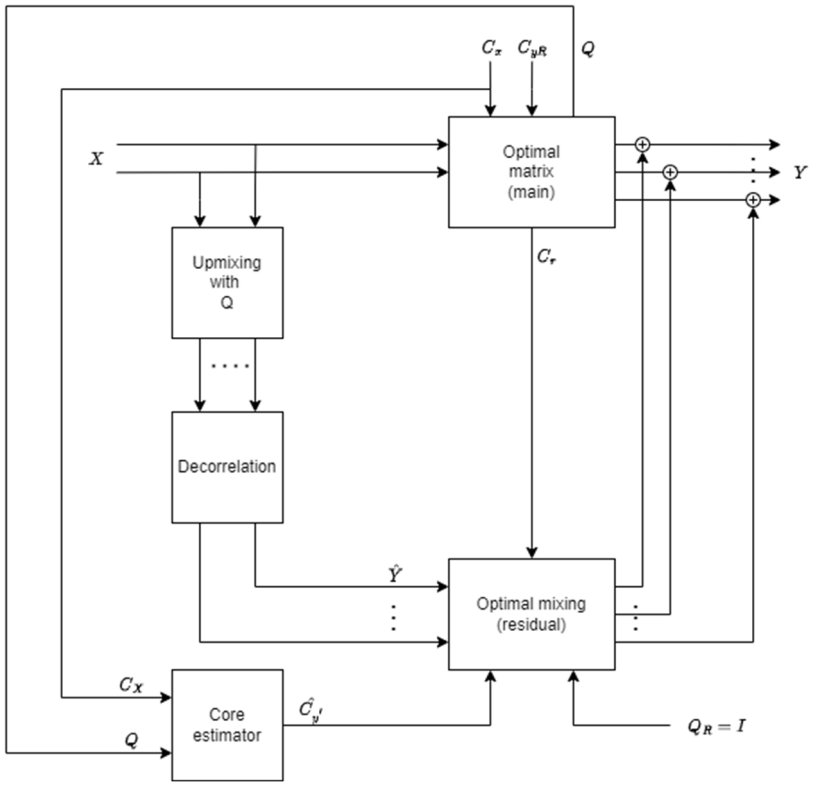


Figure 6.7-2: ParamMC Synthesis

6.7.3.5.2 ParamMC Target Covariance Calculation

The high pass filtered downmix signals s_{dmx}^{hp20} are transformed into the CLDFB domain according to clause 6.2.5.1, resulting in the CLDFB domain downmix signals S_{dmx}^{CLDFB}. For all CLDFB bands and all time slots $ts \geq 2i_T$ the covariance estimate of the down mix for this band and time slot is computed:

$$\mathbf{C}_{x,ts,b} = X_{ts}(b)X_{ts}(b)^* \quad (6.7\text{-}22)$$

where $X_{ts}(b) = \left[S^{CLDFB}_{dmx}{}_{0,ts}(b) \dots S^{CLDFB}_{dmx}{}_{n_t-1,ts}(b) \right]^T$ is the column vector of the CLDFB samples of all downmix channels of one band and one CLDFB time slot.

The covariance matrix estimate is accumulated per parameter band b_p for either all CLDFB timeslots when no transient was detected or for the CLDFB timeslots after the transient in case a transient was detected, and all CLDFB bins belonging to a specific parameter band b_p to create the sum of the covariances

$$\mathbf{C}_{x,b_p} = \sum_{b=b_{start}}^{b_{end}} \sum_{s=ts_{start}}^{ts_{end}} \mathbf{C}_{x,ts,b} \tag{6.7-23}$$

where: $ts_{start} = 2i_T$ is the first CLDFB time slot to be processed

ts_{end} is the last CLDFB time slot to be processed

$b_p = 0, \dots, n_{B,S}$

A parameter band index b_{p,max_abs_cov} is determined, where b_{p,max_abs_cov} is the highest synthesis parameter band where $b_{start}(b_p) < 20$. All entries in the (complex valued) covariance estimates for parameter bands below this index are converted to real values using the absolute value of the estimates, all entries for the parameter bands covariance values being equal or above this index are converted to real values using the real part of the complex estimates:

$$\mathbf{C}_{x,b_p} = \begin{cases} \left|\mathbf{C}_{x,b_p}\right| & if \quad b_p < b_{p,max\_abs\_cov} \\ \Re\mathrm{e}(\mathbf{C}_{x,b_p}) & else \end{cases} \tag{6.7-24}$$

where b_p is the parameter band, with b_p=0,s_p, 2s_p,…$n_{B,S} - 1$.

In case of a set transient flag, the estimated covariances are of two bands each are combined to form a new set of estimates for a new set of parameter bands with a reduced number of parameter and with less frequency resolution.

$$\mathbf{C}_{x,b_p} = \begin{cases} \mathbf{C}_{x,b_p} + \mathbf{C}_{x,b_p+1} & if \quad b_p + 1 < n_B \\ \mathbf{C}_{x,b_p} & else \end{cases} \tag{6.7-25}$$

The combined covariances are shared with all band belonging to the same combination:

$$\mathbf{C}_{x,b_{p+1}} = \mathbf{C}_{x,b_p} \quad if \quad b_p + 1 < n_B \tag{6.7-26}$$

where b_p is the parameter band, with b_p=0,s_p, 2s_p,…$n_{B,S} - 1$.

Based on this accumulated covariance estimate the target covariance for the chosen target loudspeaker format is calculated. First the covariance estimate of the prototype signals of the transported MC layout is calculated using the prototype matrix, which is identical to the downmix matrix M_{DMX} chosen according to the number of transport channels and the transported MC layout in the encoder (see Table 5.7-17).

$$\mathbf{C}_{proto,bp} = \mathbf{M}_{DMX} \mathbf{C}_{x,b_p} \mathbf{M}^{\mathrm{T}}_{DMX} \tag{6.7-27}$$

To avoid problems with numerical inaccuracies all entries in the main diagonal of $\mathbf{C}_{proto}$ that are negative are set to zero:

$$c_{proto,bp}{}_{jj} = \max\left(0, \ c_{proto,bp}{}_{jj}\right), \ j = 0, \dots, n_{ls,t} - 1 \tag{6.7-28}$$

where$n_{ls,t}$ is the number of channels in the transported MC layout.

The estimated ICCs are computed:

$$\xi_{proto}{}_{ij} = \frac{c_{proto}{}_{ij}}{\sqrt{c_{proto}{}_{ii} c_{proto}{}_{jj}}} \tag{6.7-29}$$

To get the target ICCs subset of the estimated ICCs are replaced by the transmitted ones:

$$\xi_{proto}{}_{ij} = \xi_{proto}{}_{ji} = \hat{\xi}_k, i = m_{icc}(k,0), j = m_{icc}(k,1) \tag{6.7-30}$$

where: $k = 0,1, \dots, n_{ICC,LFE}$-1 is the index of the ICC within the parameter band b_p

m_{ICC} is the ICC mapping according to Table 5.7-22

The target energies of the channels are derived using the transmitted ICLD values, the downmix covariance estimate the ICLD mapping according to Table 5.7-19 , and the ICLD normalization factors according to Table 5.7-20:

$$P_i = 10^{\frac{\hat{\chi}_k}{10}} P_{dmx,k} \tag{6.7-31}$$

$$P_{dmx,k} = f_{ICLD}(k) \sum_{l=0}^{l<n_T(k)} c_{x_{mm}}, m = m_{dmx,ild}(k,l) \tag{6.7-32}$$

where $i = m_{ild}(k)$ is the channel index of the channel in the transported MC layout and $k = 0, \ldots, n_{ILD} - 1$ is the mapped index.

The target covariance $\mathbf{C}_{y,trans_R}$ for the transported MC layout is now generated from the target ICCs together with the target energies:

$$c_{y,trans_{R_{ij}}} = \xi_{proto_{ij}} \sqrt{P_i P_j} \tag{6.7-33}$$

If the loudspeaker conversion in the covariance domain is chosen in the output processing selection the target covariance for the synthesis MC layout $\mathbf{C}_{y_R}$ is generated by applying the loudspeaker conversion matrix $\mathbf{M}_{conv}$ to the target covariance of the transported MC layout and an energy correction:

$$\mathbf{C}_{y,conv_R} = \mathbf{M}_{conv} \mathbf{C}_{y,trans_R} \mathbf{M}_{conv}^{\mathrm{H}} \tag{6.7-34}$$

$$c_{y,conv_{ii}} = \left| c_{y,conv_{ii}} \right| \tag{6.7-35}$$

$$P_{target,i} = \sum_{l=0}^{l<n_i} P_i m_{conv_{il}} \tag{6.7-36}$$

$$P_{conv,i} = c_{y,conv_{ii}} \tag{6.7-37}$$

$$a_{conv,i} = \sqrt{\frac{P_{target,i}}{P_{conv,i}}} \tag{6.7-38}$$

$$c_{y_{R_{ij}}} = c_{y,conv_{ii}} a_{conv,i} a_{conv,j} \tag{6.7-39}$$

Otherwise, the target covariance for the synthesis MC layout is the target covariance of the transported MC layout:

$$\mathbf{C}_{y_R} = \mathbf{C}_{y,trans_R} \tag{6.7-40}$$

Depending on the transient flag f_T the final target covariance and downmix covariance is either computed by smoothing (adding) the previous target covariance and downmix covariance or simply taking the current one.

$$\mathbf{C}_{y,t} = \begin{cases} \mathbf{C}_{y_R} + \mathbf{C}_{y_R}^{[-1]} & if \quad f_T = 0 \\ \mathbf{C}_{y_R} & else \end{cases} \tag{6.7-41}$$

$$\mathbf{C}_{x,t} = \begin{cases} \mathbf{C}_x + \mathbf{C}_x^{[-1]} & if \quad f_T = 0 \\ \mathbf{C}_x & else \end{cases} \tag{6.7-42}$$

6.7.3.5.3 ParamMC Mixing Matrix Calculation

6.7.3.5.3.1 Prototype Matrix and Target Covariance Preprocessing

To reduce the computational complexity of the mixing matrix calculation for all parameter bands that have no LFE signal, i.e. for $b_p = 0 \wedge f_{LFE} = 0$ and for $b_p > 0$, the entries belonging to the LFE channels of the synthesis MC layout are removed from the synthesis prototype matrix $\mathbf{Q}$ and the target covariance $\mathbf{C}_{y,t}$:

$$\mathbf{C}_{y,s} = [c_{y,t_{ij}}], \; i,j \notin i_{LFE,S} \tag{6.7-43}$$

$$\mathbf{Q}_s = [q_{ik}], \; i \notin i_{LFE,S} \tag{6.7-44}$$

$$n_s = n_{ls,S} - n_{LFE,S} \tag{6.7-45}$$

where $i, j = 0, \dots, n_s - 1$ is the index of the synthesis channel, $k = 0, \dots, n_t$ is the index of the downmix channel, $i_{LFE,S}$ is the set of the LFE indices in the synthesis MC layout, $n_{ls,S}$ is the total number of channels in the synthesis MC layout and $n_{LFE,S}$ is the number of LFE channels in the synthesis MC layout.

If the parameter band has an active LFE, i.e. for $b_p = 0 \wedge f_{LFE} = 1$:

$$\mathbf{C}_{y,s} = \mathbf{C}_{y,t} \tag{6.7-46}$$

$$\mathbf{Q}_s = \mathbf{Q} \tag{6.7-47}$$

$$n_s = n_{ls,S} \tag{6.7-48}$$

6.7.3.5.3.2 Direct Mixing Matrix

First, the diagonal of the covariance estimate of the prototype signals is computed:

$$C_{proto,diag,bp} = \mathrm{diag}(\mathbf{Q}_s \mathbf{C}_{x,b_p} \mathbf{Q}_s^{\mathrm{T}}) \tag{6.7-49}$$

Alle negative entries in the diagonal are set to zero:

$$C_{proto,diag,bp,i} = \max\left(0,\ C_{proto,diag,bp,i}\right),\ i = 0, \dots, n_S - 1 \tag{6.7-50}$$

The target covariance is decomposed using the singular value decomposition SVD:

$$\mathbf{U}_y \mathbf{S}_y \mathbf{V}_y^{\boldsymbol{H}} = \mathrm{SVD}(\mathbf{C}_{y,s}) \tag{6.7-51}$$

$$\mathbf{K}_y = \mathbf{U}_y \sqrt{\mathbf{S}_y} \tag{6.7-52}$$

Similarly, the downmix covariance is decomposed:

$$\mathbf{U}_x \mathbf{S}_x \mathbf{V}_x^{\boldsymbol{H}} = \mathrm{SVD}(\mathbf{C}_{x,bp}) \tag{6.7-53}$$

$$\mathbf{K}_x = \mathbf{U}_x \sqrt{\mathbf{S}_x} \tag{6.7-54}$$

The square roots of eigenvalues of the downmix covariance are regularized with a regularization factor $\alpha = 0.2$ and the regularized inverse of $\mathbf{K}_x$ is computed:

$$s_{x,max} = \max(\mathrm{diag}(\sqrt{\mathbf{S}_x})) \tag{6.7-55}$$

$$s'_{x_{ii}} = \max\left(\sqrt{s_{x_{ii}}}, \alpha s_{x,max} + \epsilon\right), i = 0, \dots, n_t\text{-}1 \tag{6.7-56}$$

$$\mathbf{K}_x^{-\mathbf{1}} = \mathbf{S}_x'^{-\mathbf{1}} \mathbf{U}_x^{\mathrm{T}} \tag{6.7-57}$$

A diagonal normalization matrix $\widehat{\mathbf{G}}_y$ is formulated using a normalization factor $\beta = 0.001$:

$$c_{proto,diag,max} = \max(C_{proto,diag,bp}) \tag{6.7-58}$$

$$\hat{C}_{proto,diag,i} = \max(C_{proto,diag,bp,i}, \beta c_{proto,diag,max} + \epsilon), i = 0, \dots, n_S\text{-}1 \tag{6.7-59}$$

$$\hat{g}_{y,jj} = \sqrt{\frac{c_{y,s_{jj}}}{\hat{c}_{proto,diag,j}}}, j = 0, \dots, n_S\text{-}1 \tag{6.7-60}$$

The optimal P for the direct mixing matrix is computed:

$$\mathbf{U}_\mathbf{P} \mathbf{S}_\mathbf{P} \mathbf{V}_\mathbf{P}^{\mathrm{T}} = \mathrm{SVD}(\mathbf{K}_x^{\mathrm{T}} \mathbf{Q}_s^{\mathrm{T}} \widehat{\mathbf{G}}_y^{\mathrm{T}} \mathbf{K}_y) \tag{6.7-61}$$

$$\mathbf{P} = \mathbf{V}_\mathbf{P} \boldsymbol{\lambda} \mathbf{U}_\mathbf{P}^{\mathrm{T}} \tag{6.7-62}$$

where $\boldsymbol{\lambda}$ is the row truncated identity matrix of size (n_S, n_t).

The optimal direct part mixing matrix is computed:

$$\mathbf{M}_{direct} = \mathbf{K}_y \mathbf{P} \mathbf{K}_x^{-\mathbf{1}} \tag{6.7-63}$$

The estimated direct covariance is computed:

$$\tilde{\mathbf{C}}_y = \mathbf{M}_{direct}\mathbf{C}_{x,bp}\mathbf{M}_{direct}^{\mathrm{T}} \tag{6.7-64}$$

For parameter bands with decorrelation the residual covariance matrix $\mathbf{C}_r$ is computed:

$$\mathbf{C}_r = \mathbf{C}_{y,s} - \tilde{\mathbf{C}}_y \tag{6.7-65}$$

and negative values in the main diagonal set to zero:

$$c_{r_{jj}} = \max\left(0,\ c_{r_{jj}}\right),\ j = 0,\ldots,n_S - 1 \tag{6.7-66}$$

For parameter bands without decorrelation an energy adjustment via a diagonal adjustment matrix $\mathbf{M}_{adj}$ of the direct mixing matrix is applied:

$$m_{adj_{ii}} = \begin{cases} 1 & if \quad \tilde{c}_{y_{ii}} < 0 \\ \max\left(4, \sqrt{\frac{c_{y,s_{ii}}}{\tilde{c}_{y_{ii}}+\epsilon}}\right) & else \end{cases},\ i = 0,\ldots,n_S - 1 \tag{6.7-67}$$

$$\mathbf{M}_{direct} = \mathbf{M}_{adj}\mathbf{M}_{direct} \tag{6.7-68}$$

If entries corresponding the LFE channels of the synthesis MC layout have been removed before the mixing matrix calculation rows identical to zero are inserted into $\mathbf{M}_{direct}$ at the row indices in the set of LFE indices $i_{LFE,S}$.

6.7.3.5.3.3 Residual Mixing Matrix

For parameter bands with decorrelation the residual mixing matrix based on the residual covariance $\mathbf{C}_r$ is computed.

The residual covariance $\mathbf{C}_r$ is decomposed using the singular value decomposition SVD:

$$\mathbf{U}_r\mathbf{S}_r\mathbf{V}_r^{\mathrm{T}} = \mathrm{SVD}(\mathbf{C}_r) \tag{6.7-69}$$

$$\mathbf{K}_r = \mathbf{U}_r\sqrt{\mathbf{S}_r} \tag{6.7-70}$$

The decomposition $\mathbf{K}_{proto}$ of the decorrelated prototype signals covariance is computed, $\mathbf{K}_{proto}$ is a diagonal matrix:

$$k_{proto_{ii}} = \sqrt{C_{proto,diag,bp,i}} \tag{6.7-71}$$

And the regularized inversive of $\mathbf{K}_{proto}$ is formulated using the regularization factor $\alpha = 0.2$:

$$k_{proto,max} = \max(\mathrm{diag}(\mathbf{K}_{proto})) \tag{6.7-72}$$

$$k^{-1}_{proto_{ii}} = \frac{1}{\max(k_{proto_{ii}}, \alpha k_{proto,max+\epsilon})} \tag{6.7-73}$$

A diagonal normalization matrix $\widehat{\mathbf{G}}_{proto}$ is formulated using a normalization factor $\beta = 0.001$:

$$c_{proto,diag,max} = \max(C_{proto,diag,bp}) \tag{6.7-74}$$

$$\hat{C}_{proto,diag,i} = \max(C_{proto,diag,bp,i}, \beta c_{proto,diag,max} + \epsilon),\ i = 0, \ldots, n_S\text{-}1 \tag{6.7-75}$$

$$\hat{g}_{proto_{jj}} = \sqrt{\frac{c_{y,s_{jj}}}{\hat{c}_{proto,diag,j}}},\ j = 0, \ldots, n_S\text{-}1 \tag{6.7-76}$$

The optimal $\mathbf{P}_r$ for the direct mixing matrix is computed:

$$\mathbf{U}_{\mathbf{P}_r}\mathbf{S}_{\mathbf{P}_r}\mathbf{V}_{\mathbf{P}_r}^{\mathrm{T}} = \mathrm{SVD}(\mathbf{K}_{proto}^{\mathrm{T}}\widehat{\mathbf{G}}_{proto}^{\mathrm{T}}\mathbf{K}_{proto}) \tag{6.7-77}$$

$$\mathbf{P}_r = \mathbf{V}_{\mathbf{P}_r}\mathbf{U}_{\mathbf{P}_r}^{\mathrm{T}} \tag{6.7-78}$$

The optimal residual part mixing matrix is computed:

$$\mathbf{M}_{residual} = \mathbf{K}_{proto}\mathbf{P}_\mathbf{r}\mathbf{K}^{-\mathbf{1}}_{proto} \tag{6.7-79}$$

The estimated direct covariance is computed:

$$\mathrm{diag}(\mathbf{C}_{proto}) = C_{proto,diag,bp_i},\ i = 0,\ldots,n_S - 1 \tag{6.7-80}$$

$$\tilde{\mathbf{C}}_r = \mathbf{M}_{residual}\mathbf{C}_{proto}\mathbf{M}_{residual}^{\mathrm{T}} \tag{6.7-81}$$

An energy adjustment via a diagonal adjustment matrix $\mathbf{M}_{adj,r}$ of the direct mixing matrix is applied:

$$m_{adj,r_{ii}} = \max(4, \sqrt{\frac{c_{r_{ii}}}{\tilde{c}_{r_{ii}}+\epsilon}}),\ i = 0,\ldots,n_S - 1 \tag{6.7-82}$$

$$\mathbf{M}_{residual} = \mathbf{M}_{adj,r}\mathbf{M}_{residual} \tag{6.7-83}$$

If entries corresponding the LFE channels of the synthesis MC layout have been removed before the mixing matrix calculation rows identical to zero are inserted into $\mathbf{M}_{residual}$ at the row indices in the set of LFE indices $i_{LFE,S}$.

6.7.3.5.4 Decorrelation

A decorrelated prototype signal is created for all CLDFB bands $n_{B,decorr}$ where decorrelation shall be applied. For this the CLDFB domain decorrelation according to clause 6.2.6 is applied. The decorrelator is set up with f_{onsets}=1 and with $n_{B,decorr,init} = 20$. The prototype signal computation creates prototypes according to a prototype signal information.

The prototype signal information is based on the prototype matrix $\mathbf{Q}$ and the initialization of the information starts with an empty matrix $\mathbf{Q}_{proto}$ of size (0,n_t) for the diffuse prototype and fills it and the prototype index set i_{proto} needed for the decorrelator according to the following algorithm:

$n_{proto} = 0$
for $i_c = 0, \ldots, n_s - 1$:
    $f = 0$
    for $i_p = 0, \ldots, n_{proto} - 1$:
        $d = \sum_{l=0}^{n_t-1} \left| q_{i_c l} - q_{proto_{i_p l}} \right|$
        if $d < 0.1$:
            $f = 1$
            $i_{proto}(i_c) = i_p$
            break
    if $f == 1$:
        $q_{proto_{n_{proto^*}}} = q_{i_{c^*}}$
        $i_{proto}(i_c) = n_{proto}$
        $n_{proto} = n_{proto}+1$

All entries in $\mathbf{Q}_{proto}$ close to zero are set to zero:

$$q_{proto_{ij}} = \begin{cases} 0 & if \quad q_{proto_{ij}} < \epsilon \\ q_{proto_{ij}} & else \end{cases};\ i = 0, \ldots, n_{proto} - 1;\ j = 0, \ldots, n_t - 1 \tag{6.7-84}$$

The prototype signals for the decorrelator for a CLDFB time slot ts and a CLDFB band b are computed as:

$$Y_{P,decorr,in,ts}(b) = \mathbf{Q}_{proto}X_{ts}(b) \tag{6.7-85}$$

The decorrelated prototype signals $Y_{P,decorr,ts}$ are obtained by applying the CLDFB domain decorrelator on $Y_{P,decorr,in,ts}$.

6.7.3.5.5 Upmix

The interpolator for the current frame is adapted, if $f_T = 1$ as a step function:

$$g(i) = \begin{cases} 0 & if \quad i < 2i_T \\ 1 & else \end{cases} \tag{6.7-86}$$

If $f_T = 0$ the interpolator is adapted according to clause 6.2.7.4.3.2 Eq. (6.2-102).

For each time slot ts the direct part is up-mixed from the CLFDB domain downmix signals for all synthesis parameter bands $n_{B,S}$:

$$Y_{direct,ts}(b) = ((1 - g(ts))\mathbf{M}^{[-1]}_{direct,b_p} + g(ts)\mathbf{M}_{direct,b_p})X_{ts}(b) \quad (6.7\text{-}87)$$

where $b_p = 0, \dots, n_{B,S} - 1$ are the parameter band indices for the synthesis and $b = b_{start}(b_p), \dots, b_{end}(b_p)$ are the CLDFB bands in a parameter band according to the chosen parameter band grouping.

For parameter bands with residual processing the residual part is up-mixed from the decorrelated prototype signals $Y_{P,decorr,ts}$:

$$Y_{residual,ts}(b) = ((1 - g(ts))\mathbf{M}^{[-1]}_{residual,b_p} + g(ts)\mathbf{M}_{residual,b_p})Y_{P,decorr,ts}(b) \quad (6.7\text{-}88)$$

where $b_p = 0, \dots, n_{B,r} - 1$ are the parameter band indices with residual for the synthesis and $b = b_{start}(b_p), \dots, b_{end}(b_p)$ are the CLDFB bands in a parameter band according to the chosen parameter band grouping.

The final synthesis signal for the synthesis MC layout is the sum of the both parts:

$$Y_{ts}(b) = \begin{cases} Y_{direct,ts}(b) + Y_{residual,ts}(b) & if \quad b_p < n_{B,r} \\ Y_{direct,ts}(b) & else \end{cases} \quad (6.7\text{-}89)$$

where $Y_{ts}(b) = \left[S^{CLDFB}_{synth\ 0,ts}(b) \dots S^{CLDFB}_{synth\ n_{ls,S}-1,ts}(b) \right]^T$ is the column vector of the CLDFB samples of all synthesis MC layout channels of one band and one CLDFB time slot, $b_p = 0, \dots, n_{B,r} - 1$ are the parameter band indices with residual for the synthesis, $b = b_{start}(b_p), \dots, b_{end}(b_p)$ are the CLDFB bands in a parameter band according to the chosen parameter and grouping.

If the output band width is higher than the encoded band width, all CLDFB bands in the for the synthesis signal above the encoded band width are set to zero for all time slots.

6.7.3.6 ParamMC output processing

6.7.3.6.1 ParamMC output processing selection

Depending on the transported format and the output format different output processing paths are chosen.

6.7.3.6.2 ParamMC mono/stereo output processing

The ParamMC synthesis is not applied, instead an energy compensation based on the transported MC layout and the encoded ICLDs is applied to the downmix signal in the MDCT domain prior to the inverse MDCT in the core-decoder, followed by the domain MC format conversion described in clause 6.7.7**.** The sfb band table in the MC format conversion is set to have the same parameter bands as for the ParamMC synthesis:

$$sfb(b) = \begin{cases} 0 & if \quad \text{b} = 0 \\ 16(b_{end}(b-1)+1) & else \end{cases} \quad (6.7\text{-}90)$$

where $b = 0, \dots, n_B$.

The conversion matrix for MC format conversion is changed to:

$$\acute{\mathbf{M}}_{conv} = \mathbf{M}_{conv}\mathbf{M}_{dmx} \quad (6.7\text{-}91)$$

$$\acute{\mathbf{M}}_{conv} = \frac{\acute{\mathbf{M}}_{conv}}{\max(1, \max(|\acute{\mathbf{M}}_{conv}|))} \quad (6.7\text{-}92)$$

The energy compensation is applied in the core-decoder on the fully reconstructed TCX spectra, in case of Stereo MDCT as core-coder before the minimum statistics is run , in case of MCT as core-coder before the final reconstruction with ITF and IMDCT.

First all downmix channels MDCT spectra are converted to the TCX20 resolution according to clause 6.3.6.2.3.5 and the MDST spectra are estimated as described in clause 6.3.6.2.3.2Eq. (6.3-191) From the MDCT and MDST spectra the MDCT bin covariances $\mathbf{C}_x$ are calculated:

$$\mathbf{C}_{x,b} = X(b)X(b)^{\mathrm{H}} \tag{6.7-93}$$

where $X(b) = \left[MDCT_{b,0} + iMDST_{b,0} \ldots MDCT_{b,n_t-1} + iMDST_{b,n_t-1}\right]^T$ is the column vector of the complex samples of all downmix channels of one MDCT/MDST bin.

The covariance matrix estimate is accumulated per parameter band b_p for either all CLDFB timeslots when no transient was detected or for the CLDFB timeslots after the transient in case a transient was detected, and all CLDFB bins belonging to a specific parameter band b_p to create the sum of the covariances

$$\mathbf{C}_{x,b_p} = \sum_{b=16b_{start}}^{16b_{end}+15} \mathbf{C}_{x,b} \tag{6.7-94}$$

The target channel energies P_i for the encoded MC layout are computed according to Eqs. (6.7-31) and (6.7-32).

The target energies for the output channels $P_{out,j}$ are the computed using the MC conversion matrix $\mathbf{M}_{conv}$:

$$P_{out,b_p} = \sum_{j=0}^{n_{out}} \sum_{i=0}^{n_t-1} m_{conv_{ji}} P_i \tag{6.7-95}$$

The energy of the downmix signal per MDCT bin is computed using the adapted MC layout conversion matrix $\acute{\mathbf{M}}_{conv}$:

$$E_{dmx} = \left|\acute{\mathbf{M}}_{conv} X(b)\right| \tag{6.7-96}$$

and the power of the downmix signal is summed up for a parameter band:

$$P_{dmx,b_p} = \sum_{j=0}^{n_{out}-1} \sum_{b=16b_{start}}^{16b_{end}+15} E_{dmx,j}^2 \tag{6.7-97}$$

The downmix and target energies are smoothed over time:

$$\bar{P}_{out,b_p} = (1 - 0.0435)\bar{P}_{out,b_p}^{[-1]} + 0.0435 P_{out,b_p} \tag{6.7-98}$$

$$\bar{P}_{dmx,b_p} = (1 - 0.0435)\bar{P}_{dmx,b_p}^{[-1]} + 0.0435 P_{dmx,b_p} \tag{6.7-99}$$

The original MDCT spectra are now energy adjusted according to clause 6.7.7.3 with the calculated downmix energies and target energies and the sfb table from Eq. (6.7-80) and the following gain

$$g_B = \max\left(\min\left(\sqrt{\frac{\bar{P}_{out,b_p}}{\bar{P}_{dmx,b_p}+\epsilon}}, 2\right), 0.3\right) \tag{6.7-100}$$

6.7.3.6.3 ParamMC loudspeaker output processing

6.7.3.6.3.1 Output to the encoded MC layout

The ParamMC synthesis is performed without the MC layout conversion in the covariance domain with $\mathbf{Q} = \mathbf{M}_{dmx}$. The resulting MC signal in the CLDFB domain S_{synth}^{CLDFB} is processed by the CLDFB synthesis filter bank to get the final time domain output signal s_{out}.

6.7.3.6.3.2 MC format conversion in the covariance domain

If the output MC layout is not identical to the encoded MC layout and the number of channels in the output MC layout is equal or less than the number of channels in the encoded MC layout, MC format conversion by conversion in the covariance domain is applied. For this the prototype matrix $\mathbf{Q}$ is calculated as follows:

$$\mathbf{Q} = \mathbf{M}_{conv}\mathbf{M}_{dmx}$$

where $\mathbf{M}_{conv}$ is MC layout conversion matrix according to clause 6.7.7.4.2 in case of one of the pre-defined loudspeaker layouts or 6.7.7.5 in case of a custom loudspeaker layout..

The ParamMC synthesis is performed with the MC layout conversion. The resulting MC signal in the CLDFB domain S_{synth}^{CLDFB} is processed by the CLDFB synthesis filter bank to get the final time domain output signal s_{out}.

6.7.3.6.3.3 MC format conversion in the CLDFB domain

If the output MC layout is not identical to the encoded MC layout and the number of channels n_{out} in the output MC layout is greater than the number of channels in the encoded MC layout, MC format conversion in the CLDFB domain is applied. The ParamMC synthesis is performed without the MC layout conversion in the covariance domain with $\mathbf{Q} = \mathbf{M}_{dmx}$. resulting MC signal in the CLDFB domain S_{synth}^{CLDFB} is then converted to the output MC layout in the CLDFB domain:

$$Y_{out,ts}(b) = \mathbf{M}_{conv} Y_{ts} \tag{6.7-101}$$

where $Y_{ts}(b) = \left[{S_{synth}^{CLDFB}}_{0,ts}(b) \dots {S_{synth}^{CLDFB}}_{n_{ls,S}-1,ts}(b) \right]^T$ is the column vector of the CLDFB samples of all synthesis MC layout channels of one band and one CLDFB time slot, $Y_{out,ts}(b) = \left[{S_{out}^{CLDFB}}_{0,ts}(b) \dots {S_{out}^{CLDFB}}_{n_{out}-1,ts}(b) \right]^T$ is the column vector of the CLDFB samples of all output MC layout channels of one band and one CLDFB time slot and $\mathbf{M}_{conv}$ is MC layout conversion matrix according to clause 6.7.7.4.2 in case of one of the pre-defined loudspeaker layouts or 6.7.7.5 in case of a custom loudspeaker layout.

The resulting CDFLB signal is energy compensated using the target and downmix energy of each band in a CLDFB time slot, each time slot is processed on its own:

$$E_{target}(b) = \sum_{i=0}^{n_{out}-1} \sum_{j}^{n_{synth}-1} \left| m_{ij} {S_{synth}^{CLDFB}}_{j,ts}(b) \right|^2 \tag{6.7-102}$$

$$E_{dmx}(b) = \sum_{i=0}^{n_{out}-1} \left| {S_{out}^{CLDFB}}_{i,ts}(b) \right|^2 \tag{6.7-103}$$

The downmix and target energies are smoothed over time slots:

$$\bar{E}_{target}(b) = (1 - 0.0435)\bar{E}_{target}^{[-1]} + 0.0435 E_{target} \tag{6.7-104}$$

$$\bar{E}_{dmx}(b) = (1 - 0.0435)\bar{E}_{dmx,j}^{[-1]} + 0.0435 E_{dmx} \tag{6.7-105}$$

The output MC layout channels for one time slot are the adjusted:

$$g(b) = \max\left(\min\left(\sqrt{\frac{\bar{E}_{target}(b)}{\bar{E}_{dmx}(b)+\epsilon}}, 2\right), 0.3\right) \tag{6.7-106}$$

$${S_{out}^{CLDFB}}_{i,ts}(b) = g(b) {S_{out}^{CLDFB}}_{i,ts}(b) \tag{6.7-107}$$

The resulting MC signal in the CLDFB domain S_{out}^{CLDFB} is processed by the CLDFB synthesis filter bank to get the final time domain output signal s_{out}.

6.7.3.6.4 ParamMC Ambisonics output processing

The ParamMC synthesis is performed without the MC layout conversion in the covariance domain with $\mathbf{Q} = \mathbf{M}_{dmx}$. The resulting MC signal in the CLDFB domain S_{synth}^{CLDFB} is processed by the CLDFB synthesis filter bank to get the synthesized time domain signal s_{synth}. The final time domain signal s_{out} is obtained by applying channel base audio to SBA rendering according to clause 7.5.5.2.

6.7.3.6.5 ParamMC binaural output processing

The ParamMC synthesis is performed without the MC layout conversion in the covariance domain with $\mathbf{Q} = \mathbf{M}_{dmx}$.

If the fast convolution binaural renderer without room simulation and with combined rotation is active, the encoded MC layout signal in the CLDFB domain S_{synth}^{CLDFB} is converted to 3rd order Ambisonics (using the real spherical harmonic response via table look up) to get the input signal for the fast convolution binaural renderer $S_{bin,in}^{CLDFB}$. The 3rd order Ambisonics signal is grouped into subframes and each subframe is rotated in SHD (clause 6.4.6.5.6.2) with the combined orientation belonging to said subframe and processed in the fast convolution binaural renderer (clause 7.2.2.4). The resulting binauralized signal in the CLDFB domain S_{out}^{CLDFB} is processed by the CLDFB synthesis filter bank to get the final time domain output signal s_{out}.

Otherwise, the LFE channels are added to the other channels of the encoded MC layout for the lowest CLDFB band and the LFE channels are removed from the input signal to the binaural renderer:

$$S_{out}^{CLDFB}{}_{i,ts}(0) = S_{out}^{CLDFB}{}_{i,ts}(0) + \frac{1.88364911}{n_S - n_{LFE}} \sum_{j=0}^{n_{LFE,S}-1} S_{out}^{CLDFB}{}_{i_{LFE,S}(j),ts}(0) \quad (6.7\text{-}108)$$

$$S_{bin,in,ts}^{CLDFB} = [\, S_{out}^{CLDFB}{}_{i,ts}], i \notin i_{LFE,S} \quad (6.7\text{-}109)$$

where n_S is the number of channels in the encoded MC layout, $n_{LFE,S}$ is the number of LFE channels in the encoded MC layout, $i_{LFE,S}$ is the set of channel indices of the LFE channels in the encoded MC layout and $i = 0, \dots, n_S - 1;\ i \notin i_{LFE,S}$.

The input signal to the binaural renderer signal is grouped into subframes and each subframe is processed in the fast convolution binaural renderer with the if combined orientation tracking is active, the combined orientation belonging to said subframe. The resulting binauralized signal in the CLDFB domain S_{out}^{CLDFB} is processed by the CLDFB synthesis filter bank to get the final time domain output signal s_{out}.

6.7.3.7 ParamMC PLC

If a packet loss happened, the ParamMC Parameter decoding is omitted and the transient flag is set to zero $f_t = 0$ and the transient position is set to zero $i_t = 0$.

6.7.3.8 ParamMC bit rate switching

If a bit rate switch happens and the last MC mode was also ParamMC, the transport channel configuration the number of transport audio signals n_t, the parameter band grouping, the downmix matrix $\mathbf{M}_{dmx}$ and the parameter band mapping used to determine which parameter bands are sent in a certain frame based on i_F are chosen according to the Tables 5.7-8, 5.7-9, 5.7-10, 5.7-11, 5.7-12, 5.7-13, 5.7-14, and 5.7-16 and the new bit rate.

If $n_t \neq n_t^{[-1]}$, the ICLD and ICC buffers are set to the default values ($\hat{\chi} = -92, \hat{\xi} = 0$). If the output format is mono or stereo, the approbriate MC layout conversion matrix $\mathbf{M}_{conv}$ is chosen and the adapted converision matrix $\acute{\mathbf{M}}_{conv}$ is re-calculated according to clause 6.7.3.6.2. For all other output formats the decorrelator is closed and a new decorrelator opened and the prototype signal information is re-initialized according to clause 6.7.3.5.4. If the output format is not the same as the transported MC layout, the the approbriate MC layout conversion matrix $\mathbf{M}_{conv}$ is chosen. The core-decoder is closed and a new one corresponding to the number of transport channels is opened.

If $n_t = n_t^{[-1]}$ and $n_B \neq n_B^{[-1]}$, a parameter band mapping is used to map ICLDs, ICCs, and the mixing matrices to the new parameter band mapping. The parameter band mapping is set up according to the following pseudo code:

```
for n_{B,target} = 0, ..., n_B − 1:
    l = b_{start}(n_{B,target})
    u = b_{end}(n_{B,target})
    c_b = 0
    c_B = 0
    for  n_{B,source} = 0, ..., n_B^{[-1]} − 1:
        if  b_{start}^{[-1]}(n_{B,source}) ≤ l ∧ b_{end}^{[-1]}(n_{B,source}) ≥ l:
            n_b = min(b_{end}^{[-1]}(n_{B,source}), u) − max(b_{start}^{[-1]}(n_{B,source}), l)
            if  n_b > 0:
                c_b = c_b + n_b
                i_{source,n_{B,target},c_B} = n_{B,source}
                fac_{source,n_{B,target},c_B} = n_b
                c_B = c_B + 1
    fac_{norm} = 1 / c_b
    for  i_S = 0, ..., c_B − 1:
        fac_{source,n_{B,target},c_B} = fac_{norm} fac_{source,n_{B,target},c_B}
    cnt_{n_{b,target}} = c_B
```

Generally, the mapped parameter is calculated as:

$$p_{n_B} = \sum_{k=0}^{cnt_{n_B}-1} fac_{source,n_B,k} p_{i_{source},n_B,k}^{[-1]} \quad (6.7\text{-}110)$$

where for the direct mixing matrices $p^{[-1]} = \mathbf{M}_{direct}^{[-1]}$, for the residual mixing matrices $p^{[-1]} = \mathbf{M}_{residual}^{[-1]}$, for the ICLDs $p^{[-1]} = \hat{\chi}^{[-1]}$, and for the ICCS $p^{[-1]} = \hat{\xi}^{[-1]}$.

6.7.3.9 ParamMC decoding with TSM

6.7.3.9.1 Decoding to mono/stereo

The mono/stereo processing according to 6.7.3.6.2 is already applied directly after the transport channel and metadata encoding and the TSM is applied to the mono/stereo signal. The transport channel buffer acts as a simple output buffer with 1 (mono) or 2 (stereo) channels.

6.7.3.9.2 Decoding to all other output formats

The TSM is applied to the transport channels, the transport channel buffers has to store the number of transport channels. The number of time slots in the frame to be rendered n_{TS} is determined as described in 6.2.7.2, the interpolator for the synthesis is determined as described in0 (6.2-102) with $L_{seg} = 8, L_f = n_{TS}$. The subframes are determined as described in 6.2.7.4.3.1. The transient position i_t is adapted:

$$i_{t,a} = \max(0, i_t + (L_{f,t} - 2L_{seg,t})) \quad (6.7\text{-}111)$$

where $L_{f,t} = \frac{L_f}{2}$ is the frame length to be rendered in terms of possible transient positions, $L_{seg,t} = \frac{L_{seg}}{2}$ is half the default frame length in terms of possible transient positions, $i_{t,a}$ is used in place of i_t wherever applicable in the ParamMC processing.

The ParamMC reconstruction and rendering is then applied according the processing needed for a specific output format.

In case of a bit rate switch where the previous frame was also a frame coded with ParamMC and if the number of transport channels changed, a transport channel buffer reconfiguration is done according to clause 6.2.7.5.2.

The decoder shall keep track of the already rendered samples, time slots, and sub frames.

6.7.4 Parametric upmix MC decoding mode

6.7.4.1 General

A schematic of the decoder of the parametric upmix coding mode is shown in the following Figure.

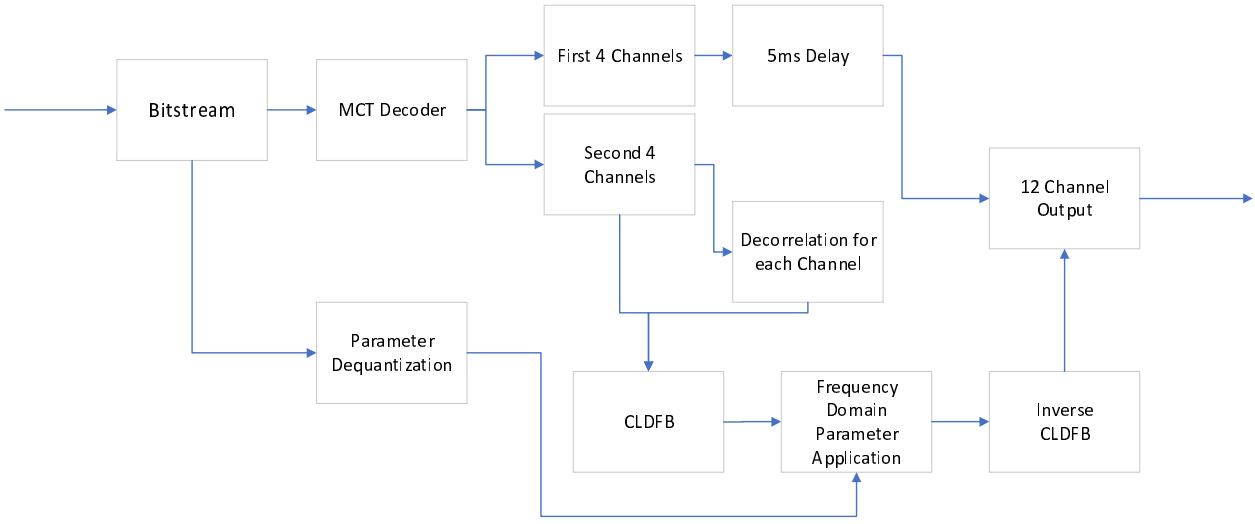


Figure 6.7-3: Schematic of parametric upmix decoder

In Parametric Upmix mode the bitstream consists of 7 MCT coded channels, an LFE coded channel, and metadata which is used to reconstruct the 7.1.4 multi-channel signal by upmixing to a total of 12 PCM channels including LFE. Before being processed by the Parametric Upmix block the 7 MCT coded channel are decoded by the MCT decoder 6.2.3.4 and the LFE channel is decoded by the LFE decoder 6.7.1. The first 3 MCT coded channels (Left, Right, Centre) and the LFE are grouped together as the "First 4 Channels" in Figure 6.7-3. The other 4 MCT coded channels ("Second 4 Channels") are grouped for separate processing described below. The first 4 channels call D_1 and correspond to section P_1 of input channels in the encoder, see clause 5.7.5.2. The second 4 channels" call D_2 and correspond to the downmix signals $M_{2,k}$, $k = 0 \ldots 3$ in the encoder.

6.7.4.2 Special case of Binaural, Stereo and Mono rendering configurations

6.7.4.2.1 Binaural rendering configuration

If rendering is set to Binaural then the final processing stage of clause 6.7.4.6 is altered and the processing of clause 6.7.4.7 is skipped. As opposed to all other rendering modes, Binaural rendering for Parametric Upmix multi-channel decoding mode is done in the CLDFB domain. In clause 6.7.4.6, after the 8 CLDFB domain paired channels have been changed to Left/Right representation, the 3 non-LFE channels from portion D_1 are transformed to CLDFB domain. The total of 11 CLDFB channels is then processed with the Fast convolutional binaural renderer, see clause 7.2.2.4. The two resulting CLDFB channels are then inverse transformed to time domain to become Left and Right channels. Finally, the LFE channel is mixed back in the time domain, see clause 7.4.7.1.

6.7.4.2.2 Stereo and Mono rendering configuration

If rendering is set to Stereo or Mono then the subsequently described parametric upmix decoding procedure is skipped and the following procedure followed instead: Each D_2 channel has a gain of 2.0 applied to it in order match levels when they would have been processed by Parametric Upmix. Then D_1 and the gain processed D_2 are propagated as a 5.1.2 multi-channel signal to downmix to Stereo or Mono processing described under 6.7.7.4.1.

6.7.4.3 Metadata Huffman Decoding

Otherwise, the metadata elements are decoded as follows.

Gain parameters called alpha and beta and defined in clause 5.7.5.3 are decoded from the metadata portion of the bitstream.

The decoding procedure starts with Huffman decoding the first alpha parameter in the bitstream, representing the absolutely coded alpha parameter quantization index for the lowest frequency band. Then each subsequent Huffman decoded value, each representing a higher frequency band, is added to the alpha quantization index from the previous frequency band in order to arrive at the current frequency band alpha quantization index. This is done until alpha quantization indices for all 12 frequency bands are recovered.

Beta quantization indices are then decoded following the identical procedure.

6.7.4.4 Dequantization

The alpha and beta quantizer indices (12 of each) are then used to dequantize alpha and beta parameters. Alpha parameters are dequantized with a simple table lookup using Table 5.7-29 in clause 5.7.5.4. Beta parameters are dequantized in a reverse process to the 2-stage quantization procedure in the encoder: for a given frequency band, the alpha quantizer index is used in a table lookup using Table 5.7-30 to identify the relevant beta quantization table. This lookup yields a beta table index that identifies the beta quantizer table to be used for beta parameter dequantization. The dequantized beta parameter is then found by looking up the dequantized beta parameter corresponding to the beta quantization index in the appropriate quantization table in the set of beta quantization tables provided in Table 5.7-31.

6.7.4.5 Decorrelation

The D_2 group of 4 channels from the MCT decoder are separated out and each channel is passed through the Decorrelator block (clause 6.2.1.3), with parameter α_{duck} set to 1.1, to form 4 additional channels.

The time domain decorrelator described in subclause 6.4.6.4.1 is used to separate the transient component of each original channel and generate a slowly-fluctuating decorrelated output. The 4 channels from the D_2 group and the

corresponding 4 decorrelated channels are then paired into 4 signal pairs, each original channel $D_{2,k}$ with its decorrelated counterpart $\widehat{D}_{2,k}$:

$$\left\{D_{2,k}, \widehat{D}_{2,k}\right\},\ k = 0 \ldots 3 \tag{6.7-112}$$

6.7.4.6 Upmixing operation

Each of the 8 paired channels is subsequently transformed into CLDFB domain, see clause 6.2.5.1, in which upmix, i.e., the inverse of the downmix operation described in clause 5.7.5.2 is carried out by applying the previously reconstructed alpha and beta parameters associated with the 12 frequency bands. For each of the channels, the transformation to CLDFB domain yields 16 time slots and 60 CLDFB bands. This leads to a CLDFB domain representation of the signal pairs defined in eq. (6.7-112) as follows:

$$\left\{d_{2,k}(b,\tau), \hat{d}_{2,k}(b,\tau)\right\},\ k = 0 \ldots 3\ , \tag{6.7-113}$$

with the band index b and the time slot index τ.

To apply the frequency band-wise alpha and beta parameters, the 12 frequency bands are mapped to the 60 CLDFB bands according to the following Table 6.7-1:

Table 6.7-1: Mapping between Frequency Bands and CLDFB Bands

Frequency Band	CLDFB Bands	Frequency Band	CLDFB Bands	Frequency Band	CLDFB Bands
0	0	4	4	8	12-15
1	1	5	5-6	9	16-21
2	2	6	7-8	10	22-31
3	3	7	9-11	11	32-59

The upmix operation for each CLDFB time slot and band is done by firstly generating a Mid/Side representation for each channel pair. The respective signals $d_{2,k}(b,\tau)$ are directly taken as Mid signal representation $m_k(b,\tau)$, which correspond to the four Mid channels $M_{2,k}\ , k = 0 \ldots 3$ defined in clause 5.7.5.2. The Side representation is constructed as a weighted combination of the signal pairs defined in eq. (6.7-112), i.e., input and output signals of the decorrelator. Smooth crossfading is facilitated with linear interpolation from alpha and beta parameters ($\alpha'(b)$ and $\beta'(b)$) of the previous frame to alpha and beta parameters ($\alpha(b)$ and $\beta(b)$) of the current frame:

$$s_k(b,\tau) = \left(\tfrac{\tau+1}{16}\alpha'(b) + \tfrac{15-\tau}{16}\alpha(b)\right)d_{2,k}(b,\tau) + \left(\tfrac{\tau+1}{16}\beta'(b) + \tfrac{15-\tau}{16}\beta(b)\right)\hat{d}_{2,k}(b,\tau)\ . \tag{6.7-114}$$

Each of the four pairs of Mid/Side representations are then converted to Left/Right channel representations according to

$$\begin{aligned} l_k(b,\tau) &= m_k(b,\tau) + s_k(b,\tau) \\ r_k(b,\tau) &= m_k(b,\tau) - s_k(b,\tau) \end{aligned}\ . \tag{6.7-115}$$

Finally, each of the 8 paired channels is transformed to time domain using the CLDFB synthesis operation, see clause 6.2.5.2.

6.7.4.7 Output

Because of the CLDFB domain upmix processing of the D_2 channels the resulting 8 channels are delayed relative to the 4 D_1 channels. Therefore, 5ms of corresponding delay is applied to the D_1 channels.

After delay adjustment operation, the resulting 12 time-domain signals represent the decoded 7.1.4 multi-channel signal.

6.7.4.8 Param Upmix PLC

During every frame without packet loss the dequantized alpha and beta parameters are stored for use in case of packet loss in a subsequent frame.

In a frame where the packet is lost and metadata is not available, the metadata Huffman decoding in clause 6.7.4.3 and the dequantization in clause 6.7.4.4 are skipped and the dequantized metadata parameters from the previous frame are used instead and applied for upmixing based on the (loss concealed) D_2 channels.

6.7.4.9 Param Upmix MC decoding with TSM

The render granularity for Param Upmix MC decoding is always 1.25 ms.

If the output format is mono or stereo the decoding of the transport channels, the decoding of the meta data, and the output to mono and stereo is done according to clause 6.7.4.2 before TSM. The transport channel buffer acts as a simple output buffer.

In all other cases the following steps are carried out before the TSM:

- LFE decoding
- Transport channel decoding and meta data decoding
- HP20 filtering

The TSM is applied to the transport channels and the LFE, the transport channel buffer has to hold the transport channels and the LFE. The interpolator for interpolating the upmix parameters is adapted to the number of slots to be rendered according to clause 6.2.7.4.3.2 Eq. (6.2-102) with all length in the equation expressed in CLDFB time slots. All remaining reconstruction and rendering steps are done after the TSM.

The decoder shall track the number of already rendered samples, time slots and subframes.

6.7.5 Discrete MC decoding mode

6.7.5.1 Overview

For the discrete MC coding mode the LFE channel is decoded separately as described in clause 6.7.1. The remaining channels are decoded using the MCT as described in clause 6.2.3.4.

6.7.5.2 Discrete MC PLC

The MCT PLC is described in clause 6.2.3.4.10.

6.7.6 MC bitrate switching

When switching to McMASA mode (see clause 6.7.2), the number of transport channels and the separate channel parameters are determined based on the new bitrate. Then the McMASA decoder is reconfigured by re-initializing all buffers and variables.

When switching to parametric MC mode (see clause 6.7.3) from a different MC mode, the correct number of transport channels is set and the necessary initializations of ParamMC buffers and variables are done. If the previous mode was also ParamMC the coder is reconfigured according to clause 6.7.3.2.

When switching to parametric upmix mode (see clause 6.7.4) from a different MC mode, the correct number of transport channels is set and the necessary initializations of paramUpmix buffers and variables are done.

When switching to DiscMC mode (see clause 6.7.5), the number of transport channels is set to the number of channels in the input MC layout.

Finally, a core-coder reconfiguration is performed to reflect possible changes in number of transport channels and per-channel bitrate.

Additionally, switching to a different bitrate in multi-channel format might also require a change of the used output renderer. The different rendering modes are described in clause 6.7.7. If the utilized renderer changes, the necessary initializations are performed.

6.7.7 MC output format conversion

6.7.7.1 Overview

For discrete multichannel, in case the output multichannel format is different from the input layout, the loudspeaker renderer performs a format conversion by applying a conversion matrix to generate the output channels. This conversion matrix can either be an upmix or downmix matrix depending on the input and output loudspeaker layouts. If the output layout is one of the supported IVAS formats (see clause 4.3.2) a conversion matrix $\mathbf{M}_{conv}$ is constructed using coefficients stored in ROM. If the output layout is a custom loudspeaker layout (see clause 7.4.9) then EFAP is used to construct the conversion matrix $\mathbf{M}_{conv}$. This matrix is of dimensions $N_{in} \times N_{out}$ where N_{in} is the number of input channels and N_{out} is the number of output channels.

Once this conversion matrix has been constructed, the matrix is used for time domain rendering by multiplication on the input channels to render the output channels. In case of ParamMC this matrix is used either in the covariance or CLDFB domain (see clause 6.7.3.6.3).

For Parametric Upmix mode, except for mono, stereo and binaural output formats, the 7.1.4 format of that coding mode is converted as described under clauses 6.7.7.4 - 6.7.7.5 to the respective target output format**.** For binaural output format the decoding is modified to binaural output format, as described in clause 6.7.4.2.1. For mono and stereo output formats the decoding is modified to output 5.1.2 format, as described in clause 6.7.4.2.2, and then further processing is applied as described under 6.7.7.4.1.

6.7.7.2 Common processing tools

6.7.7.2.1 Conversion of transport signals to TCX20 resolution

The MDCT scale factor bands for the loudspeaker conversion are initialized with a 20-ms resolution as for the SNS (cf. Table 5.3-12).

If the downmix signals in the MDCT domain are not of the same resolution, then an interleaving of two subframes is performed to convert the coefficients to a higher frequency resolution according to clause 6.3.6.2.3.5.

For signals with a TCX5 resolution this is performed twice to obtain TCX20 resolution, i.e. first to TCX10 then further to TCX20.

6.7.7.2.2 Computation of Equalization gain

After conversion of the downmix signals to 20-ms resolution and estimation of MDST spectra in clause 6.7.7.2.1, the target channel and downmix signal energies are computed according to equations (6.7-95) and (6.7-96) with the regular MC layout conversion matrix. The remaining procedure for gain computation follows clause 6.7.3.6.2.

The power of the downmix signal is summed up for a scale factor band as in equation (6.7-97) and the downmix and target energies are smoothed according to equations (6.7-98) and (6.7-99). Finally, the equalization gain g_B is computed with the smoothed energies:

$$g_{b_p} = \max\left(\min\left(\sqrt{\frac{\bar{P}_{out,b_p}}{\bar{P}_{dmx,b_p}+\epsilon}}, 2\right), 0.3\right)$$

6.7.7.3 Equalization of TCX Spectra

The TCX spectra are adjusted in their original frequency resolution with an equalization factor.

For TCX20 given a set of bandwise gain factors, the TCX spectra are adjusted based on the scale factor band borders:

$$X^{TCX20}(b) = g_B X^{TCX20}(b)$$

where $b = b_{start} \ldots b_{end}$ are the starting and ending bins of the scale factor band and B is the scale factor band index.

For TCX10 resolution, the scale factor band borders are halved, i.e. $b = 0.5(b_{start} \ldots b_{end})$. For TCX5 resolution, the interleaving based on clause 5.3.2.4.2 of [3] is taken into account for the band border adjustment.

6.7.7.4 Rendering to a supported IVAS format

6.7.7.4.1 General

The loudspeaker matrices for supported IVAS formats are constructed using coefficients stored in ROM in a compressed form. Depending on the combination of input and output format, a corresponding table is chosen, and the coefficients are used to reconstruct the conversion matrix $\mathbf{M}_{conv}$.

6.7.7.4.2 Mono and Stereo output

For Mono and Stereo output, the downmix coefficients of each speaker of the 7.1+4 layout are stored since these are the superset of loudspeaker positions for supported formats. For each channel of the input layout, a check is performed to determine whether it matches the channel position of the 7.1+4 layout, if they match, then the corresponding row is used in $\mathbf{M}_{conv}$. The Tables 6.7-2 and 6.7-3 contain the exact coefficients used for this purpose.

Table 6.7-2: Downmix coefficients for Mono output

Channel position (azimuth, elevation in degrees)	Downmix coefficient
30, 0	1
-30, 0	1
0, 0	1
LFE	1
135, 0	0.79999995
-135, 0	0.79999995
90, 0	0.79999995
-90, 0	0.79999995
30, 35	0.849999964
-30, 35	0.849999964
110, 35	0.849999964
-110, 35	0.849999964

Table 6.7-3: Downmix coefficients for Stereo output

Channel position (azimuth, elevation in degrees)	Downmix coefficient (Left channel)	Downmix coefficient (Right channel)
30, 0	1	0
-30, 0	0	1
0, 0	0.70710677	0.70710677
LFE	0.70710677	0.70710677
135, 0	0.79999995	0
-135, 0	0	0.79999995
90, 0	0.79999995	0
-90, 0	0	0.79999995
30, 35	0.849999964	0
-30, 35	0	0.849999964
110, 35	0.849999964	0
-110, 35	0	0.849999964

6.7.7.4.3 Remaining formats

Since the conversion matrix is usually sparse, only elements which are non-zero are stored to ROM and used to reconstruct the matrix as required. The coefficients are stored in a compressed table in which the value in the first row and first column indicates the number of non-zero matrix elements and the value in the first row and second column indicate the number of columns of the matrix (i.e. N_{out}). Subsequent rows contain the indices of the values if the matrix were a flattened array and the values themselves.

The compressed upmix and downmix coefficient tables for specific format pairs are detailed in Tables 6.7-4 to 6.7-18.

Table 6.7-4: 7.1 to 5.1 Downmix coefficients

Index	Value
8	6
0	1
7	1
14	1
21	1
28	1
35	1
40	1
47	1

Table 6.7-5: 5.1+2 to 5.1 Downmix coefficients

Index	Value
8	6
0	1
7	1
14	1
21	1
28	1
35	1
36	0.849999964
43	0.849999964

Table 6.7-6: 5.1+2 to 7.1 Downmix coefficients

Index	Value
8	8
0	1
9	1
18	1
27	1
36	1
45	1
48	0.849999964
57	0.849999964

Table 6.7-7: 5.1+4 to 5.1 Downmix coefficients

Index	Value
10	6
0	1
7	1
14	1
21	1
28	1
35	1
36	0.849999964
43	0.849999964
52	0.849999964
59	0.849999964

Table 6.7-8: 5.1+4 to 7.1 Downmix coefficients

Index	Value
10	8
0	1
9	1
18	1
27	1
36	1
45	1
48	0.849999964
57	0.849999964
68	0.849999964
77	0.849999964

Table 6.7-9: 5.1+4 to 5.1+2 Downmix coefficients

Index	Value
10	8
0	1
9	1
18	1
27	1
36	1
45	1
54	0.849999964
63	0.849999964
68	0.849999964
77	0.849999964

Table 6.7-10: 7.1+4 to 5.1 Downmix coefficients

Index	Value
14	6
0	1
7	1
14	1
21	1
28	1
35	1
36	0.367322683
40	0.930093586
43	0.367322683
47	0.930093586
48	0.849999964
55	0.849999964
64	0.849999964
71	0.849999964

Table 6.7-11: 7.1+4 to 7.1 Downmix coefficients

Index	Value
14	8
0	1
9	1
18	1
27	1
38	1
47	1
48	0.367322683
52	0.930093586
57	0.367322683
61	0.930093586
64	0.849999964
73	0.849999964
84	0.849999964
93	0.849999964

Table 6.7-12: 7.1+4 to 5.1+2 Downmix coefficients

Index	Value
14	8
0	1
9	1
18	1
27	1
36	1
45	1
48	0.367322683
52	0.930093586
57	0.367322683
61	0.930093586
70	1
79	1
84	0.849999964
93	0.849999964

Table 6.7-13: 7.1+4 to 5.1+4 Downmix coefficients

Index	Value
14	10
0	1
11	1
22	1
33	1
44	1
55	1
60	0.367322683
64	0.930093586
71	0.367322683
75	0.930093586
86	1
97	1
108	1
119	1

Table 6.7-14: 7.1 to 5.1+2 Upmix coefficients

Index	Value
8	8
0	1
9	1
18	1
27	1
36	1
45	1
52	1
61	1

Table 6.7-15: 7.1 to 5.1+4 Upmix coefficients

Index	Value
8	10
0	1
11	1
22	1
33	1
44	1
55	1
64	1
75	1

Table 6.7-16: 7.1 to 7.1+4 Upmix coefficients

Index	Value
8	12
0	1
13	1
26	1
39	1
54	1
67	1
76	1
89	1

Table 6.7-17: 5.1+2 to 7.1+4 Upmix coefficients

Index	Value
8	12
0	1
13	1
26	1
39	1
52	1
65	1
80	1
93	1

Table 6.7-18: 5.1+4 to 7.1+4 Upmix coefficients

Index	Value
10	12
0	1
13	1
26	1
39	1
52	1
65	1
80	1
93	1
106	1
119	1

For upmix matrices, there may be a special case where the output format is a complete superset of the input format. In such cases there are no coefficients stored in ROM, but instead the conversion matrix is simply a routing of existing input channels to the corresponding valid output channels with unity gain.

All valid mappings of input and output formats are stored in a separate table. During the initialization of the MC format conversion module, this table is used to look up the corresponding table of compressed matrix coefficients. The valid mappings are specified in Table 6.7-19, except for Mono and Stereo output for which valid mappings are specified, but the procedure described in clause 6.7.7.2.1 is used.

Table 6.7-19: Valid MC format conversion mappings

Input Format	Output Format	Compressed Table
5.1	7.1	N/A; Superset
5.1	5.1+2	N/A; Superset
5.1	5.1+4	N/A; Superset
5.1	7.1+4	N/A; Superset
7.1	5.1	Table 6.7-4
7.1	5.1+2	Table 6.7-14
7.1	5.1+4	Table 6.7-15
7.1	7.1+4	Table 6.7-16
5.1+2	5.1	Table 6.7-5
5.1+2	7.1	Table 6.7-6
5.1+2	5.1+4	N/A; Superset
5.1+2	7.1+4	Table 6.7-17
5.1+4	5.1	Table 6.7-7
5.1+4	7.1	Table 6.7-8
5.1+4	5.1+2	Table 6.7-9
5.1+4	7.1+4	Table 6.7-18
7.1+4	5.1	Table 6.7-10
7.1+4	7.1	Table 6.7-11
7.1+4	5.1+2	Table 6.7-12
7.1+4	5.1+4	Table 6.7-13

First, the conversion matrix $\mathbf{M}_{conv}$ is initialized to all zeros. Next, using the compressed representation specified in the mapping, the elements of the conversion matrix are set using the flattened index i and value v specified in the compressed table to set elements of the conversion matrix as follows:

$$a = \mathrm{i} \div \mathrm{N_{out}}$$

$$b = i(\mathrm{mod}\ N_{out})$$

$$m_{conv,ab} = v$$

Where a is the row index, b is the column index, and mod is the modulo operation.

6.7.7.5 Rendering to a custom loudspeaker layout

For custom loudspeaker output, EFAP (cf. clause 7.2.1.3) is initialized for the output layout, and each input speaker is panned as a virtual object on the output layout. EFAP returns a set of gains which are used to populate the appropriate row in the conversion matrix.

If an input LFE channel is present, it is routed to the output LFE channel with unity gain if present, otherwise the input LFE is mixed to all output channels with a factor of $\frac{1}{N_{in}-N_{inLFE}}$. Where N_{inLFE} is the number of input LFE channels.

6.7.7.6 Rendering to binaural output with head tracking

A detailed overview of binaural rendering is provided in clause 7.2.2.1. As described in Table 7.2-1, when the following criteria are fulfilled:

- Head-tracking is enabled
- Input format is MC
- IVAS Mode is ParamMC, ParamUpmix or Discrete MC with non-planar layout input
- Output format is BINAURAL or BINAURAL_ROOM_REVERB

an intermediate conversion of the signals to the spherical harmonic domain is performed to allow for a lower-complexity rotation for head-tracking.

In this case the decoded multichannel signals are rendered to 3rd order Ambisonics (using the real spherical harmonic response via table look up) and rotation in SHD (clause 6.4.6.5.6.2) is applied prior to binaural rendering. The respective renderers then render the signal from the HOA3 format to binaural.

In other cases, rotation is handled as normal by the respective renderer.

6.7.8 MC decoding with TSM

The decoding with TSM for McMASA is described in clause 6.7.2.6.

The decoding with TSM for ParamMC is described in clause 6.7.3.9.

The decoding with TSM for Parametric upmix MC is described in clause 6.7.4.8.

If the MC mode is discrete MC coding and the output format is mono and stereo the complete decoding and rendering to mono and stereo is done before the TSM and the transport channel buffer acts as a simple output buffer.

If the MC mode is discrete MC for all other output formats the following steps are done before TSM:

- LFE decoding
- MC channel decoding
- HP20
- Output format conversion to output format or intermediate format if the format is either a LS or SHD format and the number of channels in the output format is less than the number of channels of the input format

All remaining steps for discrete MC are done after the TSM. The rendering granularity is 5 ms if the output is binaural and either CRend or the TD object renderer are used, otherwise it is 1.25 ms.

If a bit rate switch is occurring and the render granularity changes from 5 ms to 1.25 ms a render flush according to 6.2.7.5.1 is done.

The transport channel buffer is reconfigured on a bit rate switch if the number of channels to be processed by the TSM or the render granularity changes.

If a rate switch is occurring and the render granularity changes from 1.25 ms to 5 ms, a rendering with discarded samples according to clause 6.2.7.5.3 is done.

The decoder shall keep track of the already rendered samples, time slots, and subframes.

6.8 Combined Object-based audio and SBA (OSBA) decoding

6.8.1 Low-bitrate pre-rendering OSBA decoding mode

In the low-bitrate mode, the OSBA processing is on the decoder side is mostly identical to that of regular SBA format decoder. All objects are rendered into the SBA input signal on the encoder side. Therefore, it is sufficient to decode and render the SBA signal in the same way as in SBA format.

However, the external output format supported additionally. This format consists of the object audio channels plus metadata files and the SBA output channels. In the low bitrate mode, the correct number of output channels and metadata files are produced by the encoder. However, the contents of the objects are part of the SBA output channels. The object channels are output empty, and the metadata are default values. This output format enables compatibility with the same postprocessing or external rendering as in the high-bitrate mode. It also avoids the need to implement a bitrate dependent post-processing or rendering of the external output.

6.8.2 High-bitrate discrete OSBA decoding mode

In the high-quality high-bitrate mode, the SBA and ISM metadata decoders run concurrently, each of them separately decoding their metadata. The MCT decodes the jointly coded audio channels of the objects and the SPAR downmix.

The SPAR upmix and the DirAC synthesis then run as in regular SBA. The objects are rendered to the selected output configuration. A factor of 0.5 is multiplied to the outputs in order to keep the loudness at approximately the same level as if both formats are encoded and decoded with two separate instances of the codec.

In external output mode, the decoded ISM metadata are written to one output file per object. The audio output files contain the object audio channels and the SBA output channels.

6.8.3 OSBA PLC

For low-bitrate pre-rendering OSBA coding mode, the OSBA decoding is mostly identical to the SBA decoding and, consequently, PLC is handled exactly the same way as in clause 6.4.8.

For high-bitrate discrete OSBA coding mode, PLC processing is performed in the SBA and ISM metadata decoders according to clauses 6.4.8 and 6.6.5, respectively, and the MCT according to clause 6.2.3.4.10.

6.8.4 OSBA bitrate switching

In OSBA format, bitrate switching entails re-configuration of both the SBA and ISM decoders. The configuration is the same as if these two decoders were running in separate instances of the IVAS decoder. This is described in clauses 6.4.9 for SBA and 6.6.6 for ISM.

A special case for OSBA is the switching between bitrates employing different OSBA coding modes. Then the decoder switches between the pre-rendering and the discrete coding modes.

When the high-bitrate mode is switched on, additional re-configurations are required as compared to bitrate switching for SBA in clause 6.4.9. Specifically, the number of MCT channels is set according to the SBA configuration and the number of objects. The ISM mode flag is set to signal discrete object coding.

When the high-bitrate mode is switched off, the ISM mode flag is set to signal pre-rendering mode.

6.8.5 OSBA output format conversion

In the pre-rendering OSBA coding mode, the decoder-side processing is identical to that in SBA format. The objects are pre-rendered into the SBA scene on the encoder side. Consequently, the output format and the specific processing associated with it are the same as described in clauses 6.4.10 and 6.4.6.5.8.

In the discrete OSBA coding mode, the output is generated by the SBA and ISM decoders concurrently. Hence, both decoders must be configured to provide the requested output format. The signals from both decoders are then summed up. The SBA output processing is again performed according to clause

6.8.6 OSBA decoding with TSM

The processing of the time-scale modified frames in the OSBA decoder follows from the processing in the ISM and SBA format.

In the low-bitrate OSBA decoding mode (bitrates < 256 kbps), only the SBA decoder runs, and the audio contents of the object are pre-rendered into the SBA signal on the encoder side. Hence, the same processing is performed as in clause 6.4.11.

In the high-bitrate OSBA decoding mode (bitrates ≥ 256 kbps), the additional transport channels containing the object audio are decoded by MCT. These channels are additionally written to the transport channel buffer. Additionally, the ISM and SBA metadata are decoded. Specifically, the high-bitrate OSBA transport-channel decoding stage comprises the following steps:

1. ISM metadata decoding (see clause 6.6.3)
2. SPAR MD decoding (see clause 6.4.3)
3. DirAC MD decoding (see clause 6.4.2)
4. Audio-channel decoding using SCE, CPE, or MCT
5. If STEREO output is configured

a. AGC decoding (see clause 5.4.7)

b. PCA decoding (see clause 5.4.6)

6. Calculation of the SPAR upmix matrix according to 6.4.5

    a. HP20 filtering of the transport channels

7. If parametric binaural or binaural room rendering is configured

    a. SBA to binaural mixing matrix is calculated (see clause 7.2.2.3)

    b. Loudness correction is applied to the SBA TCs

8. If rendering is not disabled or configured to stereo

    a. AGC and PCA are decoded (see clauses 5.4.7and 5.4.6)

9. If mono output is configured

    a. The W channel is copied to the transport-channel buffer

    b. ISMs are downmixed into a mono channel (see clause 6.6.7.2)

After this decoding, the transport channels are processed by the TSM and the metadata are interpolated. The time-scale modified audio channels are stored in the transport channel buffer according to according to clause 6.2.7.2. Local subframes are calculated according to 6.2.7.4.3.1 and fed to the rendering stage.

The metadata mapping $m_{ts,SPAR}$ for the SPAR upmix parameters is determined according to clause 6.4.11. Again the mapping is different for the high- and low-order DirAC modes. The ISM metadata are interpolated according to clause 6.6.7.

Unlike in SBA format, the granularity in OSBA format is set 5 ms as the ISMs are always processed with this time resolution and the lower resolution of both modes must be used.

The rendering stage is again equivalent to that of SBA when the bitrate is < 256 kbps. For higher bitrates, ISM rendering and SPAR and DirAC decoding run concurrently. This case involves the following steps for each 5 ms subframe.

1. If BINAURAL output is configured

    a. DirAC decoding is applied according to 6.4.6.5.3 or 6.4.6.5.4

    b. Decoded HOA3 output is rendered binaurally using FastConv (see clause 7.2.2.4)

    c. ISM objects are rendered to binaural using the TD renderer (see clause 7.2.2.2)

    d. Rendered outputs are multiplied by 0.5 and added up

2. If STEREO output is configured

    a. ISM objects are rendered to stereo (see clause 6.6.7)

    b. The SBA stereo stream generated in the TC decoding phase is loaded from the TC buffer

    c. These two outputs are added up

3. If MONO is configured

    a. No further processing is required as the output is already in the TC buffer

4. If FOA/HOA or MC output is configured

    a. SBA and ISM are rendered to the configured output format MC (see clauses 6.4.10 and 6.6.7)

    b. If BINAURAL_ROOM output is configured

        i. SBA and ISM are rendered to MC (see clauses 6.4.10 and 6.6.7)

c. MC is binauralized using the FastConv renderer (see clause 7.2.2.4)

5. If EXT output is configured

   a. SBA is rendered to HOA3 (see clauses 6.4.6.5.3 or 6.4.6.5.4)

   b. Object audio channels are copied from the transport-channel buffer to the output buffer

6.9 Combined Object-based audio and MASA (OMASA) decoding

6.9.1 OMASA format decoder overview

The combined OMASA format decoder block diagram is shown in Figure 6.9-1.

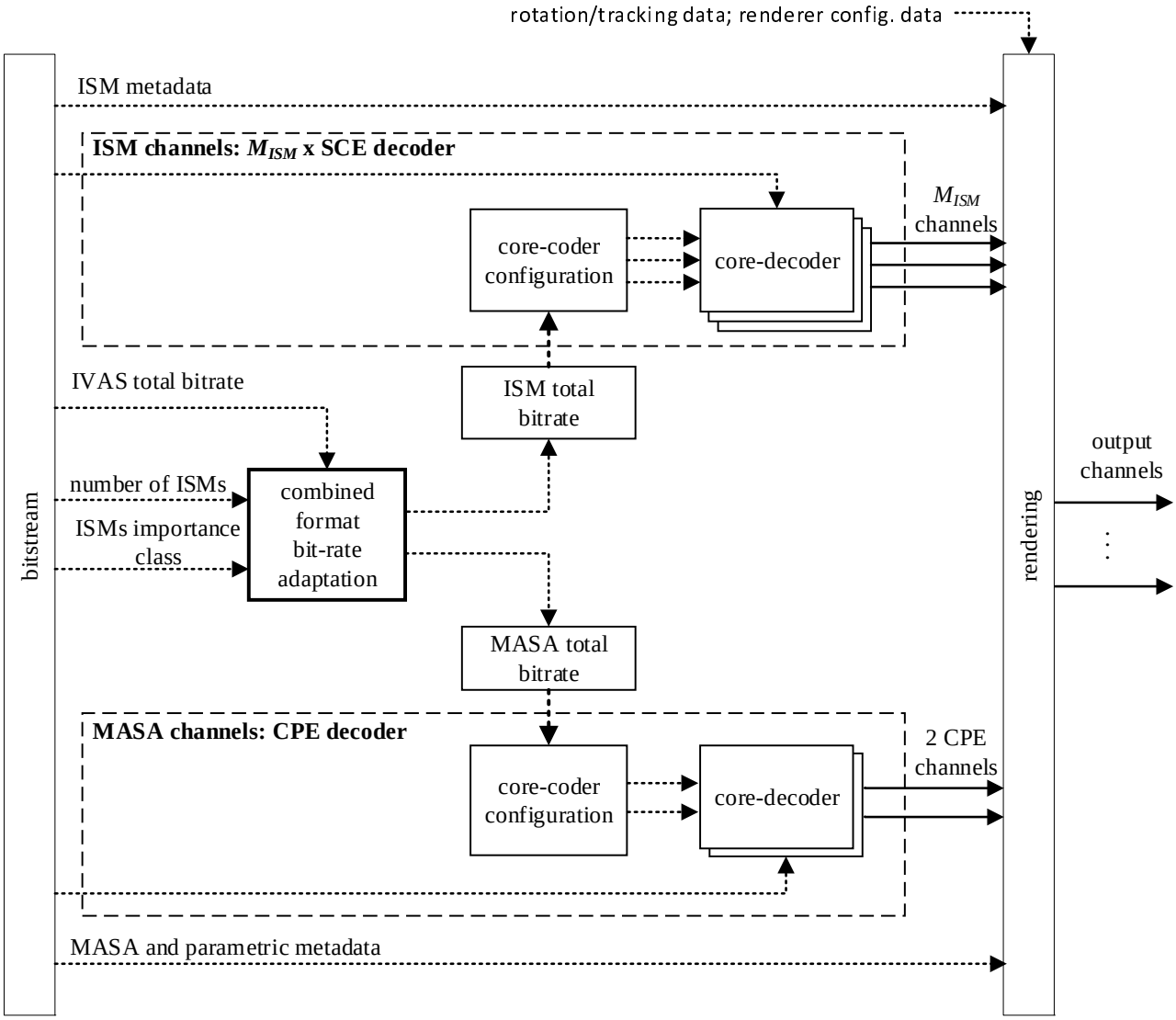


Figure 6.9-1: Block diagram of OMASA decoder

The combined OMASA format decoder from Figure 6.9-1 receives a bitstream that contains indices related to several codec modules, including audio format signalling, ISM transport channels (M_{ISM} x SCE decoding tool indices), ISM metadata, MASA transport audio channels (CPE tool indices), MASA metadata, and the combined format signalling. First, the decoder reads and decodes from the received bitstream information about the audio format. In case of the combined OMASA format, the decoder next reads the information needed to set the individual format bitrates.

In case of OMASA, the decoder reads the number of ISMs and the ISM importance class (one parameter per ISM). These parameters and the IVAS total bitrate parameter are then used to set the OMASA coding mode and to perform the combined format bitrate adaptation the same way as at the encoder (see clause 5.9.8). The output from this logic is the adapted ISM total bitrate $brate_{ism}(m)$, $m = 1, \ldots, M_{ISM}$, and the MASA total bitrate $brate_{MASA}$ parameters which are further used to configure the core-decoders for the ISM and MASA decoding. The core-decoders for ISM and MASA parts are then processed sequentially and they output M_{ISM} synthesis corresponding to M_{ISM} separately coded ISMs plus two MASA synthesis. It is noted that ISM core-decoders consist of SCE decoder tools (clause 6.2.3.2) while the MASA transport channels are decoded using the CPE decoder tool (clause 6.2.3.3). Further, ISM metadata are decoded using ISM metadata decoding (clause 6.6.3) and MASA metadata are decoded using MASA metadata decoding (clause 6.5.3).

Finally, the M_{ISM}+ 2 synthesis channels together with the decoded ISM and MASA metadata are all fed to the renderer which produces the final spatial sound in a desired output audio format (e.g. binaural, multi-channel, etc.).

6.9.2 Low-bitrate pre-rendering (Rend OMASA) decoding mode

The low-bitrate pre-rendering mode is signalled as MASA format and decoded as such, with the following differences:

- The 2 reserved bits from the MASA metadata frame are checked, and if they are not ‘00’ they are interpreted as it is specified in clause 5.9.4.2.
- When the two reserved bits are read as ‘11’, this signals 1 or 2 objects. In this instance the bit for the number of transport channels (the MASA number of transport channel signal bit) is also read in order to distinguish between 1 audio object and 2 audio objects. One audio object is signalled by ‘0’ and 2 audio objects are signalled by ‘1’. When the MASA number of transport channel signal bit is ‘0’ this indicates the case of 1 input audio object, and when the MASA number of transport channel signal bit is ‘1’ this indicates the case of 2 input audio objects.

When the two reserved bits from the MASA metadata frame are read as either- ‘01’ signifying 4 objects or - ‘10’ signifying 3 objects, then the MASA number of transport channels signal bit is used for the decoding of the combined MASA format audio signal.

6.9.3 One object with MASA (One OMASA) decoding mode

When the IVAS format is signalled as OMASA, the decoder reads the number of audio objects (i.e. the number of audio objects at the input to the encoder), the importance flag of the separated object and the ISM related flags signalling the presence of ISM metadata (see clause 5.6.5.2). Knowing the IVAS total bitrate and the number of audio objects, the decoding mode is obtained from table 5.9-1, along with the bitrate allocated to the separated object and to the MASA format data. The MASA configuration is realized based on the nominal bitrate initially allocated to the MASA part from the configurations table. The MASA metadata is next decoded, from the end of the bitstream, according to the procedures described in clause 6.5.3. After decoding the MASA metadata, the bit allocation between the one separated object and the MASA content is adjusted to conform to the procedure of clause 5.9.8. The separated object together with its metadata is then decoded. The last part of the decoding obtains the decoded MASA transport audio signal channels according to clause 5.5.4. In OMASA there are always 2 transport channels.

6.9.4 Parametric one object (Param OMASA) decoding mode

6.9.4.1 Overview

The one audio object with parametric representation mode decodes, in addition to the decoded parameters from sub clause 6.9.3, the parametric mix representation parameters consisting of the MASA-to-total energy ratios and the ISM energy ratios. The direct-to-total energy ratios are obtained as presented in clause 6.9.6 and are subsequently used for rendering the OMASA content which is presented in clause 6.9.7.

The overall decoding procedure for OMASA, is described by the following flow:

- IVAS format is read
- Number of objects is read
- The index of the separated object is read

- The number of MASA directions is read on 1 bit
- The MASA-to-total energy ratios are decoded (see clause 6.9.4.2)
- The ISM energy ratios are decoded (see clause 6.9.4.3)
- The separated object metadata is decoded (see clause 6.9.4.4)
- The MASA metadata is decoded (see clause 6.5.3)

6.9.4.2 MASA-to-total ratios decoding

The decoding of the encoded data corresponding to the MASA-to-total energy ratios depends on the number of sub frames and sub bands, which can result in one of 4, 5, 8, 12, 20, or 32 indexes being read. The case of 32 indexes is read in 4 streaks of 8 indexes each.

Reading of one streak is as follows:

1. If it is the single streak or the first one from the group of 4
    a. The sign of the DCT coefficient of order 0 is read (1 for positive, 0 for negative)
2. End if
3. Read on the next 6 bits the value of the first DCT coefficient of the streak
4. Multiply the coefficient with its sign
5. If the first DCT coefficient is not null
    a. If the length of the streak is larger than 8
        i. Read on 4 bits the position of the last index, *i_min*, that has been encoded with Golomb Rice of order GR2
        ii. Read first Golomb Rice order GR1 on 1 bit; GR1 is 1 or 2
        iii. If GR1 == 2
            1. Read 1 bit for GR2 (0 or 1)
        iv. Else
            1. GR2 = 0
        v. End if
        vi. Decode *i_min* indexes with Golomb Rice decoder with order GR2
        vii. Decode the rest of indexes with Golomb Rice decoder of order GR1
    b. Else
        i. Read Golomb Rice order, GR1, on 1 bit (1 or 0)
        ii. Decode all remaining indexes with Golomb Rice decoder of order GR1
    c. End if
    d. Reorder the indexes and dequantize the DCT coefficients using the decoded indexes and the quantization step of 0.1.
6. Else
    a. All DCT coefficients are null
7. End if

8. Group the coefficients into a matrix

9. Inverse DCT transform the matrix to obtain the decoded MASA-to-total energy ratios

6.9.4.3 ISM energy ratios decoding

The decoder is configured to receive encoded ISM energy ratio indexes relating to each audio object, which on a per audio object basis consists of the ISM energy ratio index corresponding to each sub frame and sub band of the audio frame. The encoded ISM energy ratio indexes for each audio object are then decoded to give the ISM energy ratio quantization index for each sub frame and sub band of the audio frame. Each quantization index is then used to retrieve the respective corresponding quantized ISM energy ratio value.

Before the decoding procedure, a number of verifications are first performed.

In a first verification, the MASA to total energy ratio of each TF tile is compared against a threshold (whose value is 0.98) in order to determine whether the ratio value if greater than the threshold. If it is determined that all MASA to total energy ratios of the frame are greater that the threshold then the ISM energy ratio indexes are determined to be evenly distributed across the TF tiles of the audio frame such that the ISM energy ratio indexes of the frame sum to K in a manner similar to that laid out in step 1.a of clause 5.9.6.3.3. However, if it is determined that the MASA to total energy ratios for the TF tiles of the frame are not all above the threshold then the encoded information relating to the quantized ISM energy ratio index for each TF tile of the frame is read from the bitstream and decoded.

In a second verification, a combination (for the frame) of whether the separated audio object is the last audio object and the number of audio objects is greater than two is checked. If the check is in the affirmative, then the index corresponding to the ISM energy ratio of the separated audio object is set to zero.

Since the encoding procedure is a combination of absolute coding for the first sub frame, followed by differential coding for the following sub frames of the frame, data from previous sub frames is then stored during the decoding process. The generic decoding procedure is as follows:

1. $T = N_{obj} - 1$

2. For *sf* = 1:*nblocks*

    a. Read and decode information relating to the quantized ISM energy ratio index vectors for all sub bands for *T* audio objects to give the quantized ISM energy ratio index vectors for all the sub bands and the T objects of the subframe.

    b. Save current subframe quantized ISM energy ratio index vectors for use as a previous sub frame data

    c. If $N_{obj} > 2$ and the separated object is the last audio object

        i. Interchange the quantized ISM energy ratio index for the first audio object with the last audio object.

    d. End if

    e. If *sf ==1* and all decoded quantized ISM energy ratio indexes of sub bands are zero

        i. $T = N_{obj} - 2$

    f. Else

        i. $T = N_{obj} - 1$

    g. End if

    h. Reconstruct quantized ISM energy ratios from the quantized ISM energy ratio indexes

3. End for

The decoding uses one of the following: a deindexing procedure, a differential decoding based on previous sub band or a differential decoding based on the previous sub frame.

The deindexing procedure is applied to the value *index* which is read from the bitstream as a fixed number of bits depending on the total number objects. The number of bits read is one of 3, 6, or 7 corresponding to the number of objects of 2, 3, and 4 respectively. The deindexing procedure, for the first subframe is presented in the following pseudo-code and is executed as step 2.a in the above generic decoding procedure. The result is the group or vector of indexes which sum up to $K = 2^{nb} - 1$, and represent the indexes of the quantized ISM energy ratios for all the objects, for one TF tile. The number *nb* is the number of bits used for the quantization of an individual ISM energy ratio. N_{obj} is the total number of audio objects.

6.9.4.4 ISM metadata decoding

Encoding of the directional parameter for each audio object over a frame occurs when the IVAS codec is operating in the one separated audio object with parametric representation coding mode.

In this mode, for the decoding of the audio object directional parameter, the MASA-to-total energy ratios for all the TF tiles in the frame are initially decoded together with the ISM energy ratios for all audio objects and for all TF tiles in the frame. These decoded values are then used to derive the priority values for all the audio objects in the frame according to equations (5.9-1) and (5.9-2).

A bit allocation for the directional parameter of each audio object is then calculated using the priority value calculated for the respective audio object according to equation (5.9-3).

The directional parameter (for each audio object) is then decoded using the calculated respective bit allocation. This is performed by the following procedure:

If the bit allocation is less than a threshold bit allocation value of 8 bits, a signal bit is read. If the value of the signal bit is 1, this indicates that the current audio object's azimuth and elevation directional parameter values are the same as the directional parameter values of the audio object from of the previous frame. However, if the value of the bit is zero, then a spherical grid quantizer is used to decode/de-index the azimuth and elevation values of the directional parameter of the audio object. The quantizer is based on the remaining bits of the bit allocation after the signal bit has been taken into account.

If the bit allocation is larger or equal to the threshold bit allocation of 8 bits, then the spherical direction index for the azimuth and elevation values of the directional parameter of the audio object decoded/de-indexed using a spherical grid quantizer based on the bit allocation number of bits for the directional metadata.

For the case when the bit allocation is lower than 8 and the directional metadata differs from the previous frame values, an additional smoothing procedure is performed for the azimuth value as detailed below:

1. For each object *i*
    a. If *bits_ism(i)* < 8
        i. Read one bit, *d*, from bitstream
        ii. If *d* == 1
            1. Direction is same as for previous frame
        iii. Else
            1. Read the direction index for the corresponding number of bits
            2. Decode the direction index into the quantized azimuth and elevation, $\hat{\phi}\ and\ \hat{\theta}$
            3. If $\hat{\phi} \cdot \hat{\phi}_{prev} > 0$
                a. $\Delta_\phi = \frac{360}{n_\phi}$
                b. If $\hat{\phi} - \hat{\phi}_{prev} > \frac{\Delta_\phi}{2}$
                    i. $\hat{\phi} = \hat{\phi} - \frac{\Delta_\phi}{2}$
                c. Else if $\hat{\phi}_{prev} - \hat{\phi} > \frac{\Delta_\phi}{2}$

i. $\hat{\phi} = \hat{\phi} + \frac{\Delta\phi}{2}$

d. End

4. End if

iv. End if

b. Else

i. Read the direction index for the corresponding number of bits

ii. Decode the direction index into $\hat{\phi}\ and\ \hat{\theta}$

c. End if

d. $\hat{\phi}_{prev} = \hat{\phi}$

2. End for

6.9.5 Discrete (Disc OMASA) decoding mode

In the Disc OMASA coding mode, first the adapted ISM total bitrate and adapted MASA total bitrate are set in the inter-format adaptation module (clause 5.9.8). Then, in a sequential order, the ISM format decoder (clause 6.6) decodes the M_{ISM} ISM channels including their metadata followed by the MASA format decoder (clause 6.5) decoding the two MASA transport channels including their metadata. The transport audio channels and metadata related to both ISM and MASA parts are supplied to the decoder and rendered to the requested output configuration.

6.9.6 OMASA rendering metadata generation

In the one object with parametric representation decoding mode the following is performed.

The direct-to-total energy ratios $r_{ISM,dir}(k,m,i)$ associated with the objects are determined using the decoded ISM energy ratios $rISM(k,m,i)$ and the decoded MASA-to-total energy ratios $r_{MASA2tot}(k,m)$

$$r_{ISM,dir}(k,m,i) = rISM(k,m,i)\big(1 - r_{MASA2tot}(k,m)\big)$$

where i is the object index, k is the frequency bin index, and m is the temporal subframe index.

The decoded object directions are set to the azimuth $\theta_{ISM}(m,i)$ and elevation $\phi_{ISM}(m,i)$ variables.

The spread coherence is set to zero $\zeta_{ISM}(m,i) = 0$.

The determined rendering metadata ($\theta_{ISM}(m,i)$, $\phi_{ISM}(m,i)$, $r_{ISM,dir}(k,m,i)$, and $\zeta_{ISM}(m,i)$) is used in the rendering, see clause 6.9.7 (and related clauses, such as 7.2.2.3.3 and 6.5.7.2.3).

The direct-to-total energy ratios $r_{dir}(k,m,i)$ associated with the MASA part is modified using the MASA-to-total energy ratios $r_{MASA2tot}(k,m)$

$$r_{dir}(k,m,i) \coloneqq r_{dir}(k,m,i) r_{MASA2tot}(k,m)$$

The modified direct-to-total energy ratios $r_{dir}(k,m,i)$ is used in the rendering, see clause 6.9.7 (and related clauses, such as 7.2.2.3.3 and 6.5.7.2.3).

6.9.7 OMASA rendering

6.9.7.1 Binaural rendering

6.9.7.1.1 Discrete coding mode

First, the separately coded audio signals are separated to a separate signal $s_{sep}(n,i)$, having $N_{sep} = N_{obj}$ channels. The MASA transport audio signals are separated to another signal $s_{MASA}(n,i)$, having 2 channels.

The MASA transport signals $s_{MASA}(n,i)$ are rendered to binaural output signals $s_{MASArend}(n,i)$, using the parametric binaural renderer presented in clause 7.2.2.3.

The separated signals $s_{sep}(n,i)$ are attenuated by a gain of 0.7943 and delayed by 5 milliseconds to match the delay caused by the CLDFB processing of the parametric binaural renderer. The resulting signals are $s_{g,del}(n,i)$ are rendered to binaural output signals $s_{seprend}(n,i)$, using the time-domain binaural renderer presented in clause 7.2.2.2.

Then, the rendered output signals from the two renderers are combined

$$s_{out}(n,i) = s_{MASArend}(n,i) + s_{seprend}(n,i)$$

forming the output signals of the rendering process.

6.9.7.1.2 Other coding modes

Binaural (with and without a room effect) rendering is performed using the parametric binauralizer and stereo renderer presented in clause 7.2.2.3.

6.9.7.2 Stereo rendering

Stereo rendering is performed using the parametric binauralizer and stereo renderer presented in clause 7.2.2.3.

6.9.7.3 Multi-channel loudspeaker and Ambisonics rendering

6.9.7.3.1 Overview

The multi-channel loudspeaker and Ambisonics (SBA) rendererrenders multi-channel loudspeaker and Ambisonics output signals from decoded OMASA transport audio signals, separated object signals, spatial metadata, and object metadata.

The rendering is performed in subframes, where m denotes the subframe index. A subframe contains N_{slots} CLDFB slots (in non-JBM operation, $N_{slots} = 4$, in JBM operation $N_{slots} = 1 \ldots 7$). The data determined at previous subframes (i.e., subframes m-1 and earlier) affects the rendering of the present subframe m due to temporal averaging and interpolation.

The fetching of temporally correct spatial metadata parameter values for the current subframe m so that they are in sync with the audio signals is described in clause 6.2.7. It operates differently for the JBM and non-JBM use. Fetching the correct spatial metadata values is not discussed in the following, it is assumed that it has already been correctly performed, as described in the aforementioned clause.

As an input, the renderer obtains (or receives) audio signals. First, the separately encoded audio signals are separated to a separate signal $s_{sep}(n,i)$, having $N_{sep} = 1 \ldots N_{obj}$ channels, depending on the number of objects and the OMASA coding mode. The MASA transport audio signals are separated to another signal $s_{MASA}(n,i)$, having 2 channels.

In addition, the renderer obtains (or receives) spatial metadata associated with the MASA transport audio signals. The spatial metadata obtained (or received) by the renderer contains the following parameters: azimuth $\theta(k,m,d)$, elevation $\phi(k,m,d)$, direct-to-total energy ratio $r_{dir}(k,m,d)$, spread coherence $\zeta(k,m,d)$, and surround coherence $\gamma(k,m)$.

In addition, the renderer obtains (or receives) object metadata containing azimuth $\theta_{obj}(m,i)$ and elevation $\phi_{obj}(m,i)$ angles associated with each separated object.

First, the MASA transport signals $s_{MASA}(n,i)$ are rendered to multi-channel or Ambisonic output signals $s_{MASArend}(n,j)$ using the methods described in clause 6.9.7.3.2.

Then, the separated signals $s_{sep}(n,i)$ are delayed by 5 ms to match the delay caused by the CLDFB processing of the MASA part rendering. The resulting signals are $s_{del}(n,i)$.

Then, the panning gains $g(m,i,j)$ are determined based on the azimuth $\theta_{obj}(m,i)$ and the elevation $\phi_{obj}(m,i)$ angles for each separate object i and output channel j. If the output is multi-channel loudspeakers, the panning gains are calculated using VBAP as presented in clause 7.2.1.2.2 (where the VBAP is initialized before the gain determination using the methods described in clause 7.2.1.2.1). If the output is Ambisonics, the panning gains are calculated by determining the spherical harmonic gain vectors $\mathbf{g}_{SH,\theta,\varphi}(m,i)$, which correspond to the azimuth $\theta_{obj}(m,i)$ and

elevation $\phi_{obj}(m,i)$ angles, using the ACN channel order (see clause 6.4.6.5.1) and the SN3D normalization, and setting the determined spherical harmonic gain vectors as the panning gains ($g(m,i,j) = \mathbf{g}_{SH,\theta,\varphi}(m,i)$).

Then, an interpolator is determined

$$g_{interp}(n) = \frac{n}{L_{subframe}}$$

Using these gains, the rendered signals are created by panning the object signals by

$$s_{seprend}(m;n,j) = \sum_{i=0}^{N_{sep}-1} \left(g_{interp}(n)g(m,i,j) + \left(1 - (g_{interp}(n)\right)g(m-1,i,j)\right)s_{del}(m;n,i)$$

where $g(m-1,i,j)$ refers to the last subframe of the previous frame when $m = 0$. The rendering is performed for all subframes m.

Then, the rendered output signals from the two renderers are combined

$$s_{out}(n,i) = s_{MASArend}(n,i) + s_{seprend}(n,i)$$

forming the output signals of the rendering.

6.9.7.3.2 Rendering of the MASA part

First, the direct and diffuse power factors and surround coherence ratios are computed based on the direct-to-total energy ratio $r_{dir}(k,m,i)$ and the surround coherence $\gamma(k,m)$, using the methods described in clause 6.5.7.2.2.

Then, directional responses are computed based on the MASA spatial metadata using the methods described in clause 6.5.7.2.3.

Then, diffuse responses are computed using the methods described in clause 6.5.7.2.4.

Then, the transport audio signals are transformed to the time-frequency domain with a 60-bin (with the sampling rate of 48 kHz) complex low-delay filter-bank (CLDFB) (see clause 6.2.5 for details), resulting in $S(k,n,i)$, where k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index.

Then, prototype audio signals (direct and diffuse prototype audio signals) are determined based on the transport audio signals) using the methods presented in clauses 6.5.7.2.5 and 6.5.7.2.6.

Then, the determined diffuse prototype audio signals are decorrelated using the methods presented in clause 6.5.7.2.7.

Then, the spatial audio signals are synthesized using the methods presented in clause 6.5.7.2.8, yielding $S_{out}(k,n,j)$.

Finally, the time-frequency domain signals are converted to the time domain via the inverse CLDFB (see clause 6.2.5 for details), yielding $s_{MASArend}(n,j)$, which are the synthesized time domain multi-channel loudspeaker or Ambisonic signals associated with the MASA part.

6.9.7.4 Mono rendering

The transport audio signals and the separately coded audio signals are first summed together. Then, the energy of the summed signal and the energies of the transport audio signals and the separately coded audio signals are computed. A gain is then applied to the summed signal such that the energy of the summed signal equals the sum of the energies of the transport audio signals and the separately coded audio signals. The resulting signal is then outputted as the mono audio signal.

6.9.8 OMASA PLC

In case of a packet loss, the MASA metadata and the OMASA metadata from the previous frame are used. ISM metadata and audio signal packet loss concealment follow the corresponding procedures for these operations.

6.9.9 OMASA bitrate switching

OMASA format supports bitrate switching between any two IVAS bitrates from the range 13.2 kbps to 512 kbps. At the beginning of decoding of every frame, it is verified whether the current IVAS total bitrate R_{IVAS} equals to that of the previous frame $R_{IVAS}^{[-1]}$. If this is not the case, the number of coded objects is read from the bitstream, OMASA mode selection is triggered according to the Table 5.9-1 and the following steps are performed:

- Setting of the current OMASA coding mode and number of the transport channels.
- Resetting of the ISM total bitrate and MASA total bitrate to their initial (nominal) values according to the values from Table 5.9-1.
- Reconfiguration of SCE core-decoders related to ISM coder part. If the number of ISM SCE core-decoders in the current frame is higher than the number of ISM SCE core-decoders in the previous frame, new memory handle instances are allocated and initialized. On the other hand, if the number of ISM SCE core-decoders in the current frame is lower than the number of ISM SCE core-decoders in the previous frame, obsolete memory handle instances are deallocated.
- Reconfiguration of MASA decoder.
- Reconfiguration of ISM decoder.
- Allocation/deallocation of HP20 filter memory handles to match the current number of transport channels. If new HP20 memory handles are allocated, their memories are also resetting to zeros.
- If the OMASA mode changes, selecting the rendering tool, see clause 6.9.6. Depending on the output config, changing the renderer can result in reconfiguration of HRTF data or the respective reverb module. Also, a reconfiguration and allocation/deallocation of the CLDFB instances to match the current number of transport channels and TD decorrelator can happen.

6.9.10 OMASA decoding with Time Scale Modification (TSM)

OMASA decoding with time scale modification follows the procedures from clause 6.2.7 and clause 6.9.

6.9.11 OMASA decoding to original combined input format

6.9.11.1 Overview

In this mode, the IVAS codec outputs the audio data in the same format that was input to the encoder. For the OMASA case, this means that the input should be formed by the MASA audio transport channels, MASA metadata, audio objects audio content and the objects metadata. The following sub-clauses describe the obtention of the OMASA format data in the 4 coding modes of OMASA.

6.9.11.2 Decoding to original combined input format in pre-rendering mode

In the pre-rendering coding mode of OMASA, the data is encoded and signalled in MASA format. Consequently, after reading the coding format as MASA format, additional information is read to determine if the original input format was OMASA. If that's the case, the number of objects in the input is also read, as described in clause 6.9.2. The output data is formed by the decoded MASA format data together with the corresponding number of null objects. The null objects have zero valued audio content and zero valued metadata.

6.9.11.3 Decoding to original combined input format in one object with MASA representation mode

According to what has been described in clause 6.9.3, for the one object with MASA representation coding mode, the information pertaining to the combined spatial audio signal in MASA format and the audio objects is obtained by decoding, for each frame, the encoded spatial audio signal to produce audio metadata in the MASA format and audio channel signals, the information used for decoding the separated object audio content and metadata, and the information related to the number of input audio objects to produce the same number of audio objects in the output data.

The number of objects is read on the two last bits of the bitstream. A corresponding number of null objects is generated following the steps as described further below.

The combined spatial audio signal represented in MASA format is formed of the transport audio signals s_{MASA} and the corresponding MASA metadata (azimuth θ_{MASA}, elevation ϕ_{MASA}, direct-to-total energy ratio $r_{MASA,dir}$, spread coherence ζ_{MASA}, surround coherence γ_{MASA}, and diffuse-to-total energy ratio $r_{MASA,diff}$).

The encoded separate audio object is decoded to give a separate audio object, containing the separately coded object audio signal $s_{sep}(n_{td})$ and the corresponding ISM direction (azimuth $\theta_{ISM}(m)$ and $\phi_{ISM}(m)$) (see clause 6.9.3).

The following paragraphs describe how the original input format of MASA format spatial audio format with N_{obj} audio objects are obtained from the decoded separated audio object and from the decoded combined audio signal in MASA format.

First, the MASA transport audio signals $s_{MASA}(n_{td}, i)$ are transformed to the time-frequency domain with a 60-bin (with the sampling rate of 48 kHz) complex low-delay filter-bank (CLDFB) (see clause 6.2.5 for details), resulting in $S_{MASA}(k, n, i)$, where n_{td} is the time-domain signal sample index, k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index. The audio signal of the separately coded object $s_{sep}(n_{td})$ is also transformed to the time-frequency domain with the same filter bank, resulting in $S_{sep}(k, n)$.

Then, energies in MASA frequency bands b and subframes m are determined for the MASA transport audio signals and the separate signal by

$$E_{MASA}(b,m) = \sum_{i=1}^{2}\left(\sum_{k=k_{\mathrm{low}}(b)}^{k_{\mathrm{high}}(b)}\left(\sum_{n=n_{\mathrm{first}}(m)}^{n_{\mathrm{last}}(m)}|S_{MASA}(k,n,i)|^2\right)\right)$$

$$E_{sep}(b,m) = \sum_{k=k_{\mathrm{low}}(b)}^{k_{\mathrm{high}}(b)}\left(\sum_{n=n_{\mathrm{first}}(m)}^{n_{\mathrm{last}}(m)}\left|S_{sep}(k,n)\right|^2\right)$$

where $k_{\mathrm{low}}(b)$ and $k_{\mathrm{high}}(b)$ are the first and the last bin of the frequency band b, and $n_{\mathrm{first}}(m)$ and $n_{\mathrm{last}}(m)$ are the first and the last temporal slot of the subframe m.

Then, the decoded separate audio object is converted to an object-based MASA stream, i.e., to an object-based MASA spatial metadata and to an object-based MASA transport audio signal. First, the object-based MASA spatial metadata parameters are determined as follows. The decoded ISM direction is set to the MASA direction (azimuth and elevation) of the created stream

$$\theta_{MASAsep}(b,m) = \theta_{ISM}(m)$$

$$\phi_{MASAsep}(b,m) = \phi_{ISM}(m)$$

The direct-to-total energy ratio is set to 1

$$r_{MASAsep,dir}(b,m) = 1$$

The spread coherence, the surround coherence, and the diffuse-to-total energy ratio are set to 0

$$\zeta_{MASAsep}(b,m) = 0$$

$$\gamma_{MASAsep}(b,m) = 0$$

$$r_{MASAsep,diff}(b,m) = 0$$

Then, object-based MASA transport audio signals are determined. Stereo panning gains $g(m, i)$ are determined based on the decoded ISM direction ($\theta_{ISM}(m)$, $\phi_{ISM}(m)$) using the methods described in clause 7.2.2.3.6 (for the “stereo” mode operation).

Then, an interpolator is determined

$$g'_{interp}(n_{td}) = \frac{n_{td}}{L_{subframe}}$$

where $L_{subframe}$ is the length of the subframe in samples, and n_{td} takes the values corresponding to a single subframe. The interpolator for a single subframe $g'_{interp}(n_{td})$ is concatenated four times to get the interpolator for the whole frame $g_{interp}(n_{td})$. Then, interpolated stereo panning gains are determined by

$$g(n_{td}, i) = g_{interp}(n_{td})g(m, i) + \left(1 - g_{interp}(n_{td})\right) g(m-1, i)$$

Using the determined interpolated stereo panning gains, the object-based MASA transport audio signals are determined

$$s_{MASAsep}(n_{td}, i) = g(n_{td}, i)s_{sep}(n_{td})$$

Then, the decoded MASA metadata ($\theta_{MASA}(b,m)$, $\phi_{MASA}(b,m)$, $r_{MASA,dir}(b,m)$, $\zeta_{MASA}(b,m)$, $\gamma_{MASA}(b,m)$, $r_{MASA,diff}(b,m)$) (see clause 6.9.3) and the determined object-based MASA metadata ($\theta_{MASAsep}(b,m)$, $\phi_{MASAsep}(b,m)$, $r_{MASAsep,dir}(b,m)$, $\zeta_{MASAsep}(b,m)$, $\gamma_{MASAsep}(b,m)$, $r_{MASAsep,diff}(b,m)$) are combined using the methods described in clause 5.9.3.2 using the determined energies $E_{MASA}(b,m)$ and $E_{sep}(b,m)$. The result is the combined MASA metadata ($\theta_{MASAcomb}(b,m)$, $\phi_{MASAcomb}(b,m)$, $r_{MASAcomb,dir}(b,m)$, $\zeta_{MASAcomb}(b,m)$, $\gamma_{MASAcomb}(b,m)$, $r_{MASAcomb,diff}(b,m)$), which is set to the output.

Then, the decoded MASA transport audio signals (see clause 6.9.3) and the determined object-based MASA transport audio signals are combined

$$s_{MASAcomb}(n_{td}, i) = s_{MASA}(n_{td}, i) + s_{MASAsep}(n_{td}, i)$$

The resulting combined MASA transport audio signals are set to the output.

Then, a number of null audio objects are generated as a substitute for the plurality of audio object inputted in the encoder. The number of the null audio objects corresponds to the number of objects N_{obj} read from the end of the current frame bitstream. The null audio objects contain an audio object channel signal having zero sample values and an audio object direction which is a predetermined fixed value.

The audio channel signal having zero sample values is determined by

$$s_{obj}(n_{td}, j) = 0, \;\; 1 \le j \le N_{obj}$$

The determined null object audio signals $s_{obj}(n_{td}, j)$ for each object j are set to the output.

Then, the corresponding object metadata is determined by setting the azimuth θ_{ISM} and the elevation angles ϕ_{ISM} to zero

$$\theta_{ISM}(i) = 0$$

$$\phi_{ISM}(i) = 0$$

The resulting object metadata is set to the output.

6.9.11.4 Decoding to original combined input format in parametric one object mode

According to what has been described in clause 6.9.4, for the one object with parametric representation coding mode, the information pertaining to the combined spatial audio signal in MASA format and the audio objects is obtained by decoding, for each frame, the information corresponding to the audio content of one separated audio object, the information corresponding to the MASA format representation of the initial MASA format audio signal and the rest of the audio objects, the MASA spatial metadata, the information to produce the metadata for the set of audio objects, and the information related to the audio object energy ratios and audio signal energy ratios which are the parameters defining the mix between the MASA format data and audio object data.

The number of objects is read on the two last bits of the bitstream. If there were more than one object, two additional bits are read signalling the index of the separated object. The importance of the separated object is read on two bits and serves at the bitrate adaptation according to clause 5.9.8.

The combined spatial audio signal represented in MASA format is formed of the transport audio signals s_{MASA} and the corresponding MASA metadata. The following paragraphs describe how the original input format of MASA format spatial audio format with N_{obj} audio objects are obtained from the decoded separated audio object and the decoded combined audio signal in MASA format and from the parametric signal mix information decoded according to the methods described in 6.9.4.2 and 6.9.4.3.

First, the object audio signals $s_{obj}(n_{td}, j)$ are generated for each audio object j from the transport audio signals $s_{MASA}(n_{td}, i)$, ISM ratios, the MASA-to-total energy ratios, and the object directions. As a first step, the transport audio signals $s_{MASA}(n_{td}, i)$ are transformed to the time-frequency domain with a 60-bin (with the sampling rate of 48 kHz) complex low-delay filter-bank (CLDFB) (see clause 6.2.5 for details), resulting in $S_{MASA}(k, n, i)$, where n_{td} is the time-domain signal sample index, k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index.

Then, stereo panning gains $g(i, j)$ are determined for each object j based on the decoded ISM direction ($\theta_{ISM}(j)$, $\phi_{ISM}(j)$) (see clause 6.9.4) using the methods described in clause 7.2.2.3.6 (for the "stereo" mode operation). An interpolator is determined

$$g_{interp}(n) = \frac{n}{L_{slots}}$$

where L_{slots} is the number of CLDFB slots per frame. Interpolated stereo panning gains are determined by

$$g(n, i, j) = g_{interp}(n) g(i, j) + \left(1 - g_{interp}(n)\right) g^{[-1]}(i, j)$$

where $g^{[-1]}(i, j)$ are the stereo panning gains determined for the previous frame.

Then, prototype object audio signals are determined for each object j by

$$S_{prot}(k, n, j) = \sum_{i=1}^{2} \left(g(n, i, j) S_{MASA}(k, n, i) \right)$$

Then, the energies of the prototype object audio signals and the MASA transport audio signals are determined

$$E_{prot}(k, n, j) = \left| S_{prot}(k, n, j) \right|^2$$

$$E_{MASA}(k, n) = \sum_{i=1}^{2} \left(|S_{MASA}(k, n, i)|^2 \right)$$

Then, the target energies for the object audio signals are determined

$$E_{obj,target}(k, n, j) = r_{ISM,dir}(k, m, j) E_{MASA}(k, n)$$

where $r_{ISM,dir}(k, m, j)$ is the rendering direct-to-total energy ratio for the object j (determined in clause 6.9.6 based on the ISM ratios and the MASA-to-total energy ratios), and m is the subframe index.

The energies are smoothed over time by

$$E_{prot,sm}(k, n, j) = 0.05 E_{prot}(k, n, j) + 0.95 E_{prot,sm}(k, n-1, j)$$

$$E_{obj,target,sm}(k, n, j) = 0.05 E_{obj,target}(k, n, j) + 0.95 E_{obj,target,sm}(k, n-1, j)$$

Then, the object processing gains are determined by

$$g_{obj}(k, n, j) = \min\left(4, \sqrt{\frac{E_{obj,target,sm}(k, n, j)}{E_{prot,sm}(k, n, j)}}\right)$$

and using them the rendered object audio signals are determined by

$$S_{obj,rend}(k, n, j) = g_{obj}(k, n, j) S_{prot}(k, n, j)$$

The time-frequency domain rendered object audio signals $S_{obj,rend}(k, n, j)$ are converted to the time domain via the inverse CLDFB (see clause 6.2.5 for details), yielding the rendered object audio signals $s_{obj,rend}(n_{td}, j)$ in the time domain.

The encoded separate audio object is decoded to give a separate audio object, containing a separately coded object audio signal $s_{sep}(n_{td})$. The decoded separately coded object audio signal $s_{sep}(n_{td})$ is delayed by 5 milliseconds to

match the delay caused by the CLDFB processing, yielding $s_{sep,del}(n_{td})$. Then, the delayed separately coded audio object signal $s_{sep,del}(n_{td})$ and the rendered object audio signals $s_{obj,rend}(n_{td}, j)$ are combined

$$s_{obj}(n_{td}, j) = s_{obj,rend}(n_{td}, j) + s_{sep,del}(n_{td}), \quad if\ j = j_{sep}$$

$$s_{obj}(n_{td}, j) = s_{obj,rend}(n_{td}, j), \quad if\ j \neq j_{sep}$$

where j_{sep} is the object index of the separately coded object audio signal (see clause 6.9.4.1).

The audio objects are created by assigning a corresponding directional value for each audio object signal $s_{obj}(n_{td}, j)$. The directional values (azimuths and elevations for each object j) are obtained from the decoded object directions determined in clause 6.9.4.4. The determined audio objects (containing the object audio signals and the corresponding directions) are set to the output.

Then, MASA transport audio signals associated with the MASA transport audio signals inputted in the encoder are separated from the MASA transport audio signals obtained in the decoder (also containing the object audio signal content) by processing them based on the MASA-to-total energy ratios.

The target energies for the MASA transport audio signals are determined

$$E_{MASA,target}(k, n) = r_{MASA2tot}(k, m) E_{MASA}(k, n)$$

where $r_{MASA2tot}(k, m)$ is the MASA-to-total energy ratio (determined in clause 6.9.4.2).

The energies are smoothed over time by

$$E_{MASA,sm}(k, n) = 0.05 E_{MASA}(k, n) + 0.95 E_{MASA,sm}(k, n-1)$$

$$E_{MASA,target,sm}(k, n) = 0.05 E_{MASA,target}(k, n) + 0.95 E_{MASA,target,sm}(k, n-1)$$

MASA processing gains are determined by

$$g_{MASA}(k, n) = \min\left(4, \sqrt{\frac{E_{MASA,target,sm}(k, n)}{E_{MASA,sm}(k, n)}}\right)$$

and using them the MASA transport audio signals are processed by

$$S_{MASAproc}(k, n, i) = g_{MASA}(k, n) S_{MASA}(k, n, i)$$

The determined processed MASA transport audio signals $S_{MASAproc}(k, n, i)$ are set to the output. Alongside the determined MASA transport audio signals, the decoded MASA spatial metadata (determined in clause 6.9.4.1) is set to the output.

6.9.11.5 Decoding to original combined input format in discrete mode

The decoding to original combined input format in the discrete mode is straightforward. After reading the number of objects from the end of the bitstream, and the format bitrate distribution based on object importance as described in clause 5.9.8, each of the two input formats are decoded separately and output independently.

7 Functional description of the rendering, rendering control, and pre-rendering

7.1 Rendering overview

The general rendering overview is provided in clause 4.5. This clause discusses on rendering modes for loudspeakers and headphones reproduction (clause 7.2), room acoustics rendering (clause 7.3), rendering control (clause 7.4), and pre-rendering (clause 7.5).

7.2 Rendering modes

7.2.1 Rendering for loudspeaker reproduction

7.2.1.1 Overview

Digital audio decoded by IVAS decoder can be rendered for loudspeaker reproduction. The process of rendering depends on the decoded audio format. In case of multi-channel formats, the decoded format can match the loudspeaker configuration or output channels can be generated by application of multi-channel conversion gains from conversion tables. For SBA, MASA and ISM formats the spatial audio needs to be mapped to the loudspeaker positions of the loudspeaker setup (e.g., 5.1 or 7.1.4). Depending on the decoded, format amplitude panning is employed, with either the vector-base amplitude panning (VBAP) scheme (using triangles), described in detail in clause 7.2.1.2, or an improved edge-fading amplitude panning (EFAP) scheme (using polygons), described in detail in clause 7.2.1.3. Specifically for SBA audio, the AllRAD loudspeaker decoding scheme (clause 7.2.1.4) is used with EFAP.

7.2.1.2 Vector-base amplitude panning (VBAP)

7.2.1.2.1 VBAP initialization

7.2.1.2.1.1 VBAP initialization overview

The input to the processing is the loudspeaker positions as azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles, and the number of loudspeakers N.

First, it is checked if there is a need for virtual loudspeakers, using the method presented in clause 7.2.1.2.1.2. The check is performed for the need of the virtual bottom, top, and back loudspeakers.

Based on the checks, a speaker node set $\theta(i), \phi(i)$ is determined. It contains the input loudspeaker nodes $\theta(i) = \theta_{LS}(i), \phi(i) = \phi_{LS}(i)$. In addition, it contains the virtual loudspeakers, if it was determined that there is a need for them. The locations of the virtual loudspeakers (if they are needed) are: bottom: $\theta = 0, \phi = -90$, top: $\theta = 0, \phi = 90$, and back: $\theta = 180, \phi = 0$. Thus, the resulting speaker node set $\theta(i), \phi(i)$ contains within a 3D space $N_{node} = N +$ 0 ... 3 speaker nodes depending on the need of the virtual loudspeakers.

Furthermore, for each virtual loudspeaker, the virtual loudspeaker type is determined by the method in clause 7.2.1.2.1.2. The types are: DISTRIBUTE, DISCARD.

Then, the speaker node data is initialized, using the method presented in clause 7.2.1.2.1.3. Furthermore, as an outcome, a group is obtained for each speaker node (HORIZONTAL, BOTTOM_HALF, TOP_HALF) or alternatively all nodes are assigned with group ALL, depending on the loudspeaker node configuration.

Then, a set of non-crossing planes is determined, using the method presented in clause 7.2.1.2.1.4.

Then, the connections between the speaker nodes are determined, using the method presented in clause 7.2.1.2.1.5. In this clause, the determination of the connections depends on the group. If the nodes were determined to be in the ALL group, the speaker nodes are grouped to a single group. If the speaker nodes were determined to not be in the ALL group, the speaker nodes are grouped to bottom half speaker nodes, top half speaker nodes, and horizontal nodes, and the determination of the node connections is performed in these groups.

Next, the rule to distribute virtual speaker node gains to the actual speaker nodes is presented in clause 7.2.1.2.1.8.

Then, the virtual surface triplets are determined based on the determined connections as presented in clause 7.2.1.2.1.9. The outcome of the initialization is one or two search structures, each containing the virtual surface triplets (and their speaker nodes and inverse matrices), and initial search indices for fast searching and panning.

7.2.1.2.1.2 Checking for the need of virtual loudspeakers

The basic operating principle of vector base amplitude panning (VBAP) is to formulate, for any panning azimuth and elevation, three non-negative amplitude panning gains. For a given direction, a base of three unit vectors pointing towards a subset of three loudspeakers is selected, and gains are formulated for these vectors so that the weighted sum of them points towards the panning target. For such non-negative gains to be formulable, there are requirements for the loudspeaker arrangement. For example, if there are loudspeakers only at the front half sphere, the panning gains cannot

be determined for rear directions with such methodology. Similarly, if there are only horizontal and elevated loudspeakers, non-negative panning gains cannot be determined for negative elevations.

Therefore, the present VBAP algorithm performs checks to add virtual loudspeakers to the loudspeaker arrangement. In specific, it is checked if any of a virtual bottom, virtual top, or virtual back loudspeaker needs to be added. If any of these virtual loudspeakers are added to the loudspeaker configuration, they are considered at the VBAP triangulation and triplet gain formulation as regular loudspeakers. The triangulation refers to determining a non-overlapping virtual surface arrangement covering all 3D sphere of directions. Each of the virtual surfaces in the arrangement is triangle-shaped and has corners positioned at three different speaker nodes.

When determining the final panning gains for actual loudspeakers only, any positive gains assigned to the virtual loudspeakers will be discarded or distributed to adjacent actual loudspeakers at the setup. The rule to decide discarding or distributing is detailed in clause 7.2.1.2.1.2.

First, it is checked if a virtual bottom and top directions are needed. This is checked by first finding the maximum and minimum elevation value from the existing loudspeaker setup. If the minimum elevation value of the existing loudspeaker setup is $\phi_{min} \geq -45°$, then a bottom (-90 degrees elevation) virtual loudspeaker is added to the loudspeaker setup. If the maximum elevation value of the existing loudspeaker setup is $\phi_{max} \leq 45°$, then a top (90 degrees elevation) virtual loudspeaker is added.

Next it is checked if a virtual back direction is needed. It is checked if there is at least one loudspeaker position of the existing loudspeaker setup with $|\phi(i)| < 45°$ and $|\theta(i)| > 91°$, when azimuth values are considered in range -180 … 180 degrees. If such a loudspeaker is not found, then an additional virtual loudspeaker with an orientation of directly behind (180 degrees azimuth, 0 degrees elevation) is added to the loudspeaker setup.

Since the order of the added virtual loudspeakers matters in clause 7.2.1.2.1.8, the order of adding them is defined here. If any of the virtual loudspeakers are added, they are added in the following order: bottom, top and back. They are added to the end of the loudspeaker setup arrangement. In other words, the indices of the actual loudspeakers in the loudspeaker setup do not change.

Furthermore, in case a virtual loudspeaker has been added at bottom and/or at top, then a further check is performed: For the top, if $\phi_{max} \geq 20°$ (and correspondingly for the bottom $\phi_{min} \leq -20°$), then a flag DISTRIBUTE is set for that virtual loudspeaker. Otherwise, a flag DISCARD is set for that virtual loudspeaker. In the case of a back virtual loudspeaker being added to the loudspeaker setup, then a flag DISTRIBUTE is set for that virtual loudspeaker.

In the following clauses, the added virtual bottom and/or top and/or rear loudspeakers are considered as actual loudspeaker nodes (loudspeakers) at these positions, unless specifically mentioned otherwise.

7.2.1.2.1.3 Speaker node data initialization

The loudspeaker positions (including any virtual loudspeakers) can be referred to as “speaker nodes” or simply “nodes” going forward.

The nodes are defined by their azimuth and elevation. In this clause, pre-processing operations to the node data is described. First, any nodes that have the elevation at range $-5° \leq \phi(i) \leq 5°$ is set to have $\phi(i) = 0°$

Next, the largest azimuth angle gap of the nodes with $\phi(i) = 0°$ is determined. This is formulated by finding the azimuth angle difference of each of the nodes at the elevation of $\phi(i) = 0°$ to the next node at the elevation of $\phi(i) = 0°$, and finding the largest of these differences. The angle difference is formulated for the shortest path, for example, nodes with azimuths $179°$ and $-179°$ would have a $2°$ azimuth angle difference.

If the largest gap for the horizontal nodes is 170 degrees or less, then the following steps are applied: Each horizontal node is associated with a label HORIZONTAL; Each node with positive elevation is associated with a label TOP_HALF; Each node with negative elevation is associated with a label BOTTOM_HALF. In other words, when the condition is met, the speaker node set is divided into top and bottom parts by a defined virtual plane that is the horizontal plane in the 3D space, and an additional part having nodes on the horizontal plane.

If the above condition is not met, all nodes are associated with a label ALL.

7.2.1.2.1.4 Determining non-crossing planes

For the given set of loudspeaker nodes, a set of non-crossing planes are defined. All unique elevations $\phi(i)$ of the loudspeaker nodes that do not have the label HORIZONTAL are checked. For each of these elevations, the following steps are performed: Find the largest gap of the nodes having the same elevation value (the operation of finding the

largest gap is the same as in clause 7.2.1.2.1.3 for horizontal nodes); If the largest gap is equal or smaller than 140 degrees, then the corresponding elevation ϕ is set as a non-crossing plane. The resulting set of non-crossing planes are denoted as $\phi_{nc}(p)$ where p is the index of the non-crossing plane. The number of non-crossing planes can be zero or higher, depending on the loudspeaker arrangement. The non-crossing plane information is used in clause 7.2.1.2.1.7, and its purpose is to prefer triangulation where the triangle edges are along these non-crossing planes, instead of crossing them.

7.2.1.2.1.5 Determining speaker node connections

The term connection refers to a pair of nodes (a, b) where a and b can refer to any node index i for $a \neq b$. When a set of connections is determined, the resulting set are the VBAP triangulation edges. In other words, determining a suitable set of connections therefore determines a non-overlapping arrangement of triangle-shaped virtual surfaces encompassing the 3D directions. These connections are the virtual surface sides or edges, and the nodes are the virtual surface corners.

There are two methods the VBAP can determine the set of connections for the given loudspeaker node arrangements (where some of the nodes may be virtual).

The first initialization mode is set if the speaker nodes were associated with label ALL. In this mode, all loudspeaker nodes are considered in one group. The speaker node connections are determined using the 3D method as described in clause 7.2.1.2.1.7.

Otherwise, the second initialization mode is set, where the speaker node connections are determined in the following steps: First, top half connections are determined using the 3D method as described in clause 7.2.1.2.1.7; Next, horizontal connections are determined using a horizontal method as described in clause 7.2.1.2.1.6; Next, the bottom half connections are determined using the 3D method as described in clause 7.2.1.2.1.7. After determining all connections (which are the virtual surface edges) for these three parts (top, horizontal, bottom), the resulting set of connections are combined as one unified set. These connections then are the edges of the triangle-shaped virtual surfaces within the non-overlapping virtual surface arrangement encompassing the 3D directions. When the connections are defined in three sets (top, horizontal, bottom), the resulting triangulation thus is such that the edges of the triplet surfaces do not intersect the horizontal plane (and thus also the triplet surfaces themselves do not intersect the horizontal plane.

7.2.1.2.1.6 Determining speaker node connections for horizontal node groups

The horizontal connections are determined by finding for each node a that has label HORIZONTAL the next node b that has a label HORIZONTAL. The “next node” here refers to the finding the node b (that has a label HORIZONTAL) which is most adjacent to the node a in positive azimuth direction. The adjacency accounts for any wrapping of the angle, so that if the wrapping point is at $\theta = 180°$, then for example for a node at $\theta(a) = 170°$, a further node at $-170°$ may be considered adjacent (unless there is further node in between) and $20°$ degrees at the positive azimuth direction from $170°$.

The result of this operation is a set of as many virtual connections (a, b) on the horizontal plane as there are nodes with label HORIZONTAL.

7.2.1.2.1.7 Determining speaker node connections for 3D node groups

In this clause, it is defined how to define a set of connections for a node group which includes at least one non-horizontal node. The operations in this clause can be called in different ways: for entire node set, for top node set, or for bottom node set.

First, if the processing is for the top node set, all nodes having a label BOTTOM_HALF are discarded from the following operations. And, if the processing is for the bottom node set, all nodes having a label TOP_HALF are discarded from the following operations.

Next, all the pairs (a, b) where $a \neq b$ of the remaining nodes are defined as potential connections.

Next, all connections where both nodes have the label HORIZONTAL are discarded.

Next is to check if any of the connections passes the origin closely. Denoting $\mathbf{u}_a$ and $\mathbf{u}_b$ as column unit vectors pointing towards nodes a and b, this is checked by finding if $|\mathbf{u}_a^T \mathbf{u}_b + 1| < 0.001$. If this condition is met, the connection is discarded. The unit vectors are vectors from the origin to speaker node positions in 3D space where the node distance from origin (vector magnitude) is assumed to be 1 unit.

Next is to check if there is a node behind any of the connections. This is checked for all remaining connections (a, b) by assuming a straight line joining the unit vectors $\mathbf{u}_a$ and $\mathbf{u}_b$, and then checking all unit vectors $\mathbf{u}_c$, where c is a further node index so that $c \neq a \neq b$, by the following step: For each $\mathbf{u}_c$, a unit vector $\mathbf{u}_{ab}$ is defined that has a minimal spatial angle $\cos^{-1}(\mathbf{u}_{ab}\mathbf{u}_c)$ when the condition is met that the unit vector $\mathbf{u}_{ab}$ intersects the line joining $\mathbf{u}_a$ and $\mathbf{u}_b$. If the minimal spatial angle is less than 1 degree, the connection between a and b is discarded.

All remaining connections (a, b) are then individually checked for the condition $|\mathbf{u}_a - \mathbf{u}_b| > 5|\mathbf{u}_c - \mathbf{u}_{ab}|$ for any $c \neq a \neq b$. If this condition is met, the connection is assigned with a label THIN_TRIANGLE_CONNECTION.

For the connections (a, b) that have not been discarded due to the above steps, an arc value is determined by $\alpha_{ab} = \cos^{-1}(\mathbf{u}_a^T\mathbf{u}_b)$ and a weighted arc $\hat{\alpha}_{ab}$ is determined for the connection as $\hat{\alpha}_{ab} = \alpha_{ab,mod1}\alpha_{ab,mod2}\alpha_{ab}$. The first modifier $\alpha_{ab,mod1}$ is 1, except if the connection has been assigned with the label THIN_TRIANGLE_CONNECTION then in this case the first modifier $\alpha_{ab,mod1} = 2$. The second modifier $\alpha_{ab,mod2}$ is 1, except if the elevation crosses a non-crossing plane (defined in clause 7.2.1.2.1.4), i.e., for $\phi(a)$ and $\phi(b)$ one of these elevation values is at least 1 degree larger than $\phi_{nc}(p)$ and the another is at least 1 degree smaller than $\phi_{nc}(p)$, for any p. If this condition is met, then $\alpha_{ab,mod2} = 2$.

Next, the connections (a, b) are sorted with respect to their respective weighted arc $\hat{\alpha}_{ab}$, in increasing order from smallest to largest.

Then, an empty set of valid connections is defined. From first to last, all sorted connections (a, b) are processed with the following steps.

For each connection (a, b), it is checked that if within the currently existing set of valid connections there are any connection that meets each of these criteria: The existing valid connection has nodes that are different from both a and b; From the viewpoint of the origin, a line between node pair (a, b) crosses a line between the nodes of any of the connections at the set of valid connections. If these conditions are met, the new connection (a, b) is not valid and is discarded. Otherwise, the new connection (a, b) is valid and is added to the set of valid connections.

In the above, the crossing of two different connections (a_1, b_1) and (a_2, b_2) for $a_1 \neq b_1 \neq a_2 \neq b_2$ may be formulated by defining two planes: First plane is defined by origin and points defined by the unit vectors $\mathbf{u}_{a_1}$ and $\mathbf{u}_{b_1}$, and second plane is defined by origin and points defined by the unit vectors $\mathbf{u}_{a_2}$ and $\mathbf{u}_{b_2}$. The intersection of these planes is formulated, and the result is a line in 3D space. If this line passes between $\mathbf{u}_{a_1}$ and $\mathbf{u}_{b_1}$, then a crossing is registered. If the planes are the same, then the intersection is not a line but a plane, and in this case the crossing is not registered.

When all sorted connections (a, b) are processed as in the foregoing, the resulting set of valid connections is the determined set of node connections in the 3D method.

7.2.1.2.1.8 Determining virtual loudspeaker distribution gains

For each virtual loudspeaker node (back, bottom, top) which has been added to the loudspeaker setup, a set of division gains are defined. This entails, defining for any of the back, bottom, top virtual loudspeakers a respective gain distribution column vector $\mathbf{g}_{\text{dist,back}}$, $\mathbf{g}_{\text{dist,bottom}}$, $\mathbf{g}_{\text{dist,top}}$. Each vector has as many elements as there are (non-virtual) loudspeakers in the loudspeaker arrangement. For virtual loudspeakers which have been labelled with a DISCARD flag, the elements of the corresponding gain distribution vector $\mathbf{g}_{\text{dist,x}}$ (where x is back, bottom or top) are set as zeros.

For virtual loudspeakers which have been labelled with a DISTRIBUTE flag, the following method for determining the gains is performed.

For each virtual loudspeaker node in turn, the connection set as determined according to clause 7.2.1.2.1.5 is searched for any connections (a, b) where either a or b is the virtual loudspeaker node in question. With this method, all loudspeaker nodes i can be found that are connected to the virtual loudspeaker node in question. The elements of the column gain vector $\mathbf{g}_{\text{dist,x}}$ are then set so that the value of rows which correspond to any index i that is connected to the virtual loudspeaker node is given the value $1/\text{N}_{dist,x}$ where $\text{N}_{dist,x}$ is the number of nodes connected to the virtual loudspeaker node in question, the other elements of $\mathbf{g}_{\text{dist,x}}$ are set to zero.

However for the case when the IVAS format is MASA_ISM_FORMAT, the non-zero gains are given the value $\left(\frac{1}{\text{N}_{dist,x}}\right)^{0.8}$.

Additionally, a converter matrix $\mathbf{C}$ is defined that maps the gains determined for a loudspeaker arrangement with potentially added virtual loudspeakers to a set of gains that are defined for only real loudspeakers. The matrix $\mathbf{C}$ is

defined as follows: First, an N times N identity matrix is defined which is then appended with an additional column on the right for each additional virtual loudspeaker. From clause 7.2.1.2.1.2, virtual loudspeakers are added in order (bottom, top, back), and if virtual loudspeakers are to be added to the matrix C then the matrix $\mathbf{C}$ is appended with columns in the order $\mathbf{g}_{\mathrm{dist,bottom}}$, $\mathbf{g}_{\mathrm{dist,top}}$, $\mathbf{g}_{\mathrm{dist,back}}$.

7.2.1.2.1.9 Determining virtual surface triplets

The virtual surface triplets are determined based on the connections as determined in clause 7.2.1.2.1.5. The connections are the triplet edges, and the triplets are found by finding all sets of three connections which share a set of three loudspeaker nodes. When the nodes are labelled ALL, then there is one set of virtual surface triplets determined that is the entire virtual surface arrangement. Otherwise, two sets of virtual surface triplets are defined: One has all the virtual surfaces where the nodes have labels TOP_HALF and HORIZONTAL. Another has all the virtual surfaces where the nodes have labels BOTTOM_HALF and HORIZONTAL. In other words, in that case there are two sets, divided by the horizontal plane.

Each triplet includes three (real or virtual) nodes (a, b, c) where $a \neq b \neq c$, and the set of (a, b, c) is different for each triplet. For each triplet an associated 3x3 inverse matrix is defined by the unit 3x1 vectors

$$\mathbf{R}^{-1} = [\mathbf{u}_a \quad \mathbf{u}_b \quad \mathbf{u}_c]^{-1}$$

The purpose of such a matrix is to find a 3x1 panning gain vector $\mathbf{g}_{\mathrm{tmp}}$ that satisfies the condition that $\mathbf{u}_{\mathrm{target}} = \mathbf{R}\mathbf{g}_{\mathrm{tmp}}$ where $\mathbf{u}_{\mathrm{target}}$ is a unit vector pointing towards the desired panning direction, where $\mathbf{g}_{\mathrm{tmp}}$ is a temporary triplet gain vector, not a final gain vector.

The following operations in this clause are performed independently for each of the determined virtual surface sets (which can be one set, or bottom and top halves separately).

The virtual surface triplets are arranged to an ordered virtual surface set based on the triplet azimuth angles $\theta_{triplet}(i_{tr})$ (i_{tr} is the reference index of the triplet) in an ascending order, where these triplet azimuth angles are determined by summing together the three unit vectors contributing to the triplet (i.e., $\mathbf{u}_{abc} = \mathbf{u}_a + \mathbf{u}_b + \mathbf{u}_c$), where (a, b, c) refer to the node indices within the triplet and are thus different for each triplet, and finding the azimuth angle by $\theta_{triplet}(i_{tr}) = atan2(u_{abc,y}, u_{abc,x})$, where $u_{abc,y/x}$ refer to the horizontal y/x entries of $\mathbf{u}_{abc}$.

Then, initial search indices are determined based on the sorted triplet azimuth angles $\theta_{triplet,sort}(i_{tr})$ (that correspond to the sorted triplets). Four search sectors are determined, where each sector occupies a range of azimuth angles defined by start and end azimuth angles (in degrees), which are determined for each sector $j = 0 \ldots 3$ by

$$\alpha_{start}(j) = 90j$$

$$\alpha_{end}(j) = 90(j + 1)$$

Then, the sector reference azimuth angles are determined by

$$\alpha_{ref}(j) = \frac{\alpha_{start}(j) + \alpha_{end}(j)}{2}$$

Then, for each of these sectors, one of the virtual surface triplets of the ordered set is associated to the search sector, by selecting the triplet having the azimuth angle $\theta_{triplet,sort}(i_{tr})$ the closest to the sector reference azimuth angle $\alpha_{ref}(j)$. This is set as the initial triplet for the search sector, i.e., $\xi(j) = i_{tr}$.

7.2.1.2.2 VBAP gain determination

7.2.1.2.2.1 VBAP gain determination overview

The input to the processing is a target panning direction comprising a target azimuth angle θ and target elevation angle ϕ.

First, the correct search structure is determined. If there are two search structures and if the elevation angle is larger than zero (i.e., $\phi > 0$), the second search structure is used (containing virtual surfaces associated with TOP_HALF and HORIZONTAL nodes). Otherwise, the first search structure is used (containing virtual surfaces associated with BOTTOM_HALF and HORIZONTAL, or ALL, nodes).

Then, the best virtual surface triplet from the determined virtual surface arrangement is selected and the corresponding panning gains are generated based on it using the method described in clause 7.2.1.2.2.2. In other words, triplet gains $\mathbf{g}_{\mathrm{tmp}}$ and the triplet to be used with respect to the present azimuth θ and elevation ϕ are formulated. The triplet node indices (a, b, c) for which the triplet gains $\mathbf{g}_{\mathrm{tmp}}$ are associated with are thus also determined.

The next step is to normalize the gains by

$$\mathbf{g}_{\mathrm{norm}} = \begin{bmatrix} g_a \\ g_b \\ g_c \end{bmatrix} = \frac{\mathbf{g}_{\mathrm{tmp}}}{\sqrt{\mathbf{g}_{\mathrm{tmp}}^{\mathrm{T}} \mathbf{g}_{\mathrm{tmp}}}}$$

Note that in the above, any of the indices (a, b, c) may correspond to real or virtual loudspeaker node. These normalized triplet gains $\mathbf{g}_{\mathrm{norm}}$ are converted to a panning gain vector $\mathbf{g}_{\mathrm{all}}$ that is a column vector that has N elements, i.e., as many elements as there are actual loudspeakers. This is achieved by first determining a $\mathbf{g}_{\mathrm{all,tmp}}$ which is a column vector with as many entries as there are in total real or virtual nodes, where each a:th, b:th and c:th entry is set to respectively have values g_a, g_b, g_c, and other values are zero. Next the converter matrix $\mathbf{C}$ defined in clause 7.2.1.2.1.8 is used to convert the gains to the final amplitude panning gains for the given loudspeaker setup by

$$\mathbf{g}_{\mathrm{all}} = \mathbf{C}\mathbf{g}_{\mathrm{all,tmp}}$$

These are the panning gains output by the VBAP algorithm. They are used for positioning sound in 3D space. For example, with the OMASA format, these gains are applied to an audio object signal, and the gained audio object signals are forwarded to corresponding loudspeaker channels, as a result causing that object audio signal to be positioned in the desired direction within the 3D space.

7.2.1.2.2.2 Determining the best virtual surface triplet and the panning gains

First, the correct search sector from the four search sectors is determined based on the target azimuth angle θ by

$$if\ 0 \leq \theta \leq 90 \rightarrow j = 0$$

$$if\ 90 < \theta \leq 180 \rightarrow j = 1$$

$$if\ -180 < \theta < -90 \rightarrow j = 2$$

$$if\ -90 \leq \theta < 0 \rightarrow j = 3$$

Then, the first virtual surface triplet is determined by $\xi(j)$, where $\xi(j)$ is the initial triplet for sector j determined in clause 7.2.1.2.1.9.

The next step is to iteratively find the correct virtual surface triplet (i.e., triangle). Starting from the virtual surface triplet associated with the correct search sector, the ordered virtual surface set is searched to determine a virtual surface that encloses the target panning direction. This is done by the following steps:

1. Set the initial search index as the current triplet index $i_{tr} = \xi(j)$.
2. Set triplet counter $m = 0$.
3. Check if the current triplet is the correct triplet (see details below).
4. If not, increase the counter m by one, and calculate new current triplet index, according to the equation $i_{tr} = \mathrm{mod}\left(\xi(j) + \mathrm{floor}\left(\frac{m+1}{2}\right)(-1)^m, N\right)$, where N is the number of triplets in the ordered virtual surface set, mod is a modulo function, and floor a flooring function. In practice, the function results in a pattern of $(i_{tr}, i_{tr} - 1, i_{tr} + 1, i_{tr} - 2, i_{tr} + 2, \ldots)$ until all triplets have been checked.
5. If yes, then the current triplet is the correct one, and the triplet index i_{tr} is output (alongside the triplet gains $\mathbf{g}_{\mathrm{tmp}}$).

In step 3, the checking if the current triplet is the correct triplet is performed as follows. As shown in clause 7.2.1.2.1.9, each triplet is associated with a preformulated inverse matrix $\mathbf{R}^{-1}$ that is unique to each triplet. Triplet panning gains are obtained by $\mathbf{g}_{\mathrm{tmp}} = \mathbf{R}^{-1}\mathbf{u}_{\mathrm{target}}$. If all gains are non-negative, the triplet is determined to be a correct triplet (i.e., the virtual surface triplet that encloses the target panning direction), and otherwise not. When the triplet is deemed correct,

the formulated $\mathbf{g}_{\mathrm{tmp}}$ are the triplet panning gains that are further processed in clause 7.2.1.2.2.1 to obtain the final gains for the loudspeakers.

7.2.1.3 Edge Fading Amplitude Panning (EFAP)

7.2.1.3.1 EFAP overview

The Edge Fading Amplitude Panning (EFAP) algorithm is used to compute panning gains to position a sound source at a requested point on a given set of loudspeaker nodes. These given set of loudspeaker nodes each have an associated output audio channel which can either be one of the supported output loudspeaker configurations in IVAS (cf. clause 4.3.2) or a user-specified custom loudspeaker layout (see clause 7.4.9). As mentioned in the related clauses, the output loudspeaker nodes are described by azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles, where i is the loudspeaker channel.

EFAP is used to determine panning gains which are used in the following rendering paths of the decoder:

- Parametric ISM rendering to loudspeaker or ambisonics (where 7.1+4/CICP19 is used as an intermediate layout) output.
- Discrete ISM rendering to loudspeaker
- Multichannel rendering to a user-specified custom loudspeaker layout.
- Rotating a multichannel scene on the same loudspeaker layout for combined head and/or orientation tracking for a subsequent binaural output.
- Rendering of virtual loudspeaker positions to a real loudspeaker layout in the ALLRAD (All-round ambisonic decoding) rendering of ambisonics to loudspeaker output (cf. clause 7.2.1.4).

The input audio channels are each associated with a panning position on the spherical surface specified by azimuth and elevation coordinates based on the input format. A unit radius is implicit in this setup.

Panning with EFAP consists of two steps; first determining the panning gains for the input audio channels and then applying these gains via multiplication to the input audio samples to produce the output audio samples. This operation can also be performed iteratively on all input channels as a matrix multiplication where the matrix is populated using EFAP for each of the individual input channels.

7.2.1.3.2 EFAP initialization

7.2.1.3.2.1 General

The EFAP module must be initialized for the desired set of output loudspeaker nodes. Thus, the positions of the loudspeakers as azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles and the total number of loudspeakers N must be provided. Additionally, the panning mode must also be specified, the two options are amplitude panning mode (referred to as Edge Fading Amplitude Panning) or intensity panning mode (referred to as Edge Fading *Intensity* Panning or EFIP). The difference between the modes is that for amplitude panning, the computed output loudspeaker gains sum to unity, whereas for intensity panning the sum of the squares of the gains is unity.

Given the necessary information, the first step of the initialization is to wrap the spherical coordinates of the input loudspeaker positions to uniform bounds; these are ±180° for azimuth and ±90° for elevation. The cartesian coordinates of the loudspeaker nodes are also computed. This data is stored in an internal vertex data structure in preparation for the construction of a 3D convex hull.

7.2.1.3.2.2 Addition of ghost loudspeakers

Once the output loudspeaker nodes are processed into the internal vertex data, checks are performed to evaluate the need for ghost loudspeakers. The addition of ghost loudspeakers ensure that the set of vertices (including both real and ghost nodes) are sufficient to form a convex hull. All ghost speakers are added with a special flag to indicate that they are not part of the real set of loudspeaker nodes to facilitate a downmix computation (cf. clause 7.2.1.3.2.4).

First, the presence of north and south pole ($\phi_{LS} = \pm 90°$) loudspeaker nodes is checked. These are added if they are not present. Next, the number of loudspeakers in the horizontal layer k is computed by counting all the nodes with elevation $\phi_{LS} < 45°$). Based on the value of k, additional ghost speakers are added if necessary:

- For $k = 0$, three ghost speakers are added with $\theta = \{0°, 120°, -120°\}$ and $\phi = 0°$.
- For $k = 1$, two ghost speakers are added with $\theta = \{ \theta_{LS}(k) + 120°, \theta_{LS}(k) - 120°\}$ and $\phi = 0°$.
- For $k > 1$, ghost speakers are added with $\phi = 0°$ at necessary azimuth positions to fill gaps greater than $160°$ on the horizontal plane.

7.2.1.3.2.3 Convex hull construction

Once the ghost speakers have been added, the set of real and ghost loudspeaker nodes totalling *M*nodes (henceforth referred to as vertices) are sufficient to construct a convex hull. The first step of the convex hull construction is to select 4 vertices which form an initial tetrahedron in 3-dimensional space.

This vertex selection is performed in an iterative fashion. The first vertex of the set is selected as the starting vertex. The remaining set of vertices is searched to select a vertex that is non-coincident such that the pair of vertices form an edge with non-zero length. Next, a third vertex is searched for again in the non-selected vertices such that a triangle with non-zero area may be formed (i.e. a vertex that is non-colinear with the first edge). Finally, a fourth vertex is selected from the remainder such that the set of 4 vertices form a tetrahedron with non-zero volume (i.e. all four vertices are non-coplanar).

Once a valid initial tetrahedron is formed, the centroid of this tetrahedron is computed by averaging the cartesian coordinates of the 4 vertices using

$$\frac{\sum_{i=0}^{M} \overrightarrow{\boldsymbol{v}_i}}{M}$$

where $\overrightarrow{\boldsymbol{v}_i}$ is the vector of cartesian coordinates of the i^{th} vertex. Next, the triangular faces of this tetrahedron are stored as an ordered list of the vertices that comprise them. The faces are oriented so that their normal vectors point outwards from the centroid.

At this stage, an initial convex hull has been constructed from a subset of 4 vertices of the entire vertex set. The remaining vertices are now added to the hull one at a time following the procedure listed below:

- The centroid of the current convex hull is computed by averaging the cartesian coordinates of all vertices in the hull.
- For all the faces in the current hull, the distance from the new vertex is determined, if positive (the face points towards the new vertex) the face is marked as visible from the new vertex.
- The edges of all visible faces are broken and the new vertex is used to form new faces
- The list of hull faces is updated to remove the old faces and add the new faces with the correct orientation with respect to the centroid.

After this procedure, all vertices are integrated into the convex hull and the construction is complete.

7.2.1.3.2.4 Ghost loudspeaker downmix

The ghost loudspeakers added to the vertex set in clause 7.2.1.3.2.2 must be downmixed to the real loudspeaker nodes ([14] §3.3). This is performed for each ghost loudspeaker that was added to the convex hull. If the case occurs where a ghost loudspeaker was not used for convex hull construction, it may be ignored.

The downmix is performed by creating a downmix matrix of dimensions $M \times N$, where *M*as previously mentioned is the number of real and ghost loudspeaker nodes, and *N*is the number of real loudspeakers. This downmix matrix performs a mapping from the ghost to real loudspeakers. To populate this matrix, the neighbouring vertices of each ghost loudspeaker are determined and a gain factor of either $1/N_{neighbours}$ or $\sqrt{1/N_{neighbours}}$ gain factor is used to distribute sound energy to each neighbour, depending on the panning mode (EFAP or EFIP).

7.2.1.3.2.5 Polygon formation

In addition to the list of triangular hull faces assembled during convex hull construction as described in clause 7.2.1.3.2.3, a list of polygons is formed to enable N-wise panning ([14]§2.1). To do so, the list of convex hull faces is used as a starting point and then for each face of the hull any vertices which are coplanar to that face are added to form a larger face which is a polygon. This process is repeated for all faces. Once the polygon list has been formed a second pass is performed to remove any duplicates or polygons which are subsets, leaving only a list of unique polygons corresponding to the convex hull of all the vertices.

7.2.1.3.3 EFAP gain computation

For calculating panning gains, the convex hull structure computed at initialization is used to find a subset of the output loudspeaker positions according to the panning position. This subset is essentially a two-dimensional polygon on the spherical surface and encloses the panning position or is coincident or colinear with it on one of its vertices or edges.

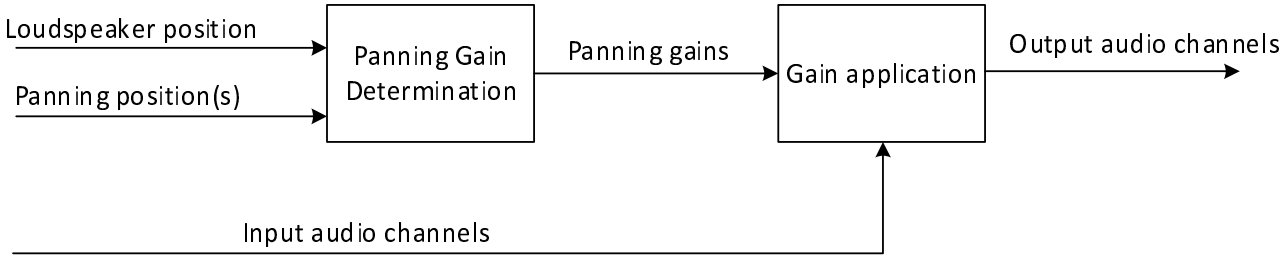


Figure 7.2-1: EFAP gain computation and application

Since the spherical surface is mapped to a two-dimensional plane, comparison of the coordinates of the panning position and loudspeakers, along with computation of Barycentric coordinates can be used to find the facet of the convex hull which contains the panning position. This search is performed similarly to what is described for adding a new convex hull vertex in clause 7.2.1.3.2.3, where the correct subset is the polygon visible from the panning position.

Once the correct speaker subset has been determined, the gains for each individual loudspeaker in the subset are derived by:

- Sub-dividing the subset polygon into triangles that contain the relevant loudspeaker as a vertex.
- Determining whether the panning position lies inside this triangle by performing a check on the Barycentric coordinates.
- Finally, the crossfading gain for the relevant loudspeaker is computed by using the normal vector of the triangle.

The check for whether the panning position lies inside a triangle can be performed by computing the coefficients λ and μ ([14]§3.2):

$$[\lambda, \mu]^T = [b - a, c - a]^{-1}(p - a), \quad (7.2\text{-}1)$$

where a, b and c are coordinate pair vectors (azimuth θ_{LS}, elevation ϕ_{LS}) of the loudspeakers in the given triangle and p is the coordinate pair for the panning position.

If the following conditions are satisfied, the panning position p is located inside the triangle:

$\lambda \geq 0$, (7.2-2)

$\mu \geq 0$, (7.2-3)

$\lambda + \mu \leq 1$. (7.2-4)

The crossfading gain g for a loudspeaker in a triangle can be determined as follows:

$$g = 1 - n(p - a) \quad (7.2\text{-}5)$$

where n is the normal vector of the given triangle, a is the coordinate pair of the relevant loudspeaker and p is coordinate pair of the panning position.

Once the crossfading gains for all the loudspeakers have been computed, a Euclidean norm is performed to obtain the final amplitude panning gains.

In case of EFIP mode, the square root of the sum-normalized amplitude panning gains is used.

Since the panning position is time-dependent, these gains are computed and applied every frame.

7.2.1.4 All-round ambisonic panning and decoding

7.2.1.4.1 Overview

AllRAD is a method which allows rendering of scene-based (ambisonic) audio to an arbitrary arrangement of loudspeakers [15]. This method is used in the IVAS codec for rendering of scene-based audio to loudspeakers, and also intermediate rendering of ambisonics to loudspeakers required for convolution with BRIRs .

IVAS supports input/output of ambisonics up to order 3 with ACN channel ordering and SN3D normalization (cf. clause 4.2.4) and several loudspeaker output formats (cf. clauses 4.3.2 and 7.4.9). To perform rendering of audio from a given ambisonics order to a specified output loudspeaker format, a rendering matrix is used. This matrix is computed by using the AllRAD algorithm (further described in clause 7.2.1.4.3).

Rendering for an input audio frame is performed by multiplying the input audio frame with the rendering matrix to yield the output audio frame with the correct number of output channels for the specific input-output format combination.

7.2.1.4.2 Mono and Stereo rendering

For Mono and Stereo output, the rendering matrix computation does not use the AllRAD algorithm and is instead more straightforward.

For Mono output, the first input channel (ACN 0, or W) is routed to the output and all other channels are ignored.

For Stereo output, the first two input channels (ACN 0 and 1, or W and Y) are used to compute the left and right stereo channels as:

$$L = \frac{W+Y}{2}$$

$$R = \frac{W-Y}{2}$$

7.2.1.4.3 Rendering to other loudspeaker outputs

Computation of the AllRAD matrix consists of two steps ([15] §6.2); decoding of the ambisonics channels to a virtual loudspeaker arrangement (a t-design) and then rendering the virtual loudspeakers onto real loudspeakers using a panning algorithm.

For the IVAS implementation of AllRAD decoding for up to order 3 ambisonics, a t-design with $t = 11$ is used, according to the recommendation in [[15] §2.2] of selecting t so that $t \geq 2N + 1$ where N is the ambisonic truncation order (in this case $N = 3$). Additionally, EFIP (cf. clause 7.2.1.3.2) is used for panning based on subjective experimentation instead of VBAP as originally described in [15].

The AllRAD decoder matrix $D$is the product of the ambisonics to virtual loudspeaker decoding matrix $Y$with the EFIP panning matrix from virtual to real loudspeakers G:

$$D = GY^T$$

where a superscript T denotes the matrix transpose operation. This is a simplified representation of Equation (31) in [15] §6.2. D is of dimension $N \times M$, where N is the number of ambisonic channels (or harmonics) of the input format given by $(l + 1)^2$ where l is the input ambisonic order and M is the number of output loudspeaker channels.

First, EFIP is initialized for the output loudspeaker format to allow computation of the gains to populate the matrix G. Next, for each set of t-design coordinates, the spherical harmonic response is computed to populate a row of the matrix

Y(ambisonics to virtual loudspeaker decoding) and EFIP panning gains are computed to populate a row of the matrix G (virtual to real loudspeaker rendering matrix). In case of a t-design with $t = 11$, this constitutes 70 points. After computation of both G and Y, they are multiplied to produce the AllRAD loudspeaker decoding matrix.

The AllRAD matrix may be computed once at initialization and re-used at runtime to perform rendering. For rendering a lower order of ambisonics, a higher order matrix may be re-used by simply multiplying with only those rows which correspond to the required harmonics (e.g. only 4 of 16 rows for FOA rendering using a HOA3 matrix).

7.2.2 Rendering for binaural headphone reproduction

7.2.2.1 Binaural rendering overview

The IVAS codec has several binaural renderers described in clauses 7.2.2.2 – 7.2.2.5. Depending on the input format, bitrate, IVAS mode and whether head-tracking is enabled different binaural renderer technology is used. Three binaural output modes are supported:

- Binaural output without room acoustic synthesis (no room), command line option BINAURAL,
- Binaural output with room acoustics synthesized using impulse responses (room with IR), command line option BINAURAL_ROOM_IR,
- Binaural output with room acoustics synthesized using parametric reverb, with or without early-reflections (room with reverb), command line option BINAURAL_ROOM_REVERB.

The table 7.2-1 provides an overview of which renderers are used at which operating point for the output formats as listed above.

Table 7.2-1: Overview of renderers used for binaural rendering output modes in the IVAS Codec

IVAS input Format	Bitrate Range [kbps]	IVAS Mode (if applicable)	Binaural rendering output mode (if applicable)	Renderer Used
SBA	13.2 – 80	-	-	Parametric Binaural Renderer
SBA	96 – 512	-	-	FastConv Binaural Renderer
MASA	13.2 - 512	-	-	Parametric Binaural Renderer
ISM	Cf. Table 5.6-1	ParamISM	-	Parametric Binaural Renderer
ISM	Cf. Table 5.6-1	DiscISM	No room or room with reverb	Time Domain Object Renderer
ISM	Cf. Table 5.6-1	DiscISM	Room with IR	Crend Binaural Renderer
MC	Cf. Table 5.7-1	McMASA	-	Parametric Binaural Renderer
MC	Cf. Table 5.7-1	ParamMC	-	FastConv Binaural Renderer
MC	Cf. Table 5.7-1	ParamUpmix	-	FastConv Binaural Renderer
MC	Cf. Table 5.7-1	DiscMC	All except below	Crend Binaural Renderer
MC Planar Layouts (5.1 and 7.1)	Cf. Table 5.7-1	DiscMC	Head tracking enabled for either no room or room with reverb	Time Domain Object Renderer
OMASA	Cf. Table 5.9-1	-	Head tracking enabled for either no room or room with reverb	Same as non-combined format
OSBA	Cf. Clause 5.8.1	-	-	Same as non-combined format

7.2.2.2 Time Domain binaural renderer

7.2.2.2.1 General

The time domain (TD) renderer operates on signals in time domain. In the IVAS decoder it is used for binaural rendering of discrete ISM, where each audio signal is encoded and decoded with a dedicated SCE module. This covers all ISM bit rates, except 3-4 objects for bit rates 24.4 kbps and 32 kbps. Further it is used in the decoder for binaural rendering of 5.1 and 7.1 signals when headtracking is enabled. In the external renderer it is used for all ISM configurations and all multichannel loudspeaker configurations, both with and without headtracking enabled. An overview of the TD binaural renderer is found in Figure 7.2-2 below. An HRIR model accepts the object position metadata along with the headtracking data and generates an HRIR filter pair. The ITD may be modelled as a part of the HRIR, or it may be modelled as a separate parameter. In case an ITD parameter is output, the ITD is synthesis is

performed in the ITD synthesis stage. The time aligned signals are then convolved with the HRIR filter pair to form a binauralized signal.

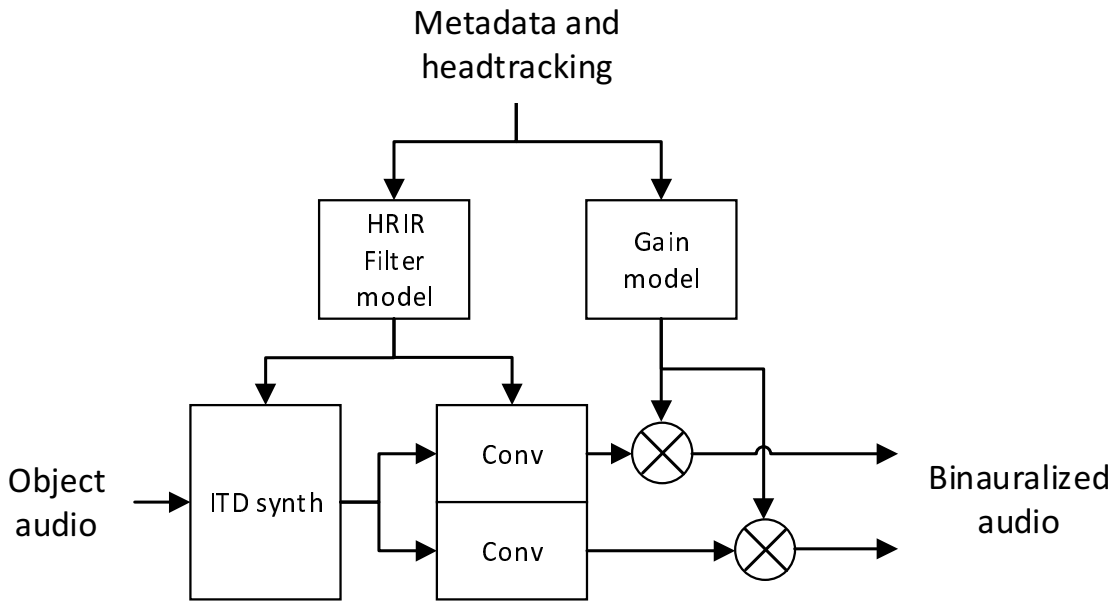


Figure 7.2-2: Overview of TD binaural renderer

7.2.2.2.2 HRIR model

The HR filters for a certain rendering position, specified by an elevation angle ϑ and an azimuth angle φ, are generated based on model parameters representing the HR filter set. The angles (ϑ, ϕ) are the relative angles between the listener and the audio object, based on the object position from the metadata and the listener orientation given by a head-tracker, if head-tracking is in use. The ITD may be modelled as part of the HRIR model or modelled separately, describing the time difference between the left and right HR filters generated from the HRIR model.

The HRIR of the left and right channels are represented as

$$\widehat{\boldsymbol{h}}^{l,r}(\vartheta,\varphi) = \sum_{p=1}^{P}\sum_{q=1}^{Q_p}\sum_{k=1}^{K}\alpha_{p,q,k}^{l,r}\Theta_p(\vartheta)\Phi_{p,q}(\varphi)\boldsymbol{e}_k, \tag{7.2-6}$$

where

$\alpha_{p,q,k}^{l}$ for $p = 1, \dots, P$, $q = 1, \dots, Q_p$, and $k = 1, \dots, K$ is a set of left filter model scalar parameters,
$\alpha_{p,q,k}^{r}$ for $p = 1, \dots, P$, $q = 1, \dots, Q_p$, and $k = 1, \dots, K$ is a set of right filter model scalar parameters,
$\Theta_p(\vartheta)$ for $p = 1, \dots, P$ defines the set of elevation basis function values at the elevation angle ϑ, and
$\Phi_{p,q}(\varphi)$ for $p = 1, \dots, P$ and $q = 1, \dots, Q_p$ defines P sets of Q_p azimuth basis function values at the azimuth angle φ;
and $\boldsymbol{e}_k$ for $k = 1, \dots, K$ is a set of canonical orthonormal basis vectors of length N.

In matrix form the HRIR representation can be written as

$$\hat{h}_k^{l,r}(\vartheta,\varphi) = \underbrace{\begin{pmatrix}\boldsymbol{b}(\vartheta_1,\varphi_1)\\ \vdots \\ \boldsymbol{b}(\vartheta_M,\varphi_M)\end{pmatrix}}_{\boldsymbol{B}}\boldsymbol{\alpha}_k^{l,r} = \boldsymbol{B}\boldsymbol{\alpha}_k^{l,r} \tag{7.2-7}$$

where

$$\begin{aligned}\boldsymbol{b}(\vartheta_m,\varphi_m) &= \left(\Theta_p(\vartheta_m)\Phi_{p,q}(\varphi_m)\colon\ p = 1,\dots,P; q = 1,\dots,Q_p\right)_{row\ vector}\\ \boldsymbol{\alpha}_k^{l,r} &= \left(\alpha_{p,q,k}^{l,r} : p = 1,\dots,P; q = 1,\dots,Q_p\right)_{column\ vector}\end{aligned} \tag{7.2-8}$$

The length of the represented filters K may typically be shorter than the length N of HR filters being modelled, i.e. $K < N$. The basis vectors $\boldsymbol{e}_k$ are chosen as the canonical orthonormal basis vectors of length N

$$\boldsymbol{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ : \\ 0 \end{pmatrix}, \quad \boldsymbol{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ : \\ 0 \end{pmatrix}, \quad \ldots \quad \boldsymbol{e}_N = \begin{pmatrix} 0 \\ : \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{7.2-9}$$

However, for the IVAS default HR filter set, only the K first basis vectors are used to represent the HR filter set. The elevation basis functions $\Theta_p(\vartheta)$ and the elevation dependent azimuth basis functions $\Phi_{p,q}(\varphi)$ are cubic B-spline functions of order $J = 4$. The elevation basis functions are standard B-spline functions while the azimuth basis functions are periodic (circular) with a period of 360 degrees.

For the elevation, the univariate B-spline basis functions of order J over the variable ϑ, where ϑ is in the interval $\theta_A \leq \vartheta \leq \theta_B$, is a set of piecewise polynomial functions of degree $J-1$ defined over that interval. The ranges over which the functions are polynomials are specified with the so-called knot sequence $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_U)$, where the sub-intervals over which the functions are polynomials are $\theta_u \leq \vartheta \leq \theta_{u+1}$ for $u = 1, \ldots, U-1$. In each sub-interval the basis function is a polynomial function of degree $J-1$, i.e.

$$\Theta_p(\vartheta) = \sum_{j=0}^{J-1} \gamma_{j,u,p}^{\Theta} \vartheta^j, \quad \text{for} \quad \theta_u \leq \vartheta < \theta_{u+1}. \tag{7.2-10}$$

The smoothness at the knot-points of a function that is a linear sum of the B-spline basis functions is controlled by the so-called multiplicity sequence $\boldsymbol{m} = (m_1, \ldots, m_U)$, which is a sequence of integers greater than 0 where the value $m_u = i$ means that the $(J-i)$-th derivative at knot-point θ_u is continuous. This means that $i = 1$ gives maximum smoothness, while $i = J$ only gives 0-th derivative continuity. Given the knot sequence and the multiplicity sequence the polynomial model coefficients $\boldsymbol{\gamma}^{\Theta} = \left\{\gamma_{j,u,p}^{\Theta} : j = 0, \ldots, J; u = 1, \ldots, U-1; p = 1, \ldots, P\right\}$ may be obtained iteratively starting with the 0-th degree polynomials using recursion, as described by [13].

Similarly, the azimuth B-spline basis functions of order J over the variable φ are represented as

$$\Phi_q(\varphi) = \sum_{j=0}^{J-1} \gamma_{j,l,q}^{\Phi} \varphi^j, \phi_l \leq \varphi < \phi_{l+1} \tag{7.2-11}$$

for the knot sequence $\{\phi_l : l = 1, \ldots, L\}$.

The azimuth angles are periodic (e.g., circular) in the meaning that the azimuth angle of φ degrees is the same point in space as the azimuth angle of $\varphi + \kappa * 360$ degrees for any integer valued κ. To obtain efficient modelling in the azimuth dimension it is important to use basis functions that are periodic in the same way, such that $f(\varphi) = f(\varphi + \kappa * 360)$. An example of such periodic azimuth basis function is shown in figure 7.2-7.

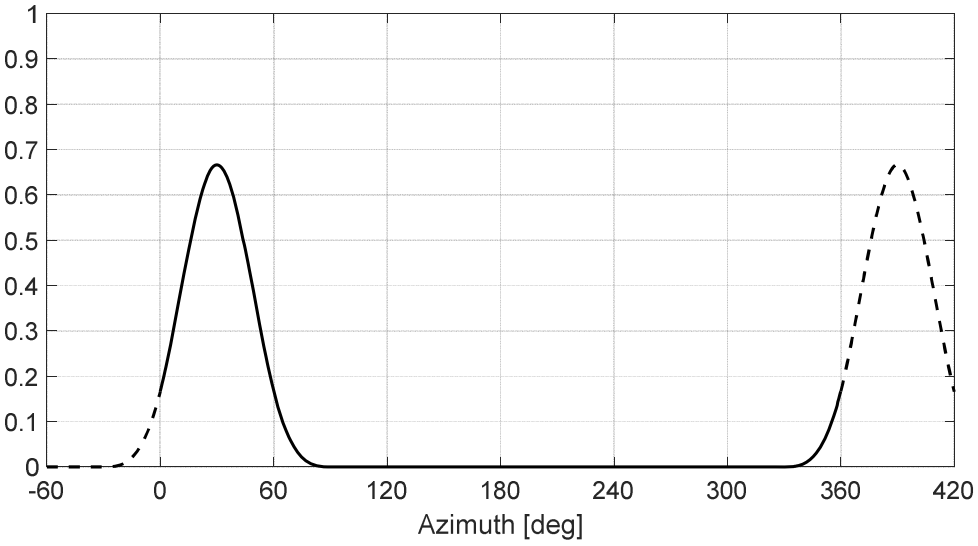


Figure 7.2-3: Example of periodic azimuth basis function.

Figure 7.2-4 and figure 7.2-5 show example standard and periodic B-spline functions used for elevation and azimuth respectively.

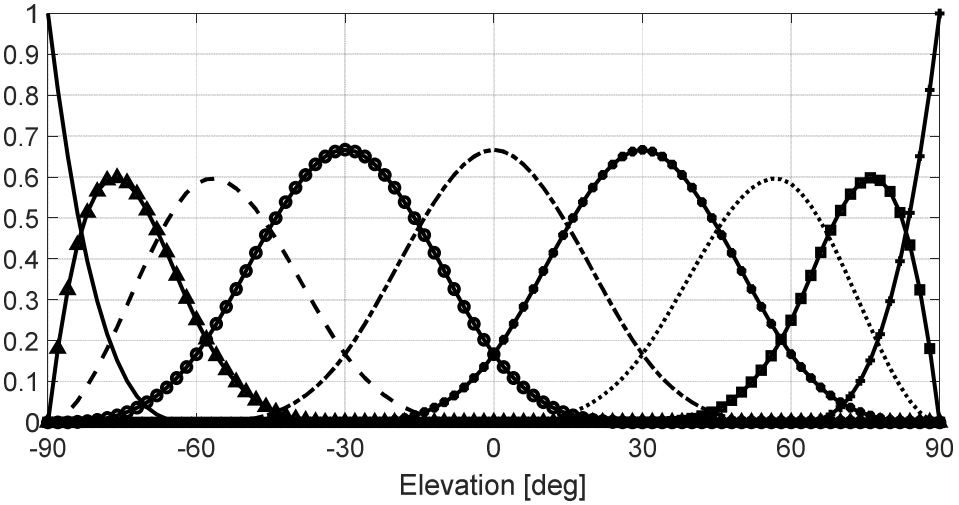


Figure 7.2-4: Example of standard B-spline basis functions for elevation.

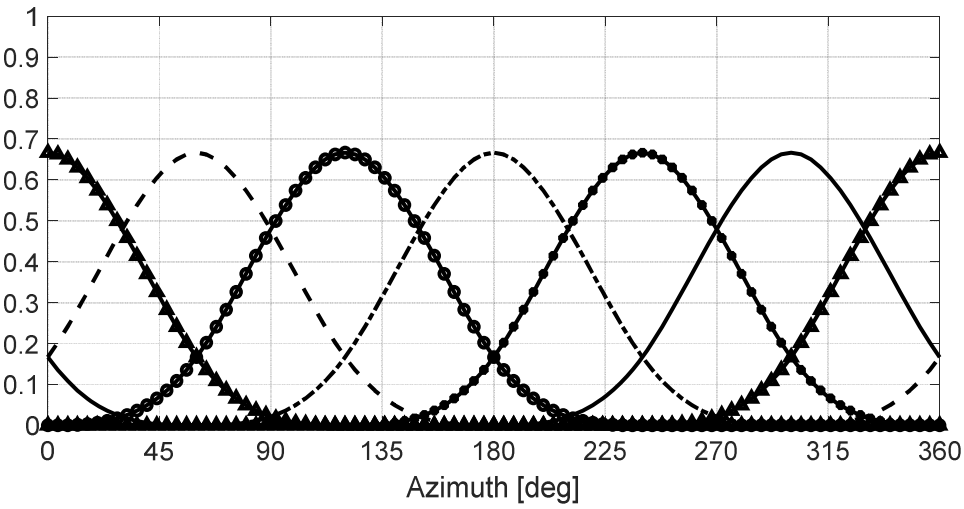


Figure 7.2-5: Example of periodic B-spline functions for azimuth

In order to efficiently generate HR filters from the model representation, compact representations of the elevation and azimuth basis functions are utilized. The compact representations typically correspond to certain non-zero parts of sampled versions of the elevation and azimuth basis functions. Shape metadata and basis function shape data, which identifies the compact representations or converted versions of the compact representations, is obtained, either as given by the model structure, from the ROM tables for the default HR filter set, or from a loaded HR filter binary representation. The shape metadata, which also indicates whether to obtain a converted version of compact representations of the basis functions for the filter generation, may comprise

- The number of basis functions N_{BF} (the number of the azimuth basis functions may be different for different elevations);
- Starting point of each basis function (within the modelling interval) indicated by $Kseq$;
- Shape indices $shape\_idx$ per basis function (identifying which of the stored shapes to use for the basis function) obtained from the model structure;
- Number of samples per segment (knot interval) N_k;
- Start indices B_{start} indicating the starting point of each compact representation of basis functions within the basis function shape data;
- Length of compact representations of basis function L_{BS};

- Conversion parameters
    - A shape resampling factor f_{sub} per basis function; and
    - A flipping indicator per basis function (indicating whether or not to time-reverse the compact representation (stored shape data) for that specific basis function).

The shape indices $shape\_idx$ and a flipping indicator per basis function is obtained from the model structure, meaning it can be determined based on in which knot interval and at which sample point within the knot interval the requested azimuth and elevation angle is located.

The first stage of the filter generation is to obtain the basis function shape data and evaluate the value of the at most four active elevation basis functions for the requested elevation angle ϑ. The non-zero elevation basis function values $B_f^{elev}[p]$ are obtained according to equation (7.2-31) of clause 7.2.2.2.3 with shape metadata ($N_{BF}, Kseq, N_k, L_{BS}$, B_{start}) and basis function shape data (B_{sh}) as obtained for the elevation basis functions. For each of the non-zero elevation basis functions, an azimuth angle within the modelled azimuth range is obtained as

$$\varphi = \mathrm{mod}(\varphi + Kseq[i_p, 0], 360) \tag{7.2-12}$$

If $\varphi < 0$, $\varphi \coloneqq \varphi + 360$ and then

$$\varphi \coloneqq \varphi - Kseq[i_p, 0] \tag{7.2-13}$$

If there is a constant azimuth basis function for a certain elevation, e.g. at the poles, the number of azimuth basis functions is $N_q[p] = 1$, the index for the azimuth basis function $I_q[p] = 0$, and $B_f = \{1,0,0,0\}$. Otherwise, the values of the non-zero azimuth basis functions $B_f^{azim}[q]$ are obtained from the basis function shape data for the requested elevation angle φ in accordance with equation (7.2-39) of clause 7.2.2.2.4 with the elevation-dependent shape metadata ($N_{BF}, Kseq, N_k, L_{BS}, B_{start}, fsub$) and basis function shape data (B_{sh}) for the azimuth basis functions.

Once the sample values of the non-zero elevation and azimuth basis functions are determined, the non-zero elements of the matrix $\boldsymbol{B}$, see equation (7.2-7), are computed as

$$B[p,q] = B_f^{elev}[p]\, B_f^{azim}[q] \;\forall\, p \in \{1, \ldots, N_{idx}^{elev}\}, q \in \{1, \ldots, N_{idx}^{azim}\} \tag{7.2-14}$$

where B_f^{elev} and N_{idx}^{elev} is the elevation basis functions shape values and number of non-zero basis functions, and B_f^{azim} and N_{idx}^{azim} is the azimuth basis functions shape values and number of non-zero basis functions. $\boldsymbol{B}$ only depends on the requested elevation and azimuth angle and is equal for the left and right HR filters. The matrix $\boldsymbol{B}$ may be represented by a vector, keeping track of which index the non-zero p, q corresponds to.

The HRIR are estimated in N_S HR filter sections $\widehat{\boldsymbol{h}}_s$ by the most important components, in terms of a determined energy metric value, for each of the sections. The obtained model parameter matrix $\boldsymbol{\alpha}^{l,r}$, consisting of $N_{PQ} = \sum_p Q_p$ basis vectors of length K, is partitioned over k into $N_S = 3$ sections, corresponding to the HR filter sections $\widehat{\boldsymbol{h}}_s$. The first two sections are of length (number of columns) $L_{sec} = \left\lfloor \frac{K}{N_S} \right\rfloor$, and the last one includes the remaining coefficients (which might be $> L_{sec}$). The N_{PQ} sub-vectors (rows) of each section correspond to each of the $\sum_p Q_p$ elevation and azimuth B-spline products $B[p,q]$ determining the weight of each $\boldsymbol{\alpha}^{l,r}[s]$ row sub-vector for the HR filter generation. To determine the importance of a sub-vector, an energy metric is determined for each row sub-vector of $\boldsymbol{\alpha}^{l,r}$ corresponding to non-zero components of $\boldsymbol{B}$ in the filter generation as

$$E_B^{l,r}[p,q,s] = B[p,q]^2\, E_\alpha^{l,r}[p,q,s] \tag{7.2-15}$$

where $E_\alpha^{l,r}[p,q,s] = \sum_k \alpha_s^{l,r}[p,q]^2$ are pre-computed energies of the row sub-vectors $\alpha_s^{l,r}[p,q]$ of $\boldsymbol{\alpha}^{l,r}$. The total energy of the left and right filter components is computed per section s as

$$E_{tot}^{l,r}[s] = \sum_{p,q} E_B^{l,r}[p,q] \tag{7.2-16}$$

The maximum number of sub-vectors used per section is predetermined as $N_c[s] = \{13,12,11\}$. The maximum number of non-zero components is 16, which is much less than the total number N_{PQ} of sub-vectors (rows) of $\boldsymbol{\alpha}^{l,r}$, but it would be further reduced at the knot points of the B-spline functions, see equations (7.2-24) and (7.2-37). Consequently, the predetermined number of vectors to be used is limited according to

$$N_c[s] \coloneqq \min\left(N_{idx}^{elev}\, N_{idx}^{azim}, N_c[s]\right) \tag{7.2-17}$$

The $N_c[s]$ largest components of $E_B^{l,r}[p,q]$ for each section s are denoted $E_B^{l,r}[\hat{p}_i[s],\hat{q}_i[s]]$ for $i \in [0,\dots,N_c[s]-1]$, meaning the sub-vectors corresponding to $p=\hat{p}_i[s]$ and $q=\hat{q}_i[s]$ are determined to be the most important sub-vectors of section s. To compensate account for lost energy, scale factors $f_E^{l,r}$ are determined as

$$f_E^{l,r}[s] = \sqrt{\frac{E_{tot}^{l,r}[s]}{E_{used}^{l,r}[s]}} \tag{7.2-18}$$

where

$$E_{used}^{l,r}[s] = \sum_i E_B^{l,r}[\hat{p}_i[s],\hat{q}_i[s]]\, f_E^{l,r}[s] = \sqrt{\frac{E_{tot}^{l,r}[s]}{E_{used}^{l,r}[s]}} \tag{7.2-19}$$

Finally, estimated HR filters for the left and right channel are produced for the requested elevation and azimuth angle (ϑ,φ), separately computed for each of the HR filter sections per filter tap k, as

$$\hat{h}_k^{l,r}(\vartheta,\varphi) = f_E^{l,r}[s]\sum_i B[\hat{p}_i[s],\hat{q}_i[s]]\ \alpha_{\hat{p}_i[s],\hat{q}_i[s],k}^{l,r} \qquad \forall\, k \in K_s \tag{7.2-20}$$

for $\hat{p}_i[s],\hat{q}_i[s]$ being the $N_c[s]$ determined most important components of each section, where K_s includes the filter taps (columns of $\boldsymbol{\alpha}^{l,r}$) of the section s.

7.2.2.2.3 Obtain standard spline sample values

The uniform length of the knot intervals (in degrees) is identified as

$$L_k = \frac{Kseq[N_{BF}-3]-Kseq[0]}{N_{BF}-3} \tag{7.2-21}$$

where N_{BF} is the number of basis functions, and $Kseq$ is the sequence of knot points, including multiplicities. The index of the closest sample point within the identified knot interval is determined as

$$i_0 = \left[\!\left[\frac{t-Kseq[0]}{L_k/N_k}\right]\!\right] \tag{7.2-22}$$

where $[\![\cdot]\!]$ denotes rounding and N_k is the number of samples per segment (knot interval). The index of the knot interval is determined as

$$i_k = \lfloor i_0/N_k \rfloor \tag{7.2-23}$$

As the B-splines are of order 4, there are at most four non-zero basis functions, However, at the knot points the last basis function is zero, i.e. the number of non-zero basis functions N_{idx} is

$$\begin{cases} N_{idx}=3 & if \quad i_0 \bmod N_k = 0 \\ N_{idx}=4 & otherwise \end{cases} \tag{7.2-24}$$

Subsequently, the basis function shape data is obtained based on the shape metadata. For each of the N_{idx} non-zero basis functions, the shape index and the start index are determined as

$$start\_idx[i_{nz}] = \max(0, i_{nz}+i_k-3)\ \forall\, i_{nz} \in \{0,1,\dots,N_{idx}-1\} \tag{7.2-25}$$

$$shape\_idx[i_{nz}] = \min(i_{nz}+i_k, \min(3, N_{BF}-1-(i_{nz}+i_k)))\ \forall\, i_{nz} \in \{0,1,\dots,N_{idx}-1\} \tag{7.2-26}$$

Then the offset from the starting point of each basis function i_d is determined as

$$i_{d0}[i_{nz}] = i_0 - start\_idx[i_{nz}]\, N_k \tag{7.2-27}$$

and further based on the shape metadata

$$\begin{aligned} i_d[i_{nz}] &= L_{BS}[shape\_idx[i_{nz}]] - 1 - i_{d0}[i_{nz}] \quad if \quad reverse\_full\_shape \\ i_d[i_{nz}] &= 2\,(L_{BS}[shape\_idx[i_{nz}]] - 1) - i_{d0}[i_{nz}] \quad if \quad reverse\_half\_shape \end{aligned} \tag{7.2-28}$$

where the flipping indicators

$$reverse\_full\_shape = \begin{cases} true & if \quad i_{nz}+i_k > N_{BF}-4 \\ false & otherwise \end{cases} \tag{7.2-29}$$

$$reverse\_half\_shape = \begin{cases} true & if \quad i_{nz} + i_k \leq N_{BF} - 4 \ \wedge \ i_{d0} > L_{BS}[shape\_idx[i_{nz}]] - 1 \\ false & otherwise \end{cases} \quad (7.2\text{-}30)$$

$\forall\, i_{nz} \in \{0, 1, \ldots, N_{idx} - 1\}$ where $\wedge$ denotes logical AND.

The evaluated value of the basis function shape data for each non-zero basis function is determined as

$$B_f[i_{nz}] = B_{sh}[B_{start}[shape\_idx[i_{nz}]] + |i_d[i_{nz}]|], \forall\, i_{nz} \in \{0, 1, \ldots, N_{idx} - 1\} \quad (7.2\text{-}31)$$

where B_{sh} is the compact representation of the basis functions, $B_{start}[shape\_idx]$ identifies which of the compact representations to be used, and i_d defines the index to read from the compact representation to obtain the compact representation or a converted version of the compact representation in determining each sample $B_f[i_{nz}]$.

To be able to efficiently compute $\hat{h}_k^{l,r}(\vartheta, \varphi)$, see equation (7.2-20), the indices of the non-zero basis functions are determined as

$$I_f[i_{nz}] = i_{nz} + i_k \quad (7.2\text{-}32)$$

This way only non-zero components may be included in the sum.

7.2.2.2.4 Obtain periodic spline sample values

The uniform length of the knot intervals (in degrees) is identified as

$$L_k = \frac{Kseq[N_{BF}] - Kseq[0]}{N_{BF}} \quad (7.2\text{-}33)$$

where N_{BF} is the number of basis functions, and $Kseq$ is the sequence of knot points, not including multiplicities.

The number of samples per segment (knot interval) is determined as

$$N_{ksub} = N_k / f_{sub} \quad (7.2\text{-}34)$$

where f_{sub} is the subsampling (shape resampling) factor and N_k is the number of samples per segment (knot interval) of the non-subsampled shape, as indicated by the shape metadata.

The index of the closest sample point within the identified knot interval is determined as

$$i_0 = \left[\!\left[\frac{t - Kseq[0]}{L_k / N_{ksub}}\right]\!\right] \quad (7.2\text{-}35)$$

where $[\![\cdot]\!]$ denotes rounding. The index of the knot interval is determined as

$$i_k = \lfloor i_0 / N_{ksub} \rfloor \quad (7.2\text{-}36)$$

As the B-splines are of order 4, there are at most four non-zero basis functions, However, on the knot points the last basis function is zero, i.e. the number of non-zero basis functions N_{idx} is

$$\begin{cases} N_{idx} = 3 & if \quad i_0 \bmod N_{ksub} = 0 \\ N_{idx} = 4 & otherwise \end{cases} \quad (7.2\text{-}37)$$

With a uniform knot sequence, the basis functions are equal and symmetric, which means a single basis function shape can be used. The basis function shape data obtained based on the shape metadata then only depends on the subsampling factor and the offset within the knot interval determined as

$$i_d[i_{nz}] = i_0 - (i_{nz} + i_k - 1)\, N_{ksub} \quad (7.2\text{-}38)$$

Given the model structure, a negative $i_d[i_{nz}]$ indicates a flip of the compact representation of the basis function. The evaluated value of the basis function shape data for each non-zero basis function is determined as

$$B_f[i_{nz}] = B_{sh}[|i_d[i_{nz}]|\ f_{sub}], \forall\, i_{nz} \in \{0, 1, \ldots, N_{idx} - 1\} \quad (7.2\text{-}39)$$

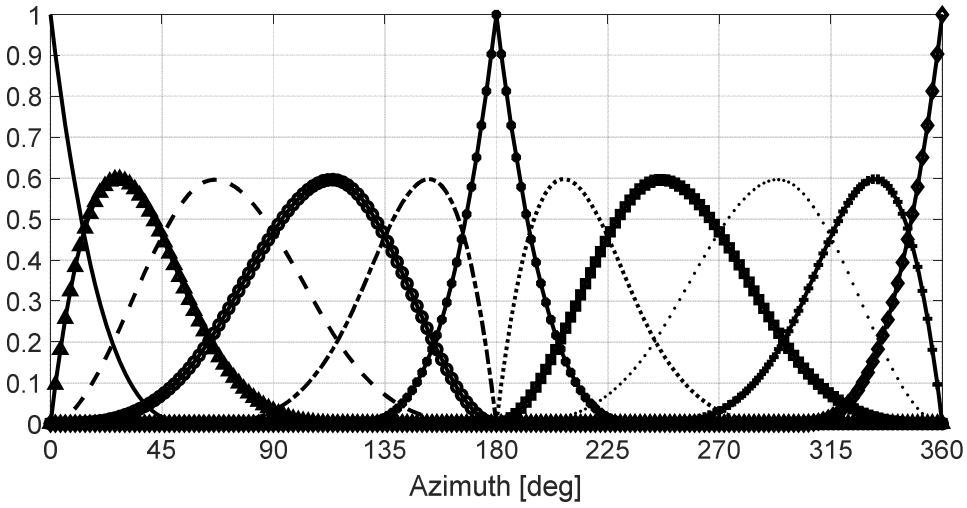
where B_{sh} is the compact representation of the basis functions, and i_d and f_{sub} define the index to read from the compact representation to obtain the compact representation or a converted (flipped or sub-sampled) version of the compact representation in determining each sample $B_f[i_{nz}]$.

To be able to efficiently compute $\hat{h}_k^{l,r}(\vartheta, \varphi)$, see equation (7.2-20), the indices of the non-zero basis functions are determined as

$$I_f[i_{nz}] = (i_{nz} + i_k) \bmod N_{BF} \tag{7.2-40}$$

This way only non-zero components may be included in the sum.

7.2.2.2.5 ITD model



For the case a separate ITD model is used to represent the HR filter set, i.e., the HRIR model represents only the zero-time delay (minimum phase) filters and an ITD model is used to generate the time difference between the left and right channels. This time difference corresponds to the inter-aural time difference.

The ITD model has the same basic structure as the HRIR model, cubic B-spline functions of order $J = 4$, see clause 7.2.2.2.2. The ITD is obtained as a linear combination of model parameters and B-spline basis function weights, evaluated for the requested elevation angle ϑ and an azimuth angle φ, according to

$$\hat{\tau}(\vartheta, \varphi) = \sum_{\tilde{p}=1}^{\tilde{P}} \sum_{\tilde{q}=1}^{\tilde{Q}_{\tilde{p}}} c_{\tilde{p},\tilde{q}} \tilde{\Theta}_{\tilde{p}}(\vartheta) \tilde{\Phi}_{\tilde{p},\tilde{q}}(\varphi) \tag{7.2-41}$$

where $\{\tilde{\Theta}_{\tilde{p}}(\vartheta)\colon \tilde{p} = 1, \ldots, \tilde{P}\}$ and $\{\tilde{\Phi}_{\tilde{p},\tilde{q}}(\varphi)\colon \tilde{q} = 1, \ldots, \tilde{Q}_{\tilde{p}}\}$ are the B-spline basis functions over the elevation angles and the azimuth angles, respectively. $\{c_{\tilde{p},\tilde{q}}\}$ is a set of model parameters.

In matrix form the ITD representation can be written as

$$\hat{\tau}(\vartheta, \varphi) = \underbrace{\begin{pmatrix} \tilde{\boldsymbol{b}}(\vartheta_1, \varphi_1) \\ \vdots \\ \tilde{\boldsymbol{b}}(\vartheta_M, \varphi_M) \end{pmatrix}}_{\tilde{\boldsymbol{B}}} \boldsymbol{c} = \tilde{\boldsymbol{B}}\boldsymbol{c} \tag{7.2-42}$$

where

$$\begin{aligned} \tilde{\boldsymbol{b}}(\vartheta_m, \varphi_m) &= \left(\tilde{\Theta}_{\tilde{p}}(\vartheta_m)\tilde{\Phi}_{\tilde{p},\tilde{q}}(\varphi_m)\colon \tilde{p} = 1, \ldots, \tilde{P}; \tilde{q} = 1, \ldots, \tilde{Q}_{\tilde{p}}\right)_{row\ vector} \\ \boldsymbol{c} &= \left(c_{\tilde{p},\tilde{q}} : \tilde{p} = 1, \ldots, \tilde{P}; \tilde{q} = 1, \ldots, \tilde{Q}_{\tilde{p}}\right)_{column\ vector} \end{aligned} \tag{7.2-43}$$

As ITDs at azimuths between 180 to 360 degrees can be considered a mirror of those at azimuths between 0 to 180 degrees, mirrored B-spline functions are used for azimuth angles of the two intervals, [0, 180] and [180, 360], i.e.

$$\tilde{\Phi}_{\tilde{q}}(\varphi) = \begin{cases} \sum_{j=0}^{J-1} \gamma_{j,l,\tilde{q}}^{\tilde{\Phi}} \varphi^j, & \varphi \le 180 \text{ and } \tilde{\phi}_l \le \varphi < \tilde{\phi}_{l+1} \\ \sum_{j=0}^{J-1} \gamma_{j,l,\tilde{q}}^{\tilde{\Phi}} (360 - \varphi)^j, & \varphi > 180 \text{ and } \tilde{\phi}_l \le 360 - \varphi < \tilde{\phi}_{l+1} \end{cases} \tag{7.2-44}$$

for the knot sequence $\{\tilde{\phi}_l\colon l = 1, \ldots, L_{ITD}\}$. Figure 7.2-6 shows an example of B-spline functions as used for the azimuth modelling of ITD. For the elevation, standard B-spline functions may be used, see example in figure 7.2-4, being polynomial functions

$$\tilde{\theta}_{\tilde{p}}(\vartheta) = \sum_{j=0}^{J-1} \gamma_{j,u,\tilde{p}}^{\tilde{\theta}} \vartheta^j, \quad \tilde{\theta}_u \leq \vartheta < \tilde{\theta}_{u+1}. \tag{7.2-45}$$

for the knot sequence $\{\tilde{\theta}_u : u = 1, \ldots, U_{ITD}\}$.

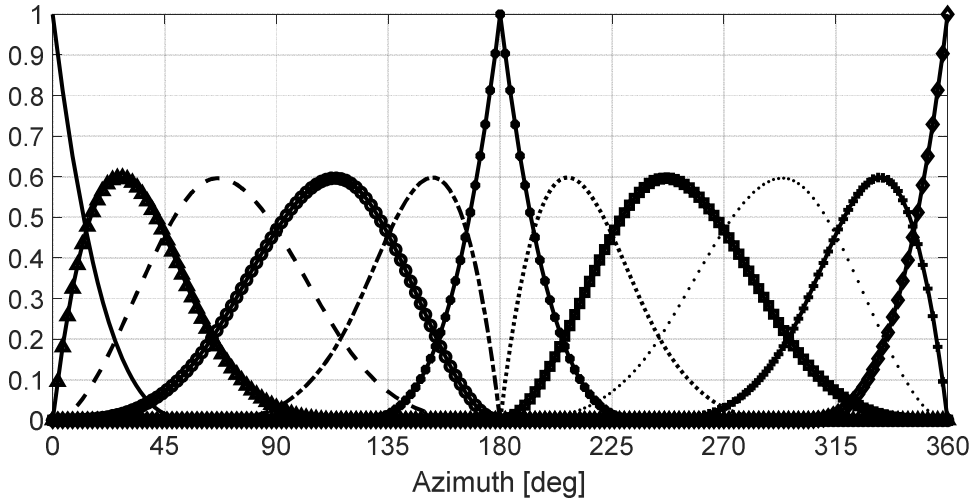


Figure 7.2-6: Example of mirrored standard B-spline functions for azimuth.

To generate an ITD value of the requested elevation angle ϑ and azimuth angle φ, the azimuth is wrapped to the supported range as specified in equation (7.2-21)-(7.2-22). The non-zero elevation basis function values $B_{f,ITD}^{elev}[\tilde{p}]$ are obtained according to equation (7.2-31) of clause 7.2.2.2.3 with shape metadata ($Kseq$, N_{BF}, N_k, L_{BS},B_{start}) and basis function shape data (B_{sh}) as obtained for the elevation basis functions for the elevation basis functions of the ITD model. Similarly, the non-zero basis function values for azimuth $B_{f,ITD}^{azim}[\tilde{q}]$ are obtained according to equation (7.2-31) of clause 7.2.2.2.3 with the elevation-dependent ($Kseq$, N_{BF}, N_k, L_{BS},B_{start}) and basis function shape data (B_{sh}) as obtained for the azimuth basis functions for the ITD model. Note, due to the mirroring $N_{BF} := (N_{BF} + 1)/2$ and $\varphi := 360 - \varphi$ if $\varphi > 180$.

At the poles $\vartheta = \{-90, 90\}$, there is one elevation basis function equal to one and a constant azimuth basis function such that

$$B[\tilde{p}, \tilde{q}] = 1 \tag{7.2-46}$$

At other positions, not being at the poles, the elements of the matrix $\tilde{\boldsymbol{B}}$, are computed by

$$\tilde{B}[\tilde{p}, \tilde{q}] = B_{f,ITD}^{elev}[\tilde{p}]\, B_{f,ITD}^{azim}[\tilde{q}] \;\forall\, \tilde{p} \in \{1, \ldots, N_{idx,ITD}^{elev}\}, \tilde{q} \in \{1, \ldots, N_{idx,ITD}^{azim}\} \tag{7.2-47}$$

Finally, an ITD for frame m is computed as

$$ITD(m) = \hat{\tau}(\vartheta, \varphi) = -[\![f_{resamp} \sum_{\tilde{p},\tilde{q}} \tilde{B}[\tilde{p}, \tilde{q}]\, c_{\tilde{p},\tilde{q}}]\!] \tag{7.2-48}$$

for $\tilde{p}, \tilde{q}$ corresponding to the non-zero B-spline functions of the requested elevation angle ϑ and an azimuth angle φ. The matrix $\tilde{\boldsymbol{B}}$ may be represented by a vector, keeping track of which index the non-zero $\tilde{p}, \tilde{q}$ corresponds to. The negation comes from different conventions between model and renderer. f_{resamp} is a resampling factor in case the rendering is performed with a different sampling rate than the ITD model is representing, typically 48 kHz.

7.2.2.2.6 ITD synthesis

The ITD synthesis adjusts the timing of the signals such that the desired ITD is achieved in the rendered signal. For audio segments where the ITD remains the same, this can is realized by buffering and delaying the signal with the later time of arrival. To keep the delay at a minimum, the signal with the later time of arrival is delayed while the other channel is rendered with zero delay. When the ITD value changes, a time scaling operation needs to be performed in order to change the alignment of the channels. In case the ITD value changes sign, this means the delayed channel needs to be adjusted to zero delay and the other channel is adjusted to be delayed with the new target ITD.

For each object signal $\hat{s}_i(n)$, either input from the decoder or fed to the external renderer, the new object signal frame is fed into a processing buffer. The L_{ITDmem} samples of memory from the preceding frame is appended in front of the new signal frame. The length of the memory is

$$L_{ITDmem} = ITD_{MAX} + L_{sinc} \tag{7.2-49}$$

where ITD_{MAX} is the maximum ITD that can be synthesized and $L_{sinc} = 5$ is the number of samples used in the polyphase resampling stage. First, a buffer length L_3 is calculated to leave look-ahead room for the resampling filter according to

$$\begin{cases} L_3 = \max(0, L_{sinc} - |ITD(m)|) \\ L_{tot} = L_{sf} - L_3 \end{cases} \tag{7.2-50}$$

where $ITD(m)$ is the ITD value of the current subframe m, L_{tot} is the total transition time in samples and L_{sf} is the length of the 5 ms subframe. The ITD transition is set to complete within the subframe, which means L_{sf} is the maximum allowed transition length. At 48 kHz sampling rate, $L_{sf} = 240$. If the sign of the ITD did not change from the previous subframe, or one of them is zero $ITD(m-1)ITD(m) \geq 0$, the time scaling operation only needs to be done on one channel. In that case the number of transition times in samples $L_1$and L_2 are calculated according to

$$\begin{cases} L_1 = L_{tot} \\ L_2 = L_{tot} - L_1 = 0 \\ n_1 = -|ITD(m-1)| \\ L_{in,1} = L_1 + |ITD(m-1)| - |ITD(m)| \\ n_2 = L_1 - |ITD(m)| \\ L_{in,2} = 0 \\ n_3 = L_{tot} \end{cases} \tag{7.2-51}$$

where n_1, n_2, n_3 denote the starting indices of each time shift segment assuming that the current input subframe starts at $n = 0$, $L_{in,1}, L_{in,2}$ is the length of the resampling segments 1 and 2. If the previous and current subframe ITD is non-zero and changes sign $ITD(m-1)ITD(m) < 0$ the transition times L_1 and L_2 are calculated according to

$$\begin{cases} L_1 = \left[\frac{L_{tot}|ITD(m-1)|}{|ITD(m-1)|+|ITD(m)|}\right] \\ L_2 = L_{tot} - L_1 \\ n_1 = -|ITD(m-1)| \\ L_{in,1} = L_1 + |ITD(m-1)| \\ n_2 = L_1 \\ L_{in,2} = L_2 - |ITD(m)| \\ n_3 = L_{tot} - |ITD(m)| \end{cases} \tag{7.2-52}$$

where $[\cdot]$ denotes rounding to the nearest integer. Next, the output buffers A and B are assembled by time-shifting the signal using the transition lengths L_1 and L_2 as illustrated in figure 7.2-7. First, a resampling is done of the buffer starting from n_1 of length $L_{in,1}$ to the first L_1 samples of output buffer A. The last $L_{sf} - L_1$ samples are populated by copying the remainder of the input buffer to output buffer A. Then, the first samples of the input buffer starting from index 0 are copied to the first L_1 samples of output buffer B. The next L_2 samples of output buffer B are created by resampling the samples starting from n_2 in the input buffer of length $L_{in,2}$. Finally, the last L_3 samples of the input buffer are copied to output buffer B. The last L_{ITDmem} samples of the input frame is stored in memory for processing the next subframe. Output buffers A and B are assigned to the output channels 0 and 1 for left and right HRIR filtering respectively. The assignment of the output channels is done based on the signs of $ITD(m)$ and $ITD(m-1)$ as follows,

$$\begin{cases} (A, B) \rightarrow (1,0), \ ITD(m-1) = 0 \wedge ITD(m) > 0 \ \vee ITD(m-1) > 0 \\ (A, B) \rightarrow (0,1), \ otherwise \end{cases} \tag{7.2-53}$$

where $\wedge$ denotes logical AND and $\vee$ denotes inclusive OR. The resampling operations are implemented using a polyphase filter with a sinc function from a lookup table.

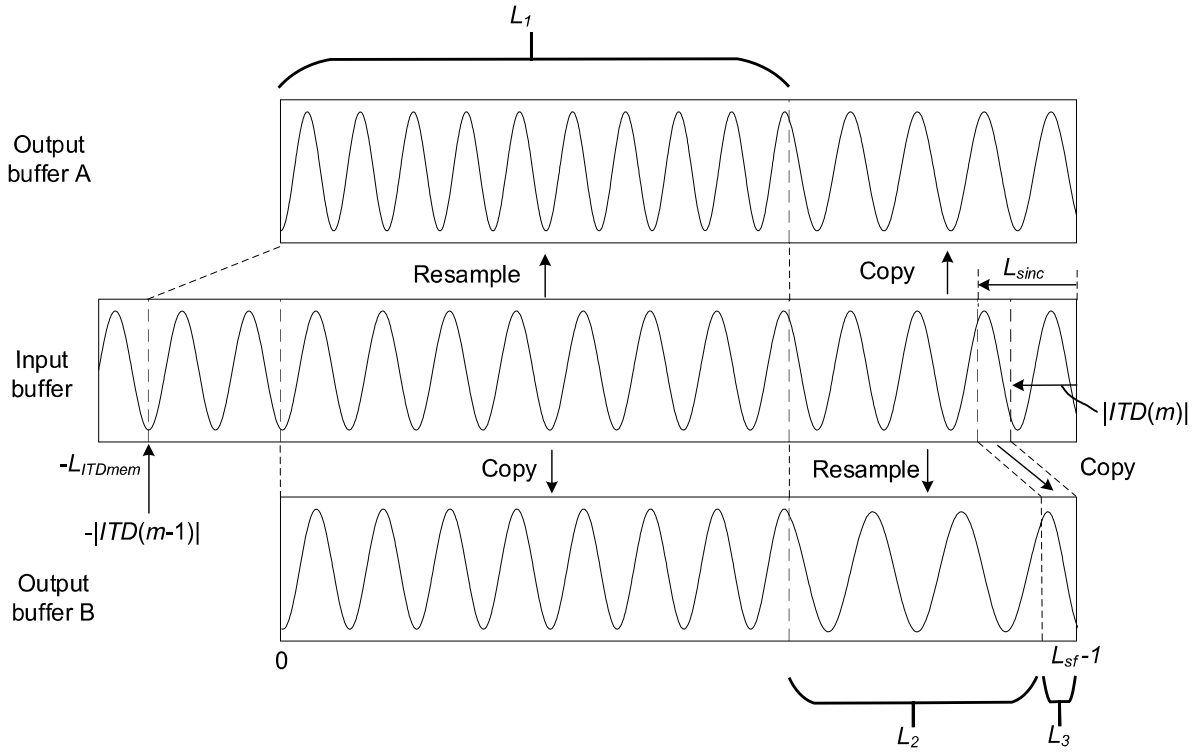


Figure 7.2-7: Resampling and copying operations when changing ITD.

7.2.2.2.7 Distance and Direction Gain

Distance gain is considered highest when the sound source is closer to the listener, and it decreases with the increased distance. Two distance attenuation models are defined to calculate the distance gain, namely $TDREND\_DIST\_ATTEN\_MODEL\_INV\_DIST$ and $TDREND\_DIST\_ATTEN\_MODEL\_INV\_DIST\_CLAMPED$. For IVAS codec use cases, the second model is utilized.

The only difference between these two models is that in the second model, the listener's relative distance to the source, $dist_{list,rel}$, is limited between $dist_{ref}$ and $dist_{max}$:

$$dist_{list.rel} = \begin{cases} dist_{ref}, & dist_{list,rel} < dist_{ref} \\ dist_{max}, & dist_{list,rel} > dist_{max} \end{cases} \quad (7.2\text{-}54)$$

The reference distance and the maximum distance are set to $dist_{ref} = 1$ and $dist_{max} = 15.75$ respectively. For the maximum distance, the highest value radius metadata parameter is allowed to take is used. The distance gain, G_{dist}, is then derived from the following equation:

$$G_{dist} = dist_{ref} / \left(dist_{ref} + f_{rollof} \left(dist_{list.rel} - dist_{ref} \right) \right) \quad (7.2\text{-}55)$$

where roll of factor, f_{rollof}, is a constant with the value of 1. The relative distance of the listener, $dist_{list,rel}$, is determined from both the difference $p_{list,abs}$ between the position of the listener $pos_{listener}$ and the position of the source pos_{src} in 3-D, and the orientation of the listener $list[front, right, up]$. The position of the listener in the XYZ domain, $pos_{listener}[x, y, z]$ is provided with the headtracking data as well as the orientation of the listener, $list[front, right, up]$.

The absolute distance of the listener $p_{list,abs}$ according to the difference is:

$$p_{list,abs}[x, y, z] = \begin{bmatrix} pos_{src}[x] - pos_{listener}[x] \\ pos_{src}[y] - pos_{listener}[y] \\ pos_{src}[z] - pos_{listener}[z] \end{bmatrix} \quad (7.2\text{-}56)$$

where $pos_{src}[x, y, z]$ refers the location of the source in the xyz domain which is provided with the metadata parameters; $azimuth$, $elevation$, and $radius$. The source position, $pos_{src}[x, y, z]$, is derived from metadata according to the following formula:

$$pos_{src}[x,y,z] = \begin{bmatrix} radius\cos\left(elevation\frac{\pi}{180}\right)\cos\left(azimuth\frac{\pi}{180}\right) \\ radius\cos\left(elevation\frac{\pi}{180}\right)\sin\left(azimuth\frac{\pi}{180}\right) \\ radius\sin\left(elevation\frac{\pi}{180}\right) \end{bmatrix} \quad (7.2\text{-}57)$$

The distance $dist_{list}[x,y,z]$ is derived from the absolute distance of the listener $p_{list,abs}$ and the orientation of the listener $list[front, right, up]$ with:

$$dist_{list}[x,y,z] = \begin{bmatrix} p_{list,abs}[x]list_{front}[0] + p_{list,abs}[y]list_{front}[1] + p_{list,abs}[z]list_{front}[2] \\ p_{list,abs}[x]list_{right}[0] + p_{list,abs}[y]list_{right}[1] + p_{list,abs}[z]list_{right}[2] \\ p_{list,abs}[x]list_{up}[0] + p_{list,abs}[y]list_{up}[1] + p_{list,abs}[z]list_{up}[2] \end{bmatrix} \quad (7.2\text{-}58)$$

The relative distance of the listener is the Euclidean norm of $dist_{list,xyz}$:

$$dist_{list,rel} = \sqrt{dist_{list}[x]^2 + dist_{list}[y]^2 + dist_{list}[z]^2} \quad (7.2\text{-}59)$$

The derivation of the orientation of the listener $list_{front,right,up}$ from the head-tracking parameters w, x, y, z includes the conversion from quaternions to rotation matrix and normalization of the orientation vectors. The conversion from quaternions to rotation matrix follows the equation in 7.4-5 as:

$$LFront[0,1,2] = [w^2 + x^2 - y^2 - z^2, 2(xy - wx), 2(xz + wy)] \quad (7.2\text{-}60)$$

$$LUp[0,1,2] = [2(xz - wy), 2(yz + wx), w^2 - x^2 - y^2 + z^2] \quad (7.2\text{-}61)$$

The normalization then involves element-wise division operation:

$$list_{front}[0,1,2] = LFront[0,1,2]/\sqrt{LFront[0]^2 + LFront[1]^2 + LFront[2]^2} \quad (7.2\text{-}62)$$

The orthonormal right vector is derived through the cross product of the normalized front vector and up vector:

$$list_{right}[0,1,2] = \begin{bmatrix} list_{front}[1]LUp[2] - list_{front}[2]LUp[1] \\ list_{front}[2]LUp[0] - list_{front}[0]LUp[2] \\ list_{front}[0]LUp[1] - list_{front}[1]LUp[0] \end{bmatrix} \quad (7.2\text{-}63)$$

$$list_{right}[0,1,2] = list_{right}[0,1,2]/\sqrt{list_{right}[0]^2 + list_{right}[1]^2 + list_{right}[2]^2} \quad (7.2\text{-}64)$$

Similarly, the orthonormal up vector is derived through the cross product of the normalized front and normalized up vectors:

$$list_{up}[0,1,2] = \begin{bmatrix} list_{right}[1]list_{front}[2] - list_{right}[2]list_{front}[1] \\ list_{right}[2]list_{front}[0] - list_{right}[0]list_{front}[2] \\ list_{right}[0]list_{front}[1] - list_{right}[1]list_{front}[0] \end{bmatrix} \quad (7.2\text{-}65)$$

$$list_{up}[0,1,2] = list_{up}[0,1,2]/\sqrt{list_{up}[0]^2 + list_{up}[1]^2 + list_{up}[2]^2} \quad (7.2\text{-}66)$$

Direction gain is determined by the orientation of the sound source and its directivity configuration. The orientation of the sound source is available for higher bitrates with the extended metadata as described in the clause 5.6.4.3. Directivity configuration parameters consist of cone inner angle, cone outer angle and cone outer gain. The direction gain is highest within the cone inner angle, meaning the value will be one. The cone outer gain reflects to the gain outside the cone, towards the back of the listener. The direction gain has the minimum value, i.e., cone outer gain, in this region which is defined with the cone outer angle. The direction gain for the region between the inner and outer angle is interpolated according to:

$$G_{dir} = 1 - (2AngleDeg - InnerAngle/OuterAngle - InnerAngle)(1 - OuterGain) \quad (7.2\text{-}67)$$

$$AngleDeg = \frac{180}{\pi}\cos^{-1}\left(PCo/\sqrt{p_{list,abs}[x]^2 + p_{list,abs}[y]^2 + p_{list,abs}[z]^2}\right) \quad (7.2\text{-}68)$$

where PCo is projecting the absolute distance of the listener, $p_{list,abs}$, onto the direction of the source, $dirSrc$ by:

$$PCo = \left(-p_{list,abs}[x]dirSrc[x]\right) + \left(-p_{list,abs}[y]dirSrc[y]\right) + \left(-p_{list,abs}[z]dirSrc[z]\right) \quad (7.2\text{-}69)$$

The direction of the source, $dirSrc$, is derived in a similar logic to the derivation of the source position in Equations (7.2-57). The orientation metadata parameters, yaw and $pitch$ are provided in the metadata file:

$$dirSrc[x,y,z] = \begin{bmatrix} \cos\left(pitch\frac{\pi}{180}\right)\cos\left(yaw\frac{\pi}{180}\right) \\ \cos\left(pitch\frac{\pi}{180}\right)\sin\left(yaw\frac{\pi}{180}\right) \\ \sin\left(pitch\frac{\pi}{180}\right) \end{bmatrix} \quad (7.2\text{-}70)$$

The directivity configuration parameters are illustrated in 2-D in Figure 7.2-8. The gain is maximum at 1 in the front region which is highlighted with the darkest shade. The lightest shade in the back reflects to the outer gain, and the region between the inner angle and outer angle appears in different shades according to the gain interpolation.

The final gain for rendering is derived from G_{dist} and G_{dir} according to:

$$G_{source} = G_{dist}G_{dir} \quad (7.2\text{-}71)$$

The gain G_{source} is applied to the rendered source at a scaling factor in the HRIR convolution in 7.2.2.2.8.

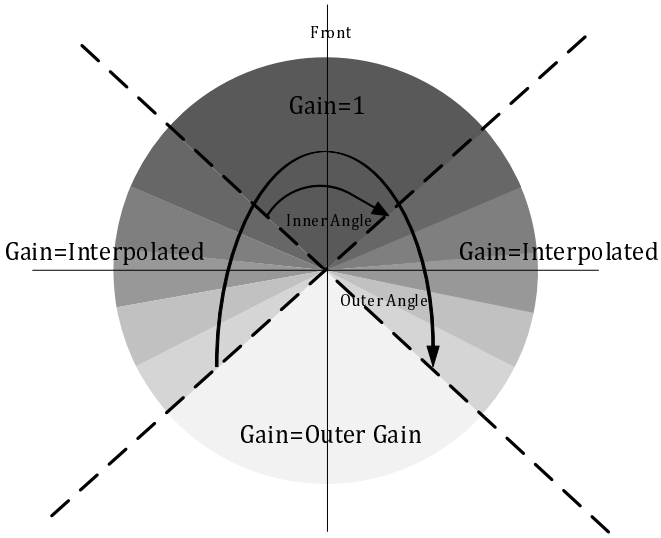


Figure 7.2-8: Directivity Configuration according to the source orientation in 2-D.

7.2.2.2.8 HRIR convolution

The gain, G_{source}, for the current subframe is transferred for HRIR convolution and the change between frames is evaluated for further gain interpolation. If the difference from the previous frame for azimuth or elevation is higher than zero, an interpolation count variable, cnt_{intp} is set and subjected to a threshold of $thr = 12$:

$$cnt_{intp} = \begin{cases} \Delta_{angle}, & \Delta_{angle} < 12 \\ 12, & else \end{cases} \quad (7.2\text{-}72)$$

$$\Delta_{angle} = \max(|\Delta_{azimuth}|100, |\Delta_{elevation}|100) \quad (7.2\text{-}73)$$

$$\Delta_{azimuth} = azimuth - azimuth^{[-1]} \quad (7.2\text{-}74)$$

$$\Delta_{elevation} = elevation - elevation^{[-1]} \quad (7.2\text{-}75)$$

If cnt_{intp} is bigger than '*zero*' and subframe update flag is set, the right and left HR filter interpolation delta is calculated.

$$\Delta h_{left} = \frac{\widehat{\boldsymbol{h}}_l - \widehat{\boldsymbol{h}}_l^{[-1]}}{cnt_{intp}}, \qquad \Delta h_{right} = \frac{\widehat{\boldsymbol{h}}_r - \widehat{\boldsymbol{h}}_r^{[-1]}}{cnt_{intp}} \quad (7.2\text{-}76)$$

Following the application of ITD to $\hat{h}_k^{l,r}(\vartheta, \varphi)$ by delaying the late channel as described in clause 7.2.2.2.6, a gain $G_{tmp}(m; n)$ is applied to the convolution of the HR filter for the audio signals of each of the left (l) and right (r) channel. To allow for smooth transition in case of abrupt gain changes, a gain parameter for the previous frame, G_{prev}, is defined and initialized to $G_{prev} = 1$. Gain, G_{source}, is calculated based on the relative distance and directivity of the listener based on the input from head tracking and metadata as described in 7.2.2.2.7. A temporary gain value for each sample in the frame, $G_{tmp}(m; n)$, is defined and initially $G_{tmp}(m; 1) = G_{source}$. The resulting signal $s_{l,r}(m; n)$ is then:

$$s_{l,r}(m; n) = \left(s_{in}(m; n) * \tilde{h}_{l,r}(m; n)\right) G_{tmp}(m; n) \quad (7.2\text{-}77)$$

where $*$ stands for convolution, and $s_{in}(m; n)$ denotes the audio signal being filtered. $\tilde{h}_{l,r}(m; n)$ is calculated based on the generated pair of HR filters $\hat{h}_k^{l,r}(\vartheta, \varphi)$, adjusted for ITD, by a linear interpolation using $\Delta h_{left,right}$ from equation (7.2-76) as a step for the samples $n < cnt\_intp$. $G_{tmp}(m; n)$ is applied via linear interpolation by incrementing its value at each step to assure a smooth transition. The gain difference between two subsequent subframes $G_{source} - G_{prev}$ is divided to the length of subframe to calculate the increment step. At the end of each subframe, the value of G_{prev} is updated to the value of G_{source}.

7.2.2.3 Parametric binauralizer and parametric stereo renderer

7.2.2.3.1 Overview

The parametric binauralizer and stereo renderer operates on the following IVAS formats and operations: MASA, OMASA, multi-channel (in McMASA mode), SBA, OSBA, and ISM, i.e., the input to the encoder has been spatial audio (containing audio signal(s), and spatial metadata in case of MASA, OMASA, OSBA, and ISM formats) in one of these formats, and it is now being rendered to binaural or stereo output. The IVAS format being processed (i.e., whether operating on MASA, OMASA, multi-channel (in McMASA mode), SBA, OSBA, or ISM format) is determined by decoding it from the received IVAS bitstream (see clause 6.1), and the renderer obtains (or receives) this information (i.e., the IVAS format).

The parametric binaural (and stereo) rendering is performed in subframes, where m denotes the subframe index. A subframe contains N_{slots} CLDFB slots (in non-JBM operation, $N_{slots} = 4$, in JBM operation $N_{slots} = 1 \ldots 7$). The data determined at the rendering of the previous subframes (i.e., subframes m-1 and earlier) affects the rendering of the present subframe m due to temporal averaging and interpolation. The binaural renderer system described in the following is also capable to render a stereo signal instead of a binaural signal. Also, binaural sound with and without room effect can be reproduced.

As an input, the renderer obtains (or receives) a spatial audio signal containing one or two audio signals (i.e., transport audio signals) and associated spatial metadata. The number of transport audio signals (i.e., one or two) is determined by decoding it from the received IVAS bitstream (see clause 6.1), and the renderer receives this information (i.e., the number of transport audio signals). The spatial metadata obtained (or received) by the renderer contains the following parameters in frequency bands b: direction (as azimuth $\theta(b, m, i)$ and elevation $\phi(b, m, i)$), direct-to-total energy ratio $r_{dir}(b, m, i)$, spread coherence $\zeta(b, m, i)$, and surround coherence $\gamma(b, m)$. In case of the SBA and OSBA formats, the spatial metadata contains SPAR metadata. The audio signals and the spatial metadata are used for providing spatial audio reproduction (i.e., to enable the rendering of spatial audio).

In addition, the input to the renderer includes a separated centre channel audio signal when the IVAS format is multi-channel and operating in McMASA “separate channel” mode. In addition, the input to the renderer contains a separated object audio signal and associated object metadata when the IVAS format is OMASA and operating in “one object with MASA representation coding mode” or “parametric one object coding mode”. In addition, the input to the renderer contains all object audio signals and associated object metadata when the IVAS format is OMASA and operating in “discrete coding mode”.

Furthermore, head orientation data and external orientation data may be received. Head tracking is described in clause 7.4.3, external orientation input is described in clause 7.4.5, and rotation combining functionality for determining the final rotation data to be used in the binaural rendering is described in clause 7.4.6. Head tracking (and external orientation information) should be used if they are available for improved spatial audio experience. In the following it is referred to head orientation and head tracking for simplicity regardless of which components the orientation data is originally composed of, and which processing has been applied to it prior to the rendering step.

In addition, the renderer (when rendering binaural output) obtains a room effect control indication, and based on this indication, it is determined whether to apply a room effect to the audio signal(s) (of the spatial audio signal) or to not apply the room effect. For rendering binaural audio with a room effect, two data sets related to binaural rendering are

obtained. The first is a pre-defined data set containing spherical harmonics to binaural conversion matrices. It has been created based on head related impulse responses or transfer functions (HRTF). The second data set contains reverberation early part energy correction gains (which are used for modifying the resulting spectrum that is obtained from the rendering according to the first data set), and late reverberation energy correction gains and reverberation times. It has been created based on binaural room impulse responses or transfer functions (BRIR). Thus, the binaural output signal is generated based on a combination of the spatial audio signal and the mentioned two data sets, i.e., the binaural output signal is generated using the transport audio signal(s) and associated metadata according to a combination of these two data sets.

The binaural audio signal with the room effect is rendered by generating a first part binaural signal based on the transport audio signal(s) and the spatial metadata, generating a second part binaural signal by applying room effect to the transport audio signal(s) (using a reverberator), and combining the first part binaural audio signal and second part binaural signal to generate a combined binaural audio signal (see 7.2.2.3.5 for details).

The fetching of the temporally correct spatial metadata parameter values for the current subframe m so that they are in sync with the audio signals is handled in clause 6.2.7. It operates differently for the JBM and non-JBM use. Fetching the correct spatial metadata values is not discussed in the following, it is assumed that it has already been correctly performed, as described in the aforementioned clause.

The binaural (or stereo) rendering is based on using covariance matrices. The transport signal covariance matrices are determined, and the target covariance matrices are determined. Based on these covariance matrices, processing matrices are determined to process the transport audio signals to a determined target.

The details of the binaural (with or without the room effect) and stereo rendering are described in the following (and the related clauses 7.2.2.3.2– 7.2.2.3.10).

First, the transport audio signals are transformed to the time-frequency domain with a 60-bin (with the sampling rate of 48 kHz) complex low-delay filter-bank (CLDFB) (see clause 6.2.5 for details), resulting in $S(k, n, i)$, where k is the frequency bin index, n is the CLDFB temporal slot index, and i is the transport audio signal channel index. In case there is only one transport audio signal, the time-frequency transformed signal $S(k, n, 1)$ is duplicated to a dual-mono signal where $S(k, n, 2) = S(k, n, 1)$. Any object audio signals, as well as the potential separated centre channel signal, are also transformed to a time-frequency representation using the same filter bank.

If the IVAS format is SBA or OSBA, MASA metadata is determined from the SPAR metadata as detailed in clause 7.2.2.3.9, and the prototype audio signals are determined as detailed in clause 7.2.2.3.10.

If the head tracking is enabled, and if the number of transport audio signals is two, the transport audio signals are processed for improved rendering quality using the methods described in clause 7.2.2.3.2.

Then, the input and target covariance matrices are determined based on the transport audio signal(s) and the spatial metadata as described in clause 7.2.2.3.3.

Then, the processing matrices are determined based on the determined input and target covariance matrices as described in clause 7.2.2.3.4.

And, finally, the binaural signals (with or without the room effect) or the stereo signals are rendered to reproduce the sound scene depicted by the input audio signals and the spatial metadata by processing the input audio signals using the determined processing matrices as described in clause 7.2.2.3.5.

7.2.2.3.2 Pre-processing the transport audio signals based on head orientation

When there are two transport audio signals (i.e., not duplicated mono transport audio signal), the transport audio signals are adapted based on the orientation or position parameter indicating user head orientation. This is performed by analysing the two channel transport audio signals to determine an inter-channel property and based on that property and the obtained head orientation parameter determining mixing information to process the two transport audio channels to two processed transport audio channels. These processed transport audio channels then replace the originals in the following clauses and are used (together with the head orientation parameter and spatial metadata) for generating the output binaural audio signals.

In case head orientation data is not available, or if there was only one transport audio channel, or if stereo output is used, none of the following steps in this clause are performed.

First part of the pre-processing of the transport audio signals is to adaptively cross-mix them in a condition when the user is facing side directions, using the following steps.

First, the rotation matrix **R** is obtained based on the head orientation (see $R_{com}(m)$ in clause 7.4.6.1). From the rotation matrix **R**, the second-row second-column entry is selected, which is denoted r_y. The subscript y refers to the entry being the y-axis-to-y-axis processing element of the rotation matrix. Then, a first mono factor is determined based on the head orientation by

$$f_{\text{mono,R}} = trunc_{0,1}\left(\frac{0.6 - |r_y|}{0.4}\right)$$

where $trunc_{0,1}$ is an operation that sets values smaller than 0 to 0, and values larger than 1 to 1. f_{mono} is the same for all bands but may vary over time.

Then, the transport audio signals are analysed to determine an inter-channel property (namely, the inter-channel level differences between the two channels of the transport audio signals) based on which the transport audio signals are to be mixed. The following processing steps are performed in frequency bands according to Table 5.5-1. For each band index b in that grouping, there is a lower and upper bin index $k_{\text{low}}(b)$ and $k_{\text{high}}(b)$.

First, the channel energies are formulated in bands b by

$$E_{in}(b,m,i) = \sum_{k=k_{\text{low}}(b)}^{k_{\text{high}}(b)} \left(\sum_{n=n_{first}(m)}^{n_{last}(m)} |\text{S}(k,n,i)|^2 \right)$$

where $n_{first}(m)$ and $n_{last}(m)$ are the first and last temporal slots of the current subframe m, and i is the transport audio signal channel index. Then, temporal averaging is performed by

$$E'_{in}(b,m,i) = \alpha_{E_{in}} E'_{in}(b,m-1,i) + (1-\alpha_{E_{in}}) E_{in}(b,m,i)$$

where $\alpha_{E_{in}} = 0.81450625$ and where $E'_{in}(b,0,i) = 0$ (for the first subframe of the first frame). Then, the inter-channel level difference (ILD) and the absolute value of it are determined by

$$ILD(b,m) = \left| 10 \log_{10} \frac{E'_{in}(b,m,1)}{E'_{in}(b,m,2)} \right|$$

Then a louder and softer channel indices l and s are defined so that if $E'_{in}(b,m,2) > E'_{in}(b,m,1)$, then $l = 2$ and $s = 1$, and otherwise $l = 1$ and $s = 2$.

Then a second mono factor is determined based on the ILD by

$$f_{\text{mono,ILD}} = trunc_{0,1}\left(\frac{ILD(b,m) - 1}{3}\right)$$

Next, mixing information is determined by combining the two mono factors

$$f_{\text{mono}} = f_{\text{mono,R}} f_{\text{mono,ILD}}$$

Then, for each bin from $k_{\text{low}}(b)$ to $k_{\text{high}}(b)$ and from each slot from $n_{first}(m)$ to $n_{last}(m)$ the transport signal is mixed by

$$S_{mix}(k,n,s) = f_{\text{mono}}\big(S(k,n,1) + S(k,n,2)\big) + (1 - f_{\text{mono}}) S(k,n,s)$$

$$S_{mix}(k,n,l) = S_{mix}(k,n,l)$$

Energy after mixing is formulated

$$E_{mix}(b,m,i) = \sum_{k=k_{\text{low}}(b)}^{k_{\text{high}}(b)} \left(\sum_{n=n_{first}(m)}^{n_{last}(m)} |\text{S}_{mix}(k,n,i)|^2 \right)$$

This is temporally averaged

$$E'_{mix}(b,m,i) = \alpha_{E_{in}} E'_{mix}(b,m-1,i) + (1-\alpha_{E_{in}}) E_{mix}(b,m,i)$$

where $E'_{mix}(b,0,i) = 0$ (for the first subframe of the first frame).

Finally, the transport signals are equalized, resulting in adapted transport audio signals, which are set to replace the original transport audio signals, for each bin from $k_{\mathrm{low}}(b)$ to $k_{\mathrm{high}}(b)$ and from each slot from $n_{first}(m)$ to $n_{last}(m)$, by

$$S(k,n,i) \coloneqq \mathrm{S}_{mix}(k,n,i)\min\left(1,\sqrt{\frac{E'_{in}(b,m,1)+E'_{in}(b,m,2)}{E'_{mix}(b,m,1)+E'_{mix}(b,m,2)}}\right)$$

Next, for the transport audio signals, it is checked if the transport audio signals need to be switched (so that left and right channels replace each other). Such processing is performed so that when the user faces rear directions, the transport audio signal aligns better in the left-right axis for the current head orientation. The processing has an interpolation factor $f_{interp}(n)$ so that $f_{interp}(0) = 0$.

The left-right channel switching process has a state, and this state can be one of these four: State **A** for "switching", state **B** for "not switching", and then there can be two transition states, denoted **A→B** and **B→A**.

As a first step, it is checked if the system is at one of the transition states. If not, then following steps are performed: If in state **A** and $r_y < -0.17$, then change state to **A→B**; If in state **B** and $r_y > 0.17$, then change state to **B→A**.

Then, if the system is in state **A**, no switching is done to the transport audio signals $S(k,n,i)$ for any of the indices n in the current subframe m, and the following operations are not performed.

If the system is in state **B**, transport audio signals are, for all indices n in the current subframe m, switched so that the data at $S(k,n,1)$ and $S(k,n,2)$ replace each other. In this condition, the following operations are not performed.

Then, the following steps are performed when the system is at either of the transition states **A→B** or **B→A**. The following steps are performed in this order for each slot n of the present subframe m.

First, the interpolation value is updated by

$$f_{interp}(n) = f_{interp}(n-1) + 0.0025$$

Next, if $f_{interp}(n) > 0.999$, then $f_{interp}(n)$ is set to 0 and if in mode **A→B** the state is set to **B**; and if in state **B→A** the mode is set to **A**. If either of these state updates is performed, the current and the remaining slots n within the subframe are processed as previously described for states **A** or **B**, depending on which state was now set, and then the following operations are not performed.

Two further factors are formulated. In case of **A→B**

$$f_{switch}(n) = \sqrt{f_{interp}(n)}$$

$$f_{orig}(n) = \sqrt{1 - f_{interp}(n)}$$

and in case of **B→A**

$$f_{switch}(n) = \sqrt{1 - f_{interp}(n)}$$

$$f_{orig}(n) = \sqrt{f_{interp}(n)}$$

Then, mixing is performed by

$$\begin{bmatrix} S_{mix}(k,n,1) \\ S_{mix}(k,n,2) \end{bmatrix} = \begin{bmatrix} f_{orig}(n) & f_{switch}(n) \\ f_{switch}(n) & f_{orig}(n) \end{bmatrix} \begin{bmatrix} S(k,n,1) \\ S(k,n,2) \end{bmatrix}$$

Then, equalization gains for the channels $i = 1,2$ are formulated by

$$g_{eq}(k,n,i) = \min\left(4,\sqrt{\frac{|S(k,n,i)f_{orig}(n)|^2 + |S(k,n,2-i)f_{switch}(n)|^2}{|S_{mix}(k,n,1)|}}\right)$$

Finally, the transport audio signals are replaced with

$$S(k,n,i) \coloneqq S_{mix}(k,n,i)g_{eq}(k,n,i)$$

The above transition-state **A→B** or **B→A** procedures are performed for each slot n until the slots n of the subframe are exhausted.

7.2.2.3.3 Determination of input and target covariance matrices

A target covariance matrix contains (stochastic) properties that the output signal is targeted to attain, and it is determined in frequency bins. Considering a binaural or stereo output, a target covariance matrix contains the channel energies and the cross-correlations for the output. The cross-correlation amplitude determines the coherence between the output channels and the phase determines the phase-difference. The energies, phases, and coherences are controlled since they are key contributors to the spatial audio perception. For the renderer, there is always two output channels (binaural or stereo), and therefore the target covariance matrix is of shape 2x2 for each frequency bin.

Correspondingly, the input covariance matrix has the information of these energetic and correlation properties of the received transport audio channels. Any additional object nor centre channels are not considered in the input covariance matrix. Therefore, also the input covariance matrices are always of shape 2x2 for each frequency bin.

Many of the described operations are identical in case of binaural and stereo outputs. Any differences are specifically mentioned.

The input covariance matrix $\mathbf{C}_{in}(k,m)$ and subframe energy $E_{in}(k,m)$ are determined for the current subframe m as follows. First, the input signal is expressed in a column vector form

$$\mathbf{s}(k,n) = \begin{bmatrix} S(k,n,1) \\ S(k,n,2) \end{bmatrix}$$

Then,

$$\mathbf{C}_{in}(k,m) = \sum_{n=n_{first}(m)}^{n_{last}(m)} \mathbf{s}(k,n)\,\mathbf{s}^{H}(k,n)$$

$$E_{in}(k,m) = tr(\mathbf{C}_{in}(k,m))$$

where $n_{first}(m)$ and $n_{last}(m)$ are the first and last temporal slots of the current subframe m, and $tr\left(\mathbf{C}_{in}(k,m)\right)$ denotes the matrix trace of $\mathbf{C}_{in}(k,m)$. The subframe energy $E_{in}(k,m)$ is the overall energy of the transport audio signals.

In case the low-bitrate EQ is enabled (i.e., if the bitrate is 13.2 or 16.4 kbps and the IVAS format is MASA or multi-channel), the subframe energy $E_{in}(k,m)$ is modified using it

$$E_{in}(k,m) \coloneqq E_{in}(k,m)g_{lbr,EQ}(k)$$

where $g_{lbr,EQ}(k)$ is the low-bitrate EQ (see table 7.2-2, bins above 16 use the value of bin 16). If it is not enabled, the subframe energy modification is not performed.

Table 7.2-2: Low-bitrate EQ

Bin	0	1	2	3	4	5	6	7	8
Gain	0.979	0.893	0.762	0.615	0.52	0.48	0.477	0.477	0.48
Bin	9	10	11	12	13	14	15	16	
Gain	0.501	0.546	0.602	0.652	0.664	0.652	0.639	0.635	

In addition, if the IVAS format is SBA or OSBA and if the number of transport audio signals is 2, the subframe energy $E_{in}(k,m)$ is modified by

$$E_{in}(k,m) \coloneqq \max\left(\frac{E_{in}(k,m)}{2}, \sum_{n=n_{first}(m)}^{n_{last}(m)} |S(k,n,1)+S(k,n,2)|^2\right)$$

For the other formats, and for the other number of transport audio signals, this modification is not performed.

The target covariance matrices are described in the following, based on the spatial metadata and the subframe energy $E_{in}(k,m)$.

First, the mean energy per output channel $E_{meanOut}(k,m)$ is formulated by

$$E_{meanOut}(k,m) = 0.5 g_{early}(k)\, E_{in}(k,m)$$

where $g_{early}(k)$ is an early part energy correction gain for the binaural rendering. When the room effect is not enabled, $g_{early}(k) = 1$. When the room effect is enabled, the values for $g_{early}(k)$ are obtained from the data set in variable *parametricEarlyPartEneCorrection* in the reference C code.

The spatial metadata that the renderer obtains may have one or two simultaneous "parametric" directions (associated with the MASA, SBA, multichannel, or ISM streams), and from one to four simultaneous "object" directions (associated with the objects part of the OMASA stream). I.e., there may be from one to six simultaneous directions. Each direction is associated with an azimuth, elevation, direct-to-total energy ratio and spread coherence parameter. The number of simultaneous directions is denoted by D, and d refers to the direction index for $1 \le d \le D$.

Thus, in the following, the azimuth $\theta_d(k,m)$, elevation $\varphi_d(k,m)$, direct-to-total energy ratio $r_d(k,m)$, and spread coherence $\zeta_d(k,m)$ parameters refer to the parameters for the parametric directions ($\theta_i(k,m)$, $\varphi_i(k,m)$, $r_i(k,m)$, and $\zeta_i(k,m)$) and the parameters for the object directions ($\theta_{ISM,i}(m)$, $\varphi_{ISM,i}(m)$, $r_{ISM,i}(k,m)$, and $\zeta_{ISM,i}(m)$).

Before formulating the rendering gains, if operating in the *separateCenterChannelRendering* mode (of McMASA separate-channel sub-mode), the spread coherence is modified by

$$\zeta_d(k,m) := \max\left(\zeta_d(k,m), 1 - \frac{\cos^{-1}(\cos(\theta_d(k,m))\cos(\varphi_d(k,m)))}{30^\circ}\right)$$

to account for the missing centre channel in the analysis of the spread coherence in the encoder.

In processing the target covariance matrix, the next step is to determine the direct part gain vector $\mathbf{g}_{direct,d}(k,m)$, which is a 2x1 column vector. It is formulated based on the metadata azimuth $\theta_d(k,m)$ and elevation $\varphi_d(k,m)$ for each d, as described in clause 7.2.2.3.6, where it is also described how the spread coherence value $\zeta_d(k,m)$ is used in determining the gains. The gains $\mathbf{g}_{direct,d}(k,m)$ may be complex valued in case of a binaural output (due to needed phase differences in the binaural output) and are real valued for stereo output.

The direct part target covariance matrix is then

$$\mathbf{C}_{out,dir}(k,m) = E_{meanOut}(k,m) \sum_{d=1}^{D} \mathbf{g}'_{direct,d}(k,m)\mathbf{g}'^{H}_{direct,d}(k,m) r_d(k,m)$$

where $r_d(k,m)$ is the direct-to-total energy ratio of direction index d.

The non-direct part of the target covariance matrix is determined as follows. First, the diffuse-to-total energy ratio (also called diffuseness, in short) is formulated as

$$r_{diff}(k,m) = \max\left(0, \left(1 - \sum_{d=1}^{D} r_d(k,m)\right)\right)$$

Also, a diffuseness value for decorrelation reduction is formulated as

$$r_{diff,dec}(k,m) = \max\left(0, \left(1 - \sum_{d=1}^{D} (1 - 0.5 * ism(d)) r_d(k,m)\right)\right)$$

where ism(d) is a function that is 1 when the direction d corresponds to an ISM direction indicated in the spatial metadata, and 0 otherwise. The value $r_{diff}(k,m)$ therefore describes the energy of the non-directional sound energy, whereas $r_{diff,dec}(k,m)$ is a related control parameter that is later used to configure the level of decorrelated energy.

Next, the per-output-channel diffuse sound energy, and a related control parameter, are formulated by

$$E_{diff}(k,m) = r_{diff}(k,m)E_{meanOut}(k,m)g_{diffMod}(k,m)$$

$$E_{diff,dec}(k,m) = r_{diff,dec}(k,m)E_{meanOut}(k,m)g_{diffMod}(k,m)$$

where $g_{diffMod}(k)$ is an energy modifier value which is 1 otherwise, except in case of the output being binaural and the IVAS format being multichannel, in which case it has the following value

$$g_{diffMod}(k,m) = \big(1 - \gamma(k,m)\big) + \gamma(k,m)E_{\gamma}(k)$$

where $0 \leq \gamma(k,m) \leq 1$ is the surround coherence value and $E_{\gamma}(k)$ is a spectral modifier that affects the binaural spectrum of surround-coherent sound. (see Table 7.2-3, bins above 5 use the value of bin 5).

Table 7.2-3: Spectral modifier for surround-coherent sound

Bin	0	1	2	3	4	5
Gain	3.0903	2.0053	1.0860	0.8072	0.7079	1

In case the output is stereo (not binaural), the non-direct part of the target covariance matrix then is

$$\mathbf{C}_{out,nonDir}(k,m) = (\gamma(k,m) + (1 - \gamma(k,m)\mathbf{I})E_{diff}(k,m)$$

where **I** denotes a 2x2 identity matrix.

In case the output is binaural, the non-direct part of the target covariance matrix is

$$\mathbf{C}_{out,nonDir}(k,m) = \big(\gamma(k,m) + (1 - \gamma(k,m)c_{diffBin}(k)\mathbf{I}\big)E_{diff}(k,m)$$

where $c_{diffBin}(k)$ is the binaural diffuse field coherence.

If the IVAS format is not SBA nor OSBA, the binaural diffuse field coherence $c_{diffBin}(k,m)$ is fixed (i.e., $c_{diffBin}(k,m) = c_{diffBin}(k)$) and can be determined by

$$c_{diffBin}(k) = \max\left(0, \left(1 - \frac{f_{center}(k)}{2700Hz}\right)\right)\frac{\sin(\pi f_{center}(k)/550Hz)}{\pi f_{center}(k)/550Hz}$$

where $f_{center}(k)$ is the centre frequency of k:th bin in Hz, i.e., $f_{center}(k) = (k + 0.5)400Hz$, when the first bin index is $k = 0$.

If the IVAS format is SBA or OSBA, the binaural diffuse field coherence rendering information is determined based on the directional distribution information for the indirect audio (the diffuse ratios in x, y, and z directions)

$$c_{diffBin}(k,m) = \varsigma_x(k,m)c_{diffBin,x}(k) + \varsigma_y(k,m)c_{diffBin,y}(k) + \varsigma_z(k,m)c_{diffBin,z}(k)$$

where $\varsigma_{x/y/z}(k,m)$ is the diffuse ratio in x/y/z-directions (see clause 7.2.2.3.9) and $c_{diffBin,x/y/z}(k)$ is a binaural diffuse field coherence in the x/y/z-directions. The determined binaural diffuse field coherence enables rendering binaural audio with the correct directional distribution of the indirect audio.

The binaural diffuse field coherence in the x/y/z-directions $c_{diffBin,x/y/z}(k)$ is determined using $c_{diffBin}(k)$ and the binaural diffuse field coherence differences in x/y/z-directions $c_{diffBin,x/y/z,diff}(k)$ by

$$c_{diffBin,x}(k) = c_{diffBin}(k) + c_{diffBin,x,diff}(k)$$

$$c_{diffBin,y}(k) = c_{diffBin}(k) + c_{diffBin,y,diff}(k)$$

$$c_{diffBin,z}(k) = c_{diffBin}(k) + c_{diffBin,z,diff}(k)$$

where the values of the binaural diffuse field coherence differences in x/y/z-directions $c_{diffBin,x/y/z,diff}(k)$ are shown in Table 7.2-4, 7.2-5, and 7.2-6 (bins above 9 use the value of bin 9).

Table 7.2-4: Binaural diffuse field coherence differences in x-direction $c_{diffBin,x,diff}$

Bin	0	1	2	3	4
Value	0.047421	0.19773	0.22582	0.10637	0.0087111
Bin	5	6	7	8	9
Value	0.012028	0.031972	0.019668	0.0079928	0

Table 7.2-5: Binaural diffuse field coherence differences in y-direction $c_{diffBin,y,diff}$

Bin	0	1	2	3	4
Value	-0.095628	-0.30569	-0.34427	-0.15425	-0.044628
Bin	5	6	7	8	9
Value	-0.057224	-0.050835	-0.035214	-0.02215	0

Table 7.2-6: Binaural diffuse field coherence differences in z-direction $c_{diffBin,z,diff}$

Bin	0	1	2	3	4
Value	0.048207	0.10796	0.11845	0.047886	0.035917
Bin	5	6	7	8	9
Value	0.045196	0.018863	0.015547	0.014157	0

Then, the target covariance matrix is formed by combining the target covariance matrices for the direct and the non-direct parts

$$\mathbf{C}_{out}(k,m) = \mathbf{C}_{out,dir}(k,m) + \mathbf{C}_{out,nonDir}(k,m)$$

The above processing was for the current subframe. Next, the covariance matrices are temporally smoothed. An averaging value is formulated by

$$\alpha_{IIR}(m) = \beta_{IIR} + (1 - q(n))$$

where $\beta_{IIR} = 16$ if the IVAS format is MASA and the bitrate is smaller than 24.4 kbps and $\beta_{IIR} = 8$ otherwise. $q(n)$ is the encoding quality based smoothing factor (i.e., a smoothing control factor), which is determined by

$$q(n) = \nu^2$$

where ν is an encoding metric indicating the quality of encoding (determined in clause 6.5.3.8). The determined averaging value $\alpha_{IIR}(m)$ is then used for temporally smoothing the covariance matrices. Thus, the spatial audio signals are generated from the transport audio signals based on the spatial metadata and the encoding metric.

Furthermore, a temporal energy ratio parameter is determined

$$E_{in,r}(k,m) = \frac{tr\left(\mathbf{C}_{in}(k,m)\right)}{tr(\mathbf{C}_{in}(k,m-1))}$$

Then an IIR energy limiter factor is determined

$$\alpha_{lim}(k,m) = \min\left(1, E_{in,r}(k,m)\alpha_{IIR}(m)\right)$$

The temporally smoothed covariance matrices are obtained by

$$\mathbf{C'}_{in}(k,m) = q(n)\mathbf{C}_{in}(k,m) + \alpha_{lim}(k,m)\mathbf{C'}_{in}(k,m-1)$$

$$\mathbf{C'}_{out}(k,m) = q(n)\mathbf{C}_{out}(k,m) + \alpha_{lim}(k,m)\mathbf{C'}_{out}(k,m-1)$$

where $\mathbf{C'}_{in}(k,0) = \mathbf{C'}_{out}(k,0) = \mathbf{0}$ for the first subframe of the first frame.

The matrices $\mathbf{C'}_{in}(k,m)$ and $\mathbf{C'}_{out}(k,m)$ are then the formulated and temporally averaged input and target covariance matrices.

7.2.2.3.4 Determining processing matrices based on input and target covariance matrices

In this clause, processing parameters called processing matrices are determined based on the input and target covariance matrices. Using these processing matrices and the transport audio signals, the binaural or stereo audio signals are generated in clause 7.2.2.3.5.

First, the output covariance matrix is Cholesky decomposed to $\mathbf{K}_{out}(k,m)$ so that

$$\mathbf{C'}_{out}(k,m) = \mathbf{K}_{out}(k,m)\mathbf{K}_{out}^{H}(k,m)$$

Then, the input covariance matrix is Eigen decomposed to $\mathbf{U}_{in}(k,m)$ and $\mathbf{S}_{in}(k,m)$ so that

$$\mathbf{C'}_{in}(k,m) = \mathbf{U}_{in}(k,m)\mathbf{S}_{in}(k,m)\mathbf{U}_{in}^{H}(k,m)$$

Then, a matrix is formulated $\mathbf{K}_{in}(k,m) = \mathbf{U}_{in}(k,m)\mathbf{S}_{in,sq}(k,m)$ where $\mathbf{S}_{in,sq}(k,m)$ is a diagonal matrix with the diagonal entries that are the square root of $\mathbf{S}_{in}(k,m)$.

Then, $\mathbf{S}_{in,sq}(k,m)$ is modified using a control parameter called the regularization factor. This is done by modifying the entries of $\mathbf{S}_{in,sq}(k,m)$ so that its diagonal values are not smaller than the regularization factor f_{reg} times the maximum diagonal value of $\mathbf{S}_{in,sq}(k,m)$, resulting in a modified $\mathbf{S}_{in,sq}(k,m)$ which is used instead of the original $\mathbf{S}_{in,sq}(k,m)$ (the same term is used here for simplicity). The regularization factor f_{reg} is determined as presented in clause 7.2.2.3.7.

Then, a matrix $\mathbf{A}(k,m)$ is formulated by

$$\mathbf{A}(k,m) = \mathbf{K}_{out}^{H}(k,m)\widehat{\boldsymbol{G}}(k,m)\mathbf{Q}\mathbf{K}_{in}(k,m)$$

where $\widehat{\boldsymbol{G}}(k,m)$ is a diagonal matrix where the diagonal entries are the diagonal entries of $\mathbf{C'}_{out}(k,m)$ divided by the diagonal entries of $\mathbf{C'}_{in}(k,m)$ and $\mathbf{Q} = \begin{bmatrix} 1 & 0.05 \\ 0.05 & 1 \end{bmatrix}$.

Then, a matrix $\mathbf{P}(k,m)$ is determined that is the nearest orthonormal matrix to $\mathbf{A}(k,m)$ by

$$\boldsymbol{P}(k,m) = \mathbf{A}(k,m)\left(\mathbf{A}^{H}(k,m)\mathbf{A}(k,m)\right)^{-\frac{1}{2}}$$

Then, a processing matrix is formulated by

$$\boldsymbol{M}(k,m) = \boldsymbol{K}_{out}(k,m)\boldsymbol{P}(k,m)\mathbf{S}_{in,sq}^{-1}(k,m)\boldsymbol{U}_{in}^{H}(k,m)$$

The effect of applying the processing matrix to the input signals is formulated by

$$\widehat{\boldsymbol{C}}'_{out}(k,m) = \boldsymbol{M}(k,m)\boldsymbol{C}'_{in}(k,m)\boldsymbol{M}^{H}(k,m)$$

In case k is smaller than the maximum decorrelation frequency bin K_{maxdec} (see clause 7.2.2.3.8), a residual covariance matrix $\boldsymbol{C}'_{res}(k,m)$ is then determined as follows, and otherwise $\boldsymbol{C}'_{res}(k,m) = \mathbf{0}$.

$$\boldsymbol{C}'_{res}(k,m) = \left(\boldsymbol{C}'_{out}(k,m) - \widehat{\boldsymbol{C}}'_{out}(k,m)\right)f_{dec}(k,m)$$

where $f_{dec}(k,m)$ is a decorrelation reduction factor, which is determined by the following rules:

- If the IVAS format is MASA_FORMAT, and the IVAS total bit rate is less than 24.4 kbps, then

$$f_{dec}(k,m) = 1$$

- Otherwise, if the IVAS format is MC_FORMAT or the IVAS format is MASA_FORMAT and the number of transport channels (prior creating dual-mono) is 1, then

$$f_{dec}(k,m) = \sqrt{\frac{E_{diff,dec}(k,m)}{E_{meanOut}(k,m)}}$$

- Otherwise, if the IVAS format is SBA_FORMAT or SBA_ISM_FORMAT, and the number of transport channels (prior creating dual-mono) is 1, then

$$f_{dec}(k,m) = 1$$

- Otherwise

$$f_{dec}(k,m) = \frac{E_{diff,dec}(k,m)}{E_{meanOut}(k,m)}$$

The residual covariance matrix $\boldsymbol{C'}_{res}(k,m)$ can be seen as a control parameter configured to control the amount of decorrelated audio signal within the two output audio signals for spatial audio reproduction, in this case, for stereo or binaural audio reproduction. It is based on the input and target covariance matrices so that necessary amount of decorrelation is provided to the output audio signals so that the target spatial properties (the target covariance matrix) according to the spatial metadata are obtained. The residual covariance matrix $\boldsymbol{C'}_{res}(k,m)$ is further controlled by the spatial metadata to suppress the decorrelation when needed to minimise the decorrelation artefacts.

Then, a residual mixing matrix $\boldsymbol{M}_{res}(k,m)$ is formulated. It is formulated by the same steps as described in the foregoing for determining $\boldsymbol{M}(k,m)$, however, with the following differences: In place of the input covariance matrix $\boldsymbol{C'}_{in}(k,m)$, the same matrix but where all non-diagonal entries are zeroed is used; in place of $\boldsymbol{C'}_{out}(k,m)$ the matrix $\boldsymbol{C'}_{res}(k,m)$ is used; and the regularization factor is set to 0.2. If $\boldsymbol{C'}_{res}(k,m) = \boldsymbol{0}$ then $\boldsymbol{M}_{res}(k,m) = \boldsymbol{0}$.

Then, the missing energy due to potential regularizations is formulated by

$$\boldsymbol{E}_{miss}(k,m) = \max\left(0, tr\left(\boldsymbol{C'}_{out}(k,m)\right) - tr\left(\boldsymbol{C'}_{res}(k,m) + \widehat{\boldsymbol{C}}'_{out}(k,m)\right)\right)$$

And, an equalization gain to compensate for the missing energy is formulated by

$$g_{miss}(k,m) = \min\left(4, \sqrt{\frac{tr\left(\boldsymbol{C'}_{res}(k,m)\right) + \boldsymbol{E}_{miss}(k,m)}{tr\left(\boldsymbol{C'}_{res}(k,m)\right)}}\right)$$

Then, the processing matrix is gain-compensated by

$$\boldsymbol{M'}(k,m) = \boldsymbol{M}(k,m) g_{miss}(k,m)$$

Furthermore, if the stream contains a separate center channel (when operating in the multi-channel format) or if the stream contains separated object channels (when operating in the OMASA format), 2x1 processing gain vectors $\boldsymbol{m}'_i(k,m)$ are determined, where i is the separate channel/object index. These are formulated by first using the method in clause 7.2.2.3.6 to obtain gains $\mathbf{g}_{direct,sep,i}(k,m)$, and then processing them as $\boldsymbol{m}'_i(k,m) = \mathbf{g}_{direct,sep,i}(k,m) g_i$, where g_i is a gain value that is 0.7943 when the IVAS format is MASA_ISM_FORMAT and 0.8414 otherwise.

7.2.2.3.5 Processing audio signals with the processing matrices

The binaural (or stereo) audio output is produced by generating different binaural audio signal parts (direct input signal part, decorrelated input signal part, separated centre/object channel part, and room-effect part) and combining the different binaural audio signal parts to generate a combined binaural audio signal (which is the output of the processing)

$$\mathbf{s}_{\mathrm{out}}(k,n) = \mathbf{s}_{\mathrm{out},1}(k,n) + \mathbf{s}_{\mathrm{out},2}(k,n) + \mathbf{s}_{\mathrm{out},3}(k,n) + \mathbf{s}_{\mathrm{out},4}(k,n)$$

where

$$\mathbf{s}_{\mathrm{out},1}(k,n) = (\boldsymbol{M'}(k,m)p(m,n) + \boldsymbol{M'}(k,m-1)(1-p(m,n)))\mathbf{s}(k,n)$$

$$\mathbf{s}_{\mathrm{out},2}(k,n) = (\boldsymbol{M}_{res}(k,m)p(m,n) + \boldsymbol{M}_{res}(k,m-1)(1-p(m,n)))\mathbf{s}_{\mathrm{decorr}}(k,n)$$

$$\mathbf{s}_{\mathrm{out},3}(k,n) = \sum_i (\boldsymbol{m'}_i(k,m)p(m,n) + \boldsymbol{m'}_i(k,m-1)(1-p(m,n)))\mathbf{s}_{sep}(k,n)$$

$$\mathbf{s}_{\mathrm{out},4}(k,n) = \mathrm{reverb}(\mathbf{s}(k,n))$$

where $\mathbf{s}(k,n)$ is the transport audio signals, $\mathbf{s}_{sep}(k,n)$ is the one or more separate channel/object signals (if there is none of them, it is set to zero), $\mathbf{s}_{\mathrm{decorr}}(k,n)$ is a decorrelated version of $\mathbf{s}(k,n)$, $\mathrm{reverb}()$ denotes applying a room effect using a reverberator, the processing coefficients in $\boldsymbol{M'}$, $\boldsymbol{M}_{res}$ and $\boldsymbol{m'}_i$ are based on the spatial metadata as described in the foregoing clauses, and $p(m,n)$ is an interpolation value determined by

$$p(m,n) = \frac{(n - n_{first}(m) + 1)}{(n_{last}(m) - n_{first}(m) + 1)}$$

As an exception, $p(m,n) = 1$ when the IVAS format is OMASA, and the ISM mode is ISM_MASA_MODE_MASA_ONE_OBJ or ISM_MASA_MODE_PARAM_ONE_OBJ. Otherwise $p(m,n)$ is determined using the equation above.

The decorrelated audio signals $\mathbf{s}_{\mathrm{decorr}}(k,n)$ are generated by processing the transport audio signal(s) $\mathbf{s}(k,n)$ as described in clause 7.2.2.3.8.

$\mathbf{s}_{\mathrm{out},4}(k,n)$ is generated by processing the input signals with a room effect when the room effect indication says to apply it. Applying the room effect is described in clause 7.3.3. The reverberator is initialized based on the late reverberation energy correction gains and reverberation times obtained from the data set in variables *parametricReverberationEneCorrections* and *parametricReverberationTimes* in the reference C code. When the room effect indication says to not apply a room effect, $\mathbf{s}_{\mathrm{out},4}(k,n)$ is set to zero.

The time-frequency domain binaural (or stereo) audio signal $\mathbf{s}_{\mathrm{out}}(k,n)$ is converted to the time domain via the inverse CLDFB (see clause 6.2.5 for details), yielding $\mathbf{s}_{\mathrm{out,td}}(n)$.

7.2.2.3.6 Determining direct part gains

In the following, it is described how direct part gains are formulated for binaural or stereo output for a given direction. For determining the gains, the direction parameter is denoted azimuth θ and elevation φ, and the output gains are denoted as a 2x1 column vector **g**. In this clause, the time and frequency indices, and most of the subscripts, are omitted for clarity of the equations.

First, the determining of stereo gains is described. In that operating mode, the azimuth and elevation are first mapped to a horizontal azimuth by

$$\theta_{map} = arctan2\left(y, \sqrt{1 - y^2}\right)$$

where

$$y = \sin(\theta)\cos(\varphi)$$

When azimuth is between -$\theta_{ref} < \theta_{map} < \theta_{ref}$, where θ_{ref} is 30 degrees, then the panning gains are formulated by

$$a_1 = \frac{\tan(\theta_{map})}{\tan(\theta_{ref})}$$

$$a_2 = \frac{a_1 - 1}{\max(0.001, a_1 + 1)}$$

$$a_3 = \frac{1}{a_2^2 + 1}$$

$$\mathbf{g}(k,m) = \begin{bmatrix} \sqrt{a_3} \\ \sqrt{1 - a_3} \end{bmatrix}$$

When $\theta_{map} \geq \theta_{ref}$, then $\mathbf{g} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. When $\theta_{map} \leq -\theta_{ref}$, then $\mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

In case the operating mode is providing binaural output, the above gains are not used, but the following formulas are used instead.

If the head orientation is tracked, the direction parameters azimuth θ and elevation φ are converted to rotated direction parameters. The rotation is performed by converting the azimuth θ and elevation φ values to a 3x1 unit vector $\mathbf{u}_{\theta,\varphi}$ pointing towards the azimuth and elevation, and then rotating it with a 3x3 rotation matrix **R** (see $R_{com}(m)$ in clause 7.4.6.1) so that a rotated unit vector is $\mathbf{u}_{\theta,\varphi,rot} = \mathbf{R}\mathbf{u}_{\theta,\varphi}$. If head orientation data is not available or not used, then $\mathbf{u}_{\theta,\varphi,rot} = \mathbf{u}_{\theta,\varphi}$.

As a second step, third order spherical harmonic 16x1 gain vector $\mathbf{g}_{HOA3,\theta,\varphi,rot}$, which corresponds to the direction of $\mathbf{u}_{\theta,\varphi,rot}$, is determined using the ACN channel order (see clause 6.4.6.5.1) and the SN3D normalization. The binaural gains are then obtained by

$$\mathbf{g} = \boldsymbol{M}_{HOA3bin}\mathbf{g}_{HOA3,\theta,\varphi,rot}$$

where $\boldsymbol{M}_{HOA3bin}$ is a 2x16 spherical-harmonic-to-binaural processing matrix (see variables *hrtfShCoeffsRe* and *hrtfShCoeffsIm* in the reference C code). Note that $\boldsymbol{M}_{HOA3bin}$ is dependent on the frequency bin k. The determined gains **g** are effectively head-related transfer functions (HRTFs) and are typically complex-valued at least in low/mid frequencies. Since the same system is used for rendering binaural and stereo, the vector **g** values are not referred to as HRTFs but more generally as "gains".

In case the rendering output is binaural (i.e., not stereo), the gains **g** are further affected by the spread coherence value ζ associated with the current azimuth θ and elevation φ. If the output is stereo, or if $\zeta = 0$ the spread-coherence-related processing in the following is not applied.

A spread coherent sound refers to directional sound that, instead of being a point source, originates coherently from more than one direction. For example, amplitude panned sound wound constitute a "spread coherent" sound. The spread coherence is expressed by a spread coherence parameter ζ ranging from 0 to 1, where $\zeta = 0$ refers to a point-source, $\zeta = 0.5$ refers to three sources at 30 degrees spacing (i.e., spanning 60 degrees in total), and $\zeta = 1$ refers to two sources at 60-degree spanning.

First, the binaural (i.e., not stereo) gain is formulated for the given azimuth θ and elevation φ. The result is the centre gains $\mathbf{g}_c$. Next, direct part gains are also formulated for directions $(\theta + 30°, \varphi)$ and $(\theta - 30°, \varphi)$. The results are the gains $\mathbf{g}_{\leftarrow}$ and $\mathbf{g}_{\rightarrow}$. The procedure of formulating these gains is as described in the foregoing, for the given angles with three different azimuth values.

Next, centre multiplier g_c and side-multiplier g_s are formulated as follows. If the spread coherence value $\zeta < 0.5$, then

$$g_s = \frac{2\zeta}{\sqrt{3}}$$

$$g_c = 1 - (2\zeta) + g_s$$

Otherwise, i.e., if $\zeta \geq 0.5$, then

$$g_{tmp} = 2 - (2\zeta)$$

$$g_s = \frac{1}{\sqrt{g_{tmp} + 2}}$$

$$g_c = g_{tmp} g_s$$

The combined spread-coherent binaural gains, prior to normalization, are formulated by

$$\mathbf{g}_{nonEq} = g_c \mathbf{g}_c + g_s(\mathbf{g}_{\leftarrow} + \mathbf{g}_{\rightarrow})$$

Then, an energy-correcting factor is formulated by

$$g_{Eq1} = \frac{\boldsymbol{g}_s^2(\mathbf{g}_{\leftarrow}^H\mathbf{g}_{\leftarrow} + \mathbf{g}_{\rightarrow}^H\mathbf{g}_{\rightarrow}) + \boldsymbol{g}_c^2\mathbf{g}_c^H\mathbf{g}_c}{\mathbf{g}_{nonEq}^H\mathbf{g}_{nonEq}}$$

Then, a set of weights are determined which, when $\zeta < 0.5$, are

$$w_1 = 1 - 2\zeta$$

$$w_2 = 2\zeta$$

$$w_3 = 0$$

Otherwise, i.e., when $\zeta \geq 0.5$

$$w_1 = 0$$

$$w_2 = 2 - 2\zeta$$

$$w_3 = 2\zeta - 1$$

The second energy correction factor is $g_{Eq2} = g_{Eq1}$ for all other formats than the multi-channel format. With the multi-channel format, it is determined as follows: if the separate centre channel rendering is enabled, then

$$g_{Eq2} = g_{Eq1}\big(w_1 + (w_2 + w_3)\big)E_\zeta$$

and if the separate centre channel rendering is not enabled, then

$$g_{Eq2} = g_{Eq1}\big(w_1 + w_2 E_{\zeta 05} + w_3 E_{\zeta 1}\big)$$

where $E_{\zeta 05}$ and $E_{\zeta 1}$ are frequency-dependent spread coherence energy correcting factors (see Table 7.2-7 and 7.2-8, bins above 5 use the value of bin 5).

Table 7.2-7: Spread coherence energy correcting factors $E_{\zeta 05}$

Bin	0	1	2	3	4	5
Gain	2.3988	1.7783	1.1220	1.1220	1.1220	1

Table 7.2-8: Spread coherence energy correcting factors $E_{\zeta 1}$

Bin	0	1	2	3	4	5
Gain	1.5975	1.1220	1.1220	1.1220	1.1220	1

The gains $\mathbf{g}_{nonEq}$ are then equalized with the energy correction factors to determine the binaural panning gains by

$$\mathbf{g} = \mathbf{g}_{nonEq} \min\big(4, \sqrt{g_{Eq2}}\big)$$

These are then the gain values for the azimuth θ and elevation φ, when the output is binaural and the spread coherence $\zeta > 0$.

7.2.2.3.7 Determining regularization factor

The regularization factor f_{reg} is a control parameter that is determined based on obtained input property parameters associated with the input spatial audio signal. These input property parameters are the bitrate of the input spatial audio signal and the codec format (IVAS format) indicating the origin of the input spatial audio signal. In clause 7.2.2.3.4 the determination of the processing parameters (the processing matrices) is then controlled based on the control parameter f_{reg}, and in clause 7.2.2.3.5 the audio signals are then generated using these processing matrices. The control parameter f_{reg} affects the determination of the processing parameters such that values closer to 1 entail higher amount of regularization, which means that the resulting amplification of small independent signal components is more limited.

For the IVAS formats SBA, OSBA, OMASA, and ISM, the regularization factor is $f_{reg} = 1$.

For the MASA format, the regularization factor is determined based on the bitrate BR:

- if $BR \geq 160$ kbps, $f_{reg} = 0.4$
- else if $BR \geq 128$ kbps, $f_{reg} = 0.5$
- else if $BR \geq 96$ kbps, $f_{reg} = 0.6$
- else if $BR \geq 64$ kbps, $f_{reg} = 0.8$
- else, $f_{reg} = 1$

For the multi-channel format, the regularization factor is determined based on the bitrate BR:

- if $BR \geq 96$ kbps, $f_{reg} = 0.4$

- else if $BR \geq 80$ kbps, $f_{reg} = 0.5$
- else if $BR \geq 64$ kbps, $f_{reg} = 0.7$
- else if $BR \geq 48$ kbps, $f_{reg} = 0.8$
- else, $f_{reg} = 1$

7.2.2.3.8 Decorrelation in the parametric binauralizer

The parametric binauralizer uses two different decorrelators: the time-domain and the frequency domain decorrelators. The time-domain decorrelator is used in these cases:

- If the IVAS format is MASA, and the bitrate is smaller than 48 kbps in case of one transport audio signal or the bitrate is smaller than 24.4 kbps in case of two transport audio signals
- If the IVAS format is multi-channel, and the bitrate is smaller than 48 kbps
- If the IVAS format is SBA or OSBA, and the number of transport audio signals is one.

In other cases, the frequency-domain decorrelator is used.

The operation of the time-domain decorrelator is presented in clause 6.2.1.3.

The operation of the frequency-domain decorrelator is presented in clause 6.2.6.

The decorrelation is applied on all frequencies with the time-domain decorrelator (i.e., $K_{maxdec} = 60$). The decorrelation is applied on frequencies below 6 kHz with the frequency-domain decorrelator (i.e., $K_{maxdec} = 15$). No decorrelation is applied when the IVAS format is ISM_FORMAT (i.e., $K_{maxdec} = 0$).

7.2.2.3.9 SPAR metadata to MASA metadata mapping

This clause presents methods for converting first spatial metadata (SPAR spatial metadata), which enables the rendering of spatial audio in a first audio format (first-order Ambisonic (FOA) audio signals), to second spatial metadata (MASA spatial metadata), which enables the rendering of spatial audio in a second audio format (binaural audio signals). I.e., the SPAR spatial metadata is configured to enable the rendering of FOA spatial audio signals using the transport audio signal(s). To enable the rendering of binaural spatial audio signals from the transport audio signal(s), the MASA spatial metadata is determined using the SPAR spatial metadata, and this MASA metadata is then used at the binaural rendering. The MASA metadata is determined directly from the SPAR metadata without rendering intermediate audio signals in between.

The mapping of the SPAR metadata to MASA metadata is performed for the SPAR frequency bands (there are B_{SPAR} SPAR frequency bands). The following processing is performed separately for each subframe m.

First, rendering information called the mixing matrix (or, in other words, the upmix matrix) is determined based on the SPAR metadata, as described in clauses 6.4.5 and 6.4.6.4.4. Then, the mixing matrix is determined for each frequency bin k based on the mixing matrix for each frequency band b (that was determined from the SPAR metadata) and the frequency ranges of the SPAR metadata frequency bands. The mixing matrices are delayed by two subframes to have them synced with the audio. The result is the mixing matrix $\mathbf{M}(k)$, where the individual entries are referred to by $M(k, i, j)$. The mixing matrix describes how to obtain from 4 input channels 4 output FOA channels. In case there are less than 4 transport audio signals (as is always the case when performing this mapping), the missing channels are replaced by the decorrelated versions of the first input channel.

Then, if the number of transport audio signals is 2 (i.e., $N_{trans} = 2$), the transport signal energies $E_{trans}(k, i)$ and the cross-correlations $c_{trans}(k)$ are determined by

$$E_{trans}(k, i) = \sum_{n=0}^{3} |S(k, n, i)|^2$$

$$c_{trans}(k) = \sum_{n=0}^{3} \mathrm{Re}\{S(k, n, 0)S^*(k, n, 1)\}$$

where n is the slot index of the CLDFB transformed time-frequency domain signal $S(k, n, i)$ corresponding to the slots of the current subframe, and where Re{} denotes taking the real part and $*$ denotes the complex conjugate.

The following processing takes place separately for each frequency bin, so the bin index k is omitted for having more clarity. Thus, all the output values are determined for each frequency bin, even though the frequency bins are not shown in the following equations.

First, the input covariance matrix $\mathbf{C}_{in}$ is determined. The individual entries are referred to by $c_{in}(i, j)$. In case of one transport channel, the diagonal elements are set to one

$$c_{in}(i, i) = 1$$

and the non-diagonal elements are set to zero

$$c_{in}(i, j) = 0, \quad i \neq j$$

In case of two transport channels, the first two diagonal elements are determined by

$$c_{in}(i, i) = E_{trans}(i), \quad i \leq 1$$

and the remaining diagonal elements by

$$c_{in}(i, i) = E_{trans}(0), \quad i > 1$$

and the cross-correlations between the first two channels are determined by

$$c_{in}(0,1) = c_{in}(1,0) = c_{trans}$$

and the remaining non-diagonal elements by

$$c_{in}(i, j) = 0, \quad else$$

Then, the FOA covariance matrix is determined by

$$\mathbf{C}_{FOA} = \mathbf{M}\mathbf{C}_{in}\mathbf{M}^T$$

The individual entries are referred to by $c_{FOA}(i, j)$. The MASA metadata is then determined using the FOA covariance matrix. First, the intensity is determined by

$$I_y = c_{FOA}(0,1)$$

$$I_z = c_{FOA}(0,2)$$

$$I_x = c_{FOA}(0,3)$$

Then, the length of the intensity vector is determined by

$$I_{len} = \sqrt{I_y^2 + I_z^2 + I_x^2}$$

The overall energy is determined by

$$E_{tot} = \frac{c_{FOA}(0,0) + c_{FOA}(1,1) + c_{FOA}(2,2) + c_{FOA}(3,3)}{2}$$

The azimuth angle is then determined by

$$\theta = \arctan(I_y, I_x)$$

and the elevation angle by

$$\phi = \arctan(I_z^2, \sqrt{I_x^2 + I_y^2})$$

where $\arctan(,)$ is a variant of arcus tangent that can determine the right quadrant.

The direct-to-total energy ratio is determined by

$$r_{dir} = \frac{I_{len}}{E_{tot}}$$

The direct-to-total energy ratios are limited to values between 0 and 1 to account for possible numerical inaccuracies.

The diffuse-to-total energy ratios are determined by

$$r_{diff} = 1 - r_{dir}$$

The spread and surround coherences are set to zero

$$\zeta = 0$$

$$\gamma = 0$$

The determined MASA spatial metadata is forwarded to the parametric binaural renderer, enabling the rendering of the binaural audio signals using the determined MASA spatial metadata and the transport audio signals.

The spatial audio comprises direct audio and indirect (e.g., non-directional and diffuse) audio. Above, the directional distribution of the direct audio in the sound scene was determined as azimuth and elevation angles. Then, the directional distribution information of the indirect audio in the sound scene is determined based on the SPAR spatial metadata (based on the mixing matrices determined from the SPAR spatial metadata). In the case of two transport audio signals, also the transport audio signals are used in the determination.

First, the diffuse gains are determined. In case of one transport channel, they are determined by

$$g_{diff,y} = |M(1,1)|$$

$$g_{diff,x} = |M(3,2)|$$

$$g_{diff,z} = |M(2,3)|$$

In case of two transport channels, they are determined by

$$g_{diff,y} = |M(1,1)E_{trans}(1)|$$

$$g_{diff,x} = |M(3,2)E_{trans}(0)| + |M(3,1)E_{trans}(1)|$$

$$g_{diff,z} = |M(2,3)E_{trans}(0)| + |M(2,1)E_{trans}(1)|$$

Then, the directional distribution information of the indirect audio is determined as diffuse ratios ς for the X, Y, and Z directions

$$\varsigma_y = \frac{g_{diff,y}}{g_{diff,y} + g_{diff,x} + g_{diff,z}}$$

$$\varsigma_x = \frac{g_{diff,x}}{g_{diff,y} + g_{diff,x} + g_{diff,z}}$$

$$\varsigma_z = \frac{g_{diff,z}}{g_{diff,y} + g_{diff,x} + g_{diff,z}}$$

In case all the diffuse gains are zeros, the diffuse ratios are set to 1/3 to avoid division by zero.

Then, in clause 7.2.2.3.3, rendering information (comprising a target covariance matrix of the audio signals) is determined corresponding to the directional distribution information of the indirect audio (i.e., the diffuse ratios), and spatial audio (binaural audio signals) is rendered using the determined rendering information, the transport audio signals, and the associated spatial metadata.

7.2.2.3.10 Prototype signal generation with SBA format input

In case of one transport audio signal, the W signal $S_W(k,n)$ is generated from the transport audio signals by applying the mixing matrix $\mathbf{M}(k)$ (see clause 7.2.2.3.9) on the transport audio signal. The W signal is duplicated, and the resulting two signals are used as the prototype audio signals.

In case of two transport audio signals, the W and Y signals $S_W(k,n)$ and $S_Y(k,n)$ are generated from the transport audio signals by applying the mixing matrix $\mathbf{M}(k)$ (see clause 7.2.2.3.9) on the transport audio signals. Then, cardioid signals pointing to ±90 degrees are generated by

$$S_{\mathrm{card}}(k,n,0) = 0.5\big(S_W(k,n) + S_Y(k,n)\big)$$

$$S_{\mathrm{card}}(k,n,1) = 0.5\big(S_W(k,n) - S_Y(k,n)\big)$$

The generated cardioid signals $S_{\mathrm{card}}(k,n,i)$ are used as the prototype audio signals.

7.2.2.4 Fast convolution binaural renderer

7.2.2.4.1 CLDFB-domain convolution

The fast convolution (FastConv) binaural renderer performs rendering from ambisonics signals of order up to 3 or multi-channel signals with the configurations in Table 7.2-9 (cf. clause 4.3.2). The rendering is achieved directly in the CLDFB domain. The main advantages of this method are that no additional filter bank synthesis and analysis are required when the output signals are already present in this domain, and the computational complexity is lower than that of a time-domain convolution.

Table 7.2-9: Loudspeaker channel configurations supported by the FastConv renderer.

configuration	number of channels (without LFE)
CICP6	5
CICP12	7
CICP14	7
CICP16	9
CICP19	11

For all supported configurations, the HRIR filters are stored in the CLDFB domain in fixed ROM tables in the reference code. The tables contain the filter taps for each CLDFB band. To reduce the computational complexity, the bandwidth of the filters is limited to 50 CLDFB bands (20 kHz), which has a negligible perceptual impact. The length of the filters is $N_{\mathrm{taps}} = 3$ taps for SBA and MC input signals with HRIRs. For the case of binaural rendering with a room effect (BRIRs), the length is $N_{\mathrm{taps}} = 96$ taps.

Let $h_i^{cldfb,L/R}(l,k)$ denote the HRIR or BRIR filter taps with the tap index l, the CLDFB band index k , for the left and right ear, and let $S_i(m,k)$ denote the ith channel of the ambisonics or multichannel signal to be binauralized with the time-slot index m. The FastConv binaural renderer then calculates the convolution of the signal with the filter in each CLDFB band as

$$S^{cldfb,L/R}(n,k) = \sum_{i=1}^{N_{\mathrm{inp}}} \sum_{l=0}^{N_{\mathrm{taps}}-1} S_i\big(n - N_{\mathrm{taps}} + 1 + l, k\big) * h_i^{cldfb,L/R}(l,k)$$

N_{inp} is the number of channels of the renderer input signal $\boldsymbol{S}(m,k)$. It is given by $N_{\mathrm{inp}} = (L+1)^2$ for ambisonics inputs of order L and according to Table 7.2-1 for multichannel input signals. Note that the time-slot index n refers to the whole signal here rather than a frame or subframe. $S^{cldfb,L/R}(n,k)$ is the binaural output signal in the CLDFB domain. It is synthesized to the time domain according to clause 6.2.5.2. It can then be played back on headphones.

7.2.2.4.2 Filter conversion

7.2.2.4.2.1 General

For the shorter HRIR and the longer BRIRs, different methods are applied to convert them from the time-domain and spatial-domain into CLDFB domain and spherical-harmonics domain. The former case is described in clause 7.2.2.4.2.1 and the latter in 7.2.2.4.2.2.

For the HRIRs, a conversion of the filters to the spherical-harmonics domain is performed to allow for the rendering of ambisonics signals to binaural output. For the BRIRs, the filters are only stored in the spatial domain. When ambisonics

signals are rendered, they are first decoded to a CICP19 loudspeaker configuration and then convolved with the filters in the spatial domain.

7.2.2.4.2.2 HRIR filters

The conversion from time domain to CLDFB domain for HRIRs in described in clause 7.4.7.3.22. For SHD HRIRs first a conversion from spatial domain to the spherical harmonic domain is applied as described in clause 7.4.7.3.1.

7.2.2.4.2.3 BRIR filters

The time-domain, spatial-domain filters $h_i^{td,sd,L/R}(l)$ are converted to the CLDFB domain versions $h_i^{cldfb,sd,L/R}(l,k)$ for each ear L and R and each channel index i as described in clause 6.2.5.1. The channels are those from a CICP19 loudspeaker layout. The filters employed in IVAS originate from measurements conducted in the Room "Mozart" at Fraunhofer IIS.

To reduce the computational complexity with the long BRIRs, the filter length is reduced from N_{taps} to N_{mix} taps.

First the initial zero taps before the onset of the responses are removed. Then the filters are cut off after a number of maximum taps, N_{mix}, which corresponds to a mixing time $t_{\text{mix}}(k)$, which is, in turn, calculated for each CLDFB band k from the energy decay curve of the measured filters.

Specifically, $t_{mix}(k)$ and N_{mix}(k) are chosen such that

$$EDC(t_{mix}(k),k) \leq EDC(0,k) + THR_{DE}$$

with the energy decay curve (EDC), which is calculated as

$$EDC(t,k) = 10 * \log_{10} \frac{s_2(t,k)}{s_1}$$

where $s_{i,1}^{L/R}(k) = \sum_{l=0}^{l=N_{\text{taps}}} \left|h_i^{cldfb,sd,L/R}(l,k)\right|^2$, $s_{i,2}^{L/R}(k) = \sum_{l=N_0(t)}^{l=N_{\text{taps}}} \left|h_i^{cldfb,sd,L/R}(l,k)\right|^2$, and $N_0(t) = SR * t$ the number of taps at time t. Hence, $EDC(t,k)$ is the ratio of the energy in the room response after t over the energy of the whole room response.

Finally, $N_{max}(k)$ is set to values given in Table 7.2-10 for groups of CLDFB bands in terms of the maximum number of tabs over all bands, $N_{\text{mix,max}} = \max_{\text{k}}(\text{N}_{\text{mix}}(k))$.

Table 7.2-10: N_max for groups of CLDFb bands

CLDFB bands	N_mix
1-5	$\lceil 1.0\ N_{\text{mix,max}} \rceil$
6-10	$\lceil 0.6\ N_{\text{mix,max}} \rceil$
11-20	$\lceil 0.5\ N_{\text{mix,max}} \rceil$
21-30	$\lceil 0.4\ N_{\text{mix,max}} \rceil$
31-45	$\lceil 0.3\ N_{\text{mix,max}} \rceil$

7.2.2.4.3 Late reverb model

7.2.2.4.3.1 Reverb signal generation

The remaining long tail, i.e. all taps after N_{mix}, of the filter is modelled parametrically. For this purpose, parameters are extracted from the filters. These parameters include, for each CLDFB band, and energy ENRG and a reverberation time RT60.

The parametric reverberator includes a feedback delay loop processor (see Figure 7.2-9) for different frequency subband signals. These signals consist of different time-frequency tiles of the binaural signal $\boldsymbol{S}_{\text{bin}}^{\text{cldfb}}(m,k)$ obtained according to clause 7.2.2.4.2.2 in the CLDFB domain, which are delayed by different loop delay amounts to acquire the reverberated frequency subband signals.

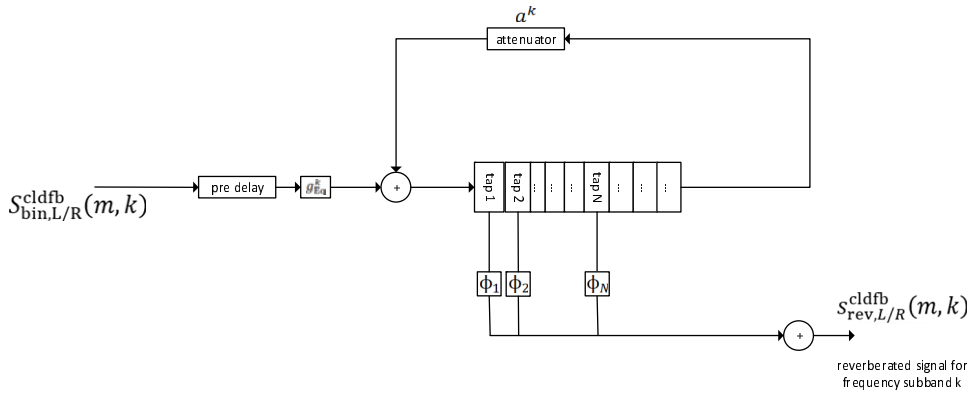


Figure 7.2-9: feedback loop processor

These signals comprise one reverberated frequency subband signal corresponding for each time-frequency tile. For example, a first reverberated frequency subband signal corresponds to the first time-frequency tile and a second reverberated subband signal corresponds to a second reverberated frequency subband signal.

To each frequency subband signal $S^{\text{cldfb}}_{\text{bin,L/R}}(m,k)$ (for $k = 0,1,2 \ldots N_{bands}$) a pre-delay is applied first. The left channel is additionally phase shifted by 90°.

These subband signals are then multiplied by reverb equalization gains g^k_{Eq} and fed into delay lines L_k according to Figure 7.2-9, where each delay line has $N^k_{\text{delay\_taps}}$ sparse taps such that $N^k_{\text{delay\_taps}} < N^k_{\text{time\_slots}}$ with the number of CLDFB time slots buffered in the delay line, $N^k_{\text{time\_slots}}$. Each tap provides a delayed version of the signal. For example, the first delay line provides the $N^1_{\text{delay\_taps,L/R}}$ delayed signals $s^{\text{cldfb}}_{\text{i,L/R}}(m,1)$, where the index i runs over all taps in the first delay line. The second delay line provides the $N^2_{\text{delay\_taps,L/R}}$ delayed signals $s^{\text{cldfb}}_{\text{i,L/R}}(m,2)$ and so on. The left and right (L and R) channels each have their own independent delay lines. The feedback loops are closed by inputting the output signal to the input of each delay line attenuated by the attenuation factor a^k (see Figure 7.2-7).

A combiner then combines the delayed signals $s^{\text{cldfb}}_{\text{i},L/R}(m,k)$ from the taps with a per-tap phase factor $\phi_i(k)$ as $s^{\text{cldfb}}_{\text{rev},L/R}(m,k) = \sum_i \phi_i(k)\, s^{\text{cldfb}}_{\text{i},L/R}(m,k)$ to obtain the reverberated frequency subband signals $s^{\text{cldfb}}_{\text{rev},L/R}(m,k)$ for the time-frequency tile indexed by m and k. The phase factors are always one of 0°, 90°, 180°, or 270°. This allows for a very efficient implementation using only swaps of the real and imaginary parts and sign changes. This combination is calculated for each frequency subband independently.

The first feedback loop together with the first delay line represent a first loop delay amount and wherein the second feedback loop together with the second delay line represent a second loop delay amount, the second loop delay amount being different from the first loop delay amount.

The outputs of the N_{bands} feedback loop processors for the corresponding frequency subbands are then processed further to acquire the reverberated binaural audio signal. Specifically, the reverberated left and right binaural audio channels are inter-mixed with fixed gains in order to simulate an incoherent binaural signal according to

$$s^{\text{cldfb}}_{\text{rev},L}(m,k) \leftarrow g^k_c\, s^{\text{cldfb}}_{\text{rev},L}(m,k) + g^k_x\, s^{\text{cldfb}}_{\text{rev},R}(m,k),$$

$$s^{\text{cldfb}}_{\text{rev},R}(m,k) \leftarrow g^k_c\, s^{\text{cldfb}}_{\text{rev},R}(m,k) + g^k_x\, s^{\text{cldfb}}_{\text{rev},L}(m,k),$$

where g^k_c and g^k_x are the coherence and cross-mix gains, respectively.

The reverberated signal of all frequency subbands, $s^{\text{cldfb}}_{\text{rev},L/R}(m,k)$, are then combined in order to acquire the reverberated signal with the combined bandwith.

7.2.2.4.3.2 Model parameter derivation

The above processing strongly depends on the choice of the model parameters:

- the length of the delay lines of each frequency band, N^k_{delay}
- the number of taps in the delay lines $N^k_{\text{delay\_taps}}$

- the positions of the taps along the delay line $n_1^k, \ldots, n_{N_{\text{delay\_taps}}^k}^k$
- the phase factors $\phi_1^k, \ldots, \phi_{N_{delay\_taps}}^k$
- the gains g_c^k and g_x^k
- the attenuation factors a^k

These parameters are determined from the BRIRs $h_i^{cldfb,sd,L/R}(l,k)$.

The delay-line length is determined according to the formula $N_{delay}^k = \max(\lfloor 1.45 * \lfloor t_{rev}^k * 150 \rfloor + 1 \rfloor, N_{max}^k)$ with $N_{\text{max}}^k = \left\lfloor \frac{500}{1+k} + 60 - k \right\rfloor$.

The number and positions of the taps, $N_{\text{delay\_taps}}^k$ and $n_1^k, \ldots, n_{N_{\text{delay\_taps}}^k}^k$, the phase factors $\phi_1^k, \ldots, \phi_{N_{delay\_taps}}^k$, and the reverb equalization gains g_{Eq}^k are determined randomly according to the following pseudo-code, which is run for each band index k.

```
        for each of L and R channels
        for each bin with index k up to N_bands
        {
            energyBuildup = 0
            currentEnergy = 1
            N^k_delay_taps = 0;

            for each time slot index i_ts in the delay line sample up to N^k_delay
            {
                intendedEnergy += currentEnergy;

                /* The randomization at the energy build up affects where the sparse taps are
located */
                energyBuildup += currentEnergy + 0.1 * random_number

                if ( energyBuildup >= 1.0f ) /* A new filter tap is added at this condition */
                {
                    /* Four efficient phase operations: n*pi/2, n=0,1,2,3 */
                    φ^k_tap = n*pi/2 with random number 1, 2, 3 ,or 4

                    /* Set the tap pointer to point to the determined sample at the loop buffer */
                    n^k_tap = i_ts

                    energyBuildup -= 1.0f; /* A tap is added, thus remove its energy from the
buildup */
                    N^k_delay_taps++;
                    actualizedEnergy += 1.0f;
                }
                currentEnergy *= (attenuationFactorPerSample^k)^2 ;
            }
            N^k_delay_taps  = tap; /* Number of taps determined at the above random procedure */
        }
        g^k_Eq = sqrt(E^k_rev)  /* Determined reverb spectrum */
        g^k_Eq = g^k_Eq sqrt(intendedEnergy / actualizedEnergy) /* Correction of random effects at the decorrelator design
*/
g^k_Eq = g^k_Eq sqrt(0.5 ( 1 − (attenuationFactorPerSample^k)^2 )) /* Correction of IIR decay rate */
```

The sequence of pseudo-random numbers is generated in a reproducible way by the following simple C-language function.

```
static uint16_t binRend_rand(
    REVERB_STRUCT_HANDLE hReverb /* i/o: binaural reverb handle          */
)
{
    hReverb->binRend_RandNext = hReverb->binRend_RandNext * 1103515245 + 12345;

    return (uint16_t) ( hReverb->binRend_RandNext / 65536 ) % 32768;
}
```

The initial value of binRend_RandNext is 1. The attenuation factor $attenuationFactorPerSample$ for each CLDFB band k is calculated according to the formula

$$attenuationFactorPerSample^k = 10^{-\frac{3}{N_{\text{rev}}^k}}$$

with N_{rev}^k the number of CLDFB time slots in the reverberation time $RT60$. The loop attenuation factor is given by

$$a^k = \left(10^{-\frac{3}{N_{\text{rev}}^k}}\right)^{N_{delay}^k}$$

The RT60 per CLDFB band is listed in Table 7.2-11. They have been extracted from the measured BRIRs $h_i^{td,sd,L/R}(l)$.

Table 7.2-11: RT60 per CLDFB band

```
0.429201, 0.205110, 0.202338, 0.208383, 0.215664, 0.236545, 0.230598, 0.228400, 0.227467,
0.218956, 0.226083, 0.220702, 0.221501, 0.223471, 0.223705, 0.227063, 0.227899, 0.223071,
0.220000, 0.218583, 0.220417, 0.218250, 0.213250, 0.210333, 0.207417, 0.198750, 0.196250,
0.194917, 0.190333, 0.184500, 0.180333, 0.176167, 0.176500, 0.177583, 0.183583, 0.195917,
0.203250, 0.208417, 0.214667, 0.220000, 0.222917, 0.230417, 0.233928, 0.233647, 0.236333,
0.237428, 0.241629, 0.241118, 0.238847, 0.242384, 0.246292, 0.245948, 0.246100, 0.245396,
0.243951, 0.244123, 0.239270, 0.241474, 0.234824, 0.253040
```

The long-tail reverberation energy is given by $E_{\text{rev}}^k = \sum_{l=N_{\text{mix}}}^{l=N_{\text{taps}}} \left| h_i^{cldfb,sd,L/R}(l,k) \right|^2$. The values found from the filters used in IVAS are listed in Table 7.2-12.

Table 7.2-12: E_{rev}^k per CLDFB band

```
0.000584, 0.000210, 0.000233, 0.000212, 0.000257, 0.001518, 0.001154, 0.001097, 0.001265,
0.001298, 0.002320, 0.002432, 0.002686, 0.002702, 0.002632, 0.002564, 0.002732, 0.002727,
0.002609, 0.002524, 0.003417, 0.001783, 0.000987, 0.000699, 0.000606, 0.000536, 0.000511,
0.000569, 0.000600, 0.000543, 0.001257, 0.001209, 0.000957, 0.000601, 0.000274, 0.000106,
0.000072, 0.000051, 0.000040, 0.000030, 0.000024, 0.000018, 0.000014, 0.000013, 0.000012,
0.000011, 0.000009, 0.000009, 0.000008, 0.000008, 0.000007, 0.000006, 0.000005, 0.000003,
0.000002, 0.000002, 0.000001, 0.000001, 0.000000, 0.000000
```

Effectively, this results in different numbers of taps and different gaps between the taps in the different delay lines corresponding to different frequency subbands. If one considers a first and a second part of the delay line such that the second part comes after the first, the average number of taps per time slot is lower for the second part than the first. This is because, in the above pseudo code, the `energyBuildup` increases more slowly with the time-slot index. In other words, this means that the average gap between the taps is larger in the second part than in the first.

The gains g_c^k and g_x^k are determined based on an inter-channel correlation measure. Specifically, the correlation coefficient between the left and right binaural channels, C_{IC}^k, is modelled. From this coefficient, the cross-mix gain is calculated as

$$g_x^k = \text{sign}(C_{IC}^k)\sqrt{\left|\frac{1-\sqrt{1-(C_{IC}^k)^2}}{2}\right|}$$

The coherence direct gain is calculated as

$$g_c^k = \sqrt{1-\left|\frac{1-\sqrt{1-(C_{IC}^k)^2}}{2}\right|}$$

The coefficient C_{IC}^k is calculated from the empirically determined formula

$$C_{IC}^{k} = \frac{\sin\left(\pi \frac{f^{k}}{550} + 10^{-20}\right)}{\pi \frac{f^{k}}{550} + 10^{-20}}\left(1 - \frac{f^{k}}{2700}\right)$$

with the center frequency of the band k, f^{k} in Hz. For the zeroth bin, $C_{IC}^{k} = 1$, and, for $f^{k} > 2700$, $C_{IC}^{k} = 0$.

7.2.2.4.4 Head tracking

Prior to the convolution of the signal $\boldsymbol{S}(m, k)$ with the HRIR or BRIR filters a rotation is applied to this signal. The method to apply this rotation depends on the type of signal.

For ambisonics signals, a rotation matrix in the spherical-harmonics domain is pre-multiplied to the signal vector. This matrix multiplication is performed only for the channels up to the cutoff order L. The higher-order channels are reconstructed in the requested orientation by the DirAC decoder according to clause 6.4.6.5.6.2.

For multichannel signals, a rotation matrix for the spatial domain is computed using EFAP (cf. clause 7.2.1.3) and is pre-multiplied to the signal vector to perform the desired signal rotation.

7.2.2.5 Crend binaural renderer

7.2.2.5.1 Terms and Definitions

- DFT : Discrete Fourier Transform, in IVAS binaural renderer convolver, MDFT/IMDFT is used
- IDFT : Inverse Discrete Fourier Transform
- HRIR : Head Related Impulse Response
- BRIR : Binaural Room Impulse Response
- $z^{-L.m}$ block delay line operator. Block delay of m frequency blocks of size L (i.e., $L.m$ frequency samples). L behind the size of a frame which is also equal to the size of DFT used (MDFT/IMDFT [5.2.5])
- $*_{[0\ ;\ \ldots\ ;\ f]}$ 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)
- The energy of a time-domain single-channel signal $x(n)$ is defined by :

$$E(x) = \sum_{n=0}^{+\infty} x^{2}(n)$$

7.2.2.5.2 General

The Crend binaural renderer operates on signals in time domain. In the IVAS decoder, it is used for binaural rendering of multichannel signals, where each audio signal is encoded and decoded using discrete multi-channel mode or for discrete ISM with binaural output with room acoustics synthesized using impulse responses (7.2.2.1). The convolver (7.2.2.5.3) uses a zero-delay block DFT implementation. DFT/IDFT is implanted using MDFT/IMDFT [5.2.5] allowing buffer size being equal to the decoder frame size.

7.2.2.5.3 Convolver

The Crend binaural convolver implementation allows to convolve time domain audio input signal with BRIR efficiently. The complexity of the process depends on how the original BRIR are parametrized. A parametrization process is described in section 7.4.7.3. Default rom values are generated using this process. It is based on the following principle (see figure 7.2-10). In typical BRIRs measured in the same room, after the mixing time (Tm) late reverberation characteristics are similar, and high-frequency energy decays faster than low-frequency energy.

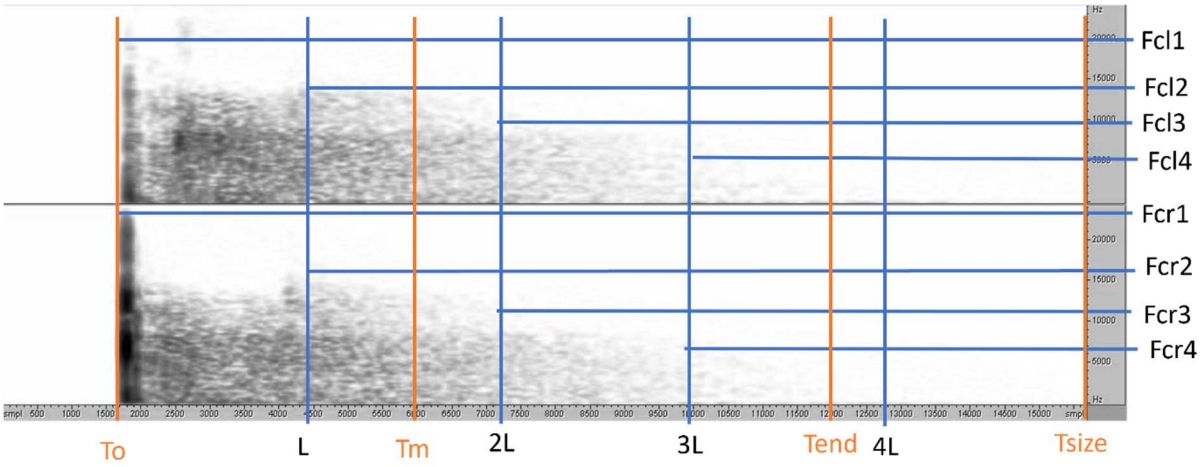


Figure 7.2-10: Typical Time frequency image of a BRIR split in different time-frequency block.

Considering a set of N $BRIR_k^{l,r}$, $k \in [1, N]$, $[l, r]$ left and right ears, the principle of the optimisation is to split the $BRIR_k^{l,r}$ in blocks of the size of the DFT.

Each $BRIR_k^{l,r}$ has:

- a starting or propagation time $To_k^{l,r}$,
- a mixing time $Tm_k^{l,r}$
- an ending time $Tend_k^{l,r}$
- a size $Tsize$ (size of the $BRIR_k^{l,r}$ set, for example).

Let's define:

- $To = \min_{k,l,r}\left(To_k^{l,r}\right)$, the minimal value of the propagation time for the BRIR set.
- $Tm = \max_{k,l,r}\left(Tm_k^{l,r}\right)$, the maximal value of the mixing time for the BRIR set.
- $Tend = \max_{k,l,r}\left(Tend_k^{l,r}\right)$, the maximal value of the ending time for the BRIR set . $Tend_k^{l,r}$ can be set to $Tsize$ or computed by looking when the remaining energy after the ending time becomes negligible.

Each $BRIR_k^{l,r}$ are split in blocks of impulse responses of length L, size of the MDFT (5ms), starting from To. For zero delay convolution the size of the MDFT must be the size of the input audio signal buffer.

$$\boldsymbol{b}_k^{l,r}(m) = BRIR_k^{l,r}(To + m * L, To + (m+1).L - 1) \quad (7.2\text{-}78)$$

$$\boldsymbol{B}_k^{l,r}(m) = MDFT\left(\boldsymbol{b}_k^{l,r}(m)\right) \quad (7.2\text{-}79)$$

$$k \in [1, N], 0 \leq m \leq Tsize * F/L$$

Since in typical BRIRs high-frequency energy decays faster than low-frequency energy, to reduce the number of complex multiplications during convolution processing, cut-off frequencies can be extracted for each block, above which frequency bins multiplications are not performed.

For each block a cut-off frequency is defined:

$$fc_k^{l,r}(m), k \in [1, N], 0 \leq m \leq Tsize * F/L$$

Each $BRIR_k^{l,r}$ is split in a set of direct blocks and a set of diffuse blocks. The limit between direct and diffuse blocks is computed using mixing time $Tm_k^{l,r}$ and the DFT size L for each $BRIR_k^{l,r}$. The maximum value m_{tm} of $Tm_k^{l,r}$ can then be computed using following formulas:

$m_{tm}{}_k^{l,r}$ is defined as the number of direct blocks for $BRIR_k^{l,r}$

Thus $m_{tm}{}_k^{l,r} * L \leq (Tm_k^{l,r} - To) * F < m_{tm}{}_k^{l,r} + 1$

Then $m_{tm} = \max_{k,l,r}\left(m_{tm}{}_k^{l,r}\right)$

The ending block $m_{end}{}_k^{l,r}$ for each $BRIR_k^{l,r}$ is computed using $Tend_k^{l,r}$ and the DFT size L for each $BRIR_k^{l,r}$. The size the BRIR being reduced the complexity is reduced.

$$m_{end}{}_k^{l,r} * L \leq (Tend_k^{l,r} - To) * F < m_{end}{}_k^{l,r} + 1$$

Then $m_{end} = \max_{k,l,r}\left(m_{end}{}_k^{l,r}\right)$

To reduce even more the complexity, diffuse blocks for each channel (blocks after the mixing time) are replace by filter blocks that are common to all inputs. These left, right filter blocks are obtained by averaging $\boldsymbol{b}_k^{l,r}(m)$ over k:

$$BRIR_{diff}^{l,r}(t > Tm) = \frac{1}{N}\sum_{k=1}^{N} W_k^{l,r}.\, BRIR_k^{l,r}(t > Tm) \quad (7.2\text{-}80)$$

$$W_k^{l,r} = 1/\sqrt{E(BRIR_k^{l,r}(t > Tm))}$$

$$b_{diff}^{l,r}(m) = BRIR_{diff}^{l,r}(Tm + m * N_d; Tm + (m+1).N_d - 1)$$

$$B_{diff}^{l,r}(m) = MDFT\left(b_{diff}^{l,r}(m)\right) \quad (7.2\text{-}81)$$

$$0 \leq m \leq Tend_k^{l,r} * F/L$$

Finally, the left/right output for current audio frame is given by the following formula:

If $m_{end} > 0$:

$$O^{l,r} = \sum_{k=1}^{N}\left(\sum_{m=0}^{m_{tm}-1}\left(B_k^{l,r}(m) *_{\left[0;\ldots;fc_k^{l,r}(m)\right]} (z^{-L.m}.\, I_k)\right)\right) + \sum_{m=0}^{m_{end}-1}\left(B_{diff}^{l,r}(m) *_{\left[0;\ldots;f_c^{l,r}(m)\right]} (z^{-L.m}.\, D^{l,r})\right) \quad (7.2\text{-}82)$$

Else:

$$O^{l,r} = \sum_{k=1}^{N}\left(\sum_{m=0}^{m_{tm}-1}\left(B_k^{l,r}(m) *_{\left[0;\ldots;fc_k^{l,r}(m)\right]} (z^{-L.m}.\, I_k)\right)\right)$$

where

- $O^{l,r}$: left/right binaural output frame of size L
- I_k : DFT of current frame of size L of input k, DFT used in IVAS is the MDFT [5.2.5].
- $D^{l,r} = \sum_{k=1}^{N} G_k^{l,r}.\,(z^{-L.m_{tm}}.\, I_k)$, is a weighted sum of the I_k delayed by $L.\, m_{tm}$ blocks.
- $G_k^{l,r} = \frac{G(k)}{W_k^{l,r}}$ are weighting gains to adjust the level of each input in the reverberated outputs (l, r).
    - $1/W_k^{l,r}$ adjust the energy of $b_{diff}^{l,r}$ to have the same energy as $b_k^{l,r}$.
    - $G(k)$ are a predeterminate gain (by default set to 1) allowing to adjust the level of reverberation for each input see (7.4.6.1). It can be useful to adjust the level of late reverberation depending on the type of input (voice, musique, cinema, VR, AR, …).

7.3 Room acoustics rendering

7.3.1 Introduction to room acoustics rendering

IVAS rendering supports synthesis of room acoustics for realistic immersive effect. The room acoustics can be synthesized using room impulse response convolution or late reverb, optionally combined with early reflections. The room impulse response data (BRIRs) are discussed in clause 7.4.7.

The general reverb processing block diagram is shown in 7.3-1.

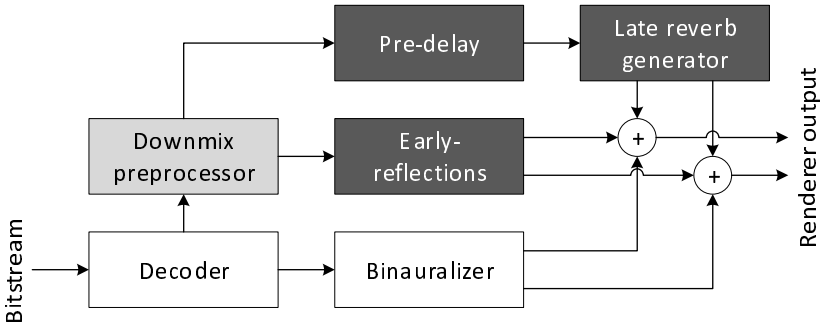


Figure 7.3-1: Synthetic reverb processing block diagram

The reverb is provided with the (partially) decoded audio downmix as input. In the case of ParamBin and FastConv renderers, the downmix is provided in CLDFB domain as binaural signal. In the case of the time-domain renderers (Crend and Time-domain object renderer), the downmix is provided as time-domain mono signal.

The late reverb and early reflection synthesis are driven by the set of parameters that are discussed in details in clause 7.4.8.

7.3.2 Room impulse response convolution

The direct sound components and early reflections are rendered by fast convolution with the BRIRs in the FastConv binaural renderer according to clause 7.2.2.4.2.1 with the filter truncation according to clause 7.2.2.4.2.2.

7.3.3 Sparse frequency-domain reverberator

The late reverberations are rendered using a model approach. The Sparse frequency-domain reverberator that affords the simulation of these responses is described in clause 7.2.2.4.3.1. The parameters of this model are determined from the measured BRIRs according to clause 7.2.2.4.3.1.

The parameters of this model can be also computed from the room acoustics parameters as discussed in clause 7.4.7. In this case, the reverb model parameters T60 and DSR frequency domain representation, as well as the average HRTF power for left and right channel are resampled using linear interpolation to the uniform CLDFB grid, as described in clause 7.3.4.2 using the CLDFB uniform frequency grid calculated as:

$$f_c[k] = \left(k + \frac{1}{2}\right)\frac{fs}{2 \cdot N_{CLDFB}} \tag{7.3-1}$$

for $k = 0, \ldots, N_{CLDFB}$.

The energy parameters of the reverberation model are computed from DSR parameters based on difference between input pre-delay and acoustic pre-delay Δ_{delay}:

$$ene[k] = DSR_{output}[k] \cdot e^{max\left(min\left(\frac{\Delta_{delay}\log(1e^6)}{T60_{output}[k]}, 23.0\right), -23.0\right)} \cdot \frac{\overline{P}_L^{HRTF}[k] + \overline{P}_R^{HRTF}[k]}{2} \cdot g_{dmx} \tag{7.3-2}$$

where:

$$g_{dmx} = \sqrt{\frac{4\pi d_{def}^2}{1e^{-3}}} \quad (7.3\text{-}3)$$

7.3.4 Feedback-delay network reverberator

7.3.4.1 Overview

The feedback-delay network reverberator provided in IVAS decoder/renderer is based on the Jot reverberator. Additionally, filters have been added to control interaural correlation and ear-dependent coloration. A schematic depiction of the modified Jot reverberator is shown in Figure 7.3-2.

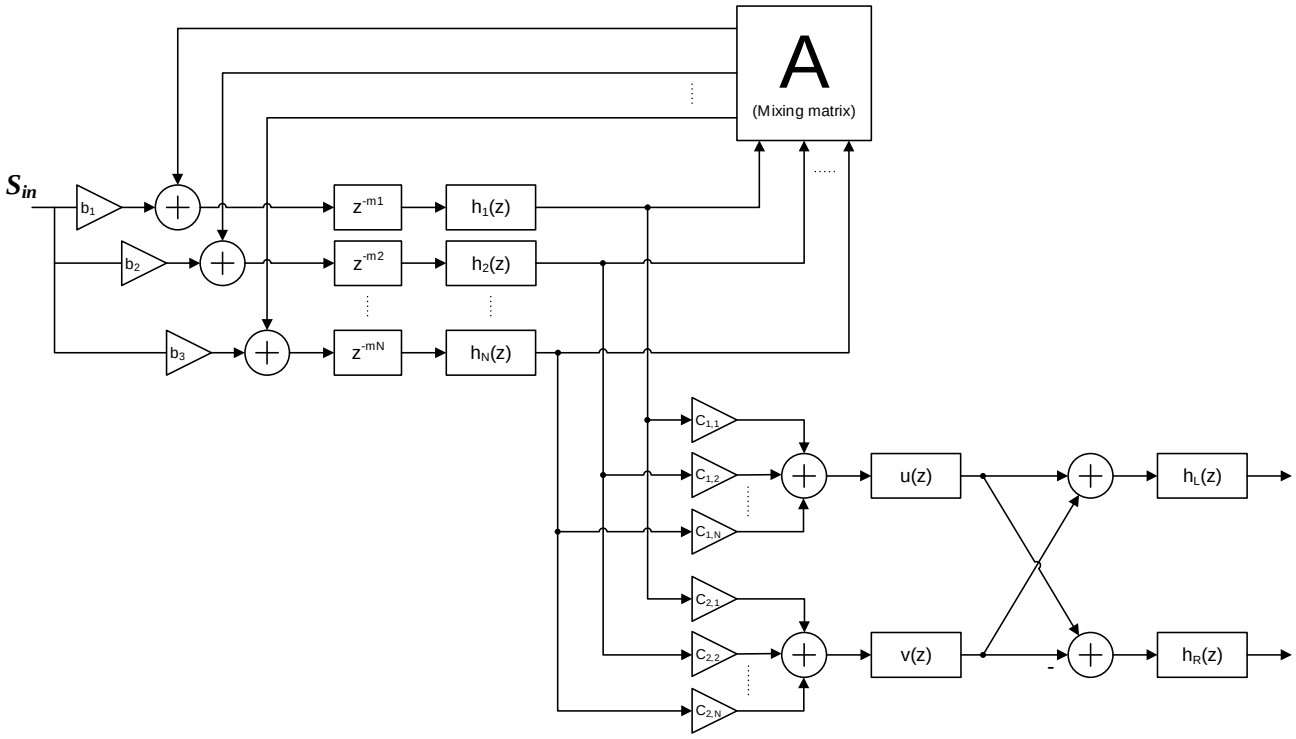


Figure 7.3-2: Feedback-delay network reverberator

The weights *b* and *c* control input and output of the feedback-delay network. Interaural coherence is controlled using $u(z)$ and $v(z)$ filter coefficients, while ear-dependent coloration using $h_L(z)$ and $h_R(z)$. To match with the direct path binaural filter characteristics, the coloration filters are pre-computed based on reverb characteristics, and on HRIR used for binauralization.

7.3.4.2 Reverberator configuration

The Jot reverberator makes use of 8 feedback loops (branches). The loop delays $d_{loop}[l]$ are set depending on the output sample rate f_s as provided in Table 7.3-1.

Table 7.3-1: Jot reverberator loop delays per branch

Loop delay index *l*	f_s = 48 *kHz*	f_s = 32 *kHz*	f_s = 16 *kHz*
0	2309	1531	769
1	1861	1237	619
2	1523	1013	509
3	1259	839	421
4	1069	709	353
5	919	613	307
6	809	541	269
7	719	479	239

The 8x8 matrix for mixing the 8 feedback loop outputs back to the 8 inputs are calculated using the following matrix:

$$M_{feedback} = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad (7.3\text{-}4)$$

The 8 loops are matrix multiplied by a gain matrix, defined as:

$$M_{output}[l, ch] = \begin{cases} 1 & if \quad ch = 0 \\ (-1)^l & if \quad ch = 1 \end{cases} \quad (7.3\text{-}5)$$

Reverb is generated in subframes with length equivalent to 1/4 of an IVAS frame. As parts of reverb processing is performed in FFT domain, the FFT size must fit a corresponding subframe size. These relations are illustrated in Table 7.3-2.

Table 7.3-2: Jot reverberator loop delays per branch

Sample rate	Frame size	Subframe size	FFT size
f_s = 48 *kHz*	960	240	512
f_s = 32 *kHz*	640	160	512
f_s = 16 *kHz*	320	80	256

The uniform frequency grid is calculated as:

$$f_c[k] = \frac{k \cdot fs}{N_{FFT}} \quad (7.3\text{-}6)$$

for $k = 0, \ldots, N_{FFT}/2$.

The T60 and DSR frequency domain representation are resampled using linear interpolation to the uniform FFT grid. If the number of bands in which the room acoustics data is defined is given by: $n_{room\_acoustics\_bands}$, with the center frequencies $f_{c,input}[k_{in}]$, on the uniform grid the values are determined as:

$$T60_{output}[k] = T60_{input}[k_{in}] + \frac{f_c[k] - f_{c,input}[k_{in}]}{f_{c,input}[k_{in}+1] - f_{c,input}[k_{in}]}\left(T60_{input}[k_{in}+1] - T60_{input}[k_{in}]\right) \quad (7.3\text{-}7)$$

$$DSR_{output}[k] = DSR_{input}[k_{in}] + \frac{f_c[k] - f_{c,input}[k_{in}]}{f_{c,input}[k_{in}+1] - f_{c,input}[k_{in}]}\left(DSR_{input}[k_{in}+1] - DSR_{input}[k_{in}]\right) \quad (7.3\text{-}8)$$

for $k = 0, \ldots, N_{FFT}/2$ and k_{in} is determined as the lowest k_{in} such that $f_c[k] \leq f_{c,input}[k_{in}]$. If $f_c[k] < f_{c,input}[0]$, the values are determined as:

$$T60_{output}[k] = T60_{input}[0] \tag{7.3-9}$$

$$DSR_{output}[k] = DSR_{input}[0] \tag{7.3-10}$$

If $f_c[k] > f_{c,input}\left[n_{room\_acoustics\_bands} - 1\right]$, the values are determined as:

$$T60_{output}[k] = T60_{input}\left[n_{room\_acoustics\_bands} - 1\right] \tag{7.3-11}$$

$$DSR_{output}[k] = DSR_{input}\left[n_{room\_acoustics\_bands} - 1\right] \tag{7.3-12}$$

The DSR values are adapted based on the difference of the input pre-delay $t_{input,pre}$ and the acoustic pre-delay t_{pre} (both expressed in seconds) $\Delta_{delay} = t_{input,pre} - t_{pre}$. For each bin k= 0, … , $N_{FFT}/2$, the DSR value is adjusted according to:

$$DSR'_{output}[k] = DSR_{output}[k] \cdot e^{max\left(min\left(\frac{\Delta_{delay}\log(1e^6)}{T60_{output}[k]}, 23.0\right), -23.0\right)} \tag{7.3-13}$$

The downmix gain is calculated as:

$$g_{dmx} = \sqrt{\frac{4\pi d_{def}^2}{1e^{-3}}} \tag{7.3-14}$$

With the default source distance of $d_{def} = 1.5m$

Given the acoustic pre-delay t_{pre} expressed in seconds, and the output sample rate f_s, the pre-delay (in samples) is calculated as following.

$$n_{pre} = min\left(max\left(\left\lfloor t_{pre} \cdot f_s + 0.5 \right\rfloor - d_{loop}[L-1], 0\right), f_s/50\right) \tag{7.3-15}$$

To calculate the T60 filter coefficients, first the normalized frequencies are calculated as:

$$f_{norm}[k] = \frac{f_c[k]}{0.5 \cdot f_s} \tag{7.3-16}$$

for $k = 0, \ldots, N_{FFT}/2$. Then, for each loop (branch), the following procedure is repeated:

For each frequency bin $k = 0, \ldots, N_{FFT}/2$, the target gain (in dB) is determined as:

$$g_{target}[k] = max\left(-60.0\frac{d_{loop}[l]}{f_s \cdot T60_{output}[k]}, -120.0\right) \tag{7.3-17}$$

The target gains (in dB) are then averaged for a low and a high frequency band.

$$g_{target,lo} = \frac{1}{k_{lo,up} - k_{lo,down} + 1} \sum_{k=k_{lo,down}}^{k_{lo,up}} g_{target}[k] \tag{7.3-18}$$

$k_{lo,down}$ is the lowest band k for which $f_{norm}[k] \geq \frac{100.0}{0.5 \cdot f_s}$

$k_{hi,down}$ is the highest band k for which $f_{norm}[k] \leq \frac{250.0}{0.5 \cdot f_s}$

$$g_{target,hi} = \frac{1}{k_{hi,up} - k_{hi,down} + 1} \sum_{k=k_{hi,down}}^{k_{hi,up}} g_{target}[k] \tag{7.3-19}$$

$k_{hi,down}$ is the lowest band k for which $f_{norm}[k] \geq \frac{5000.0}{0.5 \cdot f_s}$

$k_{hi,down}$ is the highest band k for which $f_{norm}[k] \leq \frac{7950.0}{0.5 \cdot f_s}$

The mid crossing gain is then calculated as the average:

$$g_{target,mid} = \frac{g_{target,lo} + g_{target,hi}}{2} \tag{7.3-20}$$

The cutoff frequency for the subsequent shelf filter is then determined as the $f_{norm}[k]$ for which the $g_{target}[k]$ is closest to $g_{target,mid}$.

The linear gains are determined as:

$$A_{lo} = 10^{\frac{g_{target,lo}}{20.0}}, \qquad A_{hi} = 10^{\frac{g_{target,hi}}{20.0}} \tag{7.3-21}$$

Given the frequency f_0 and the linear low and high frequency gains A_{lo} and A_{hi}, a first order IIR filter is calculated:

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \tag{7.3-22}$$

If the gain ratio $R = \frac{A_{lo}}{A_{hi}} < 1.0$, the coefficients are calculated as:

$$b_0 = A_{hi}\frac{1+\omega_0 R}{1+\omega_0}, \qquad b_1 = A_{hi}\frac{\omega_0 R - 1}{1+\omega_0}, \qquad a_1 = \frac{\omega_0 - 1}{1+\omega_0} \tag{7.3-23}$$

Else, the coefficients are calculated as:

$$b_0 = A_{hi}\frac{1+\omega_0}{1+\omega_0/R}, \qquad b_1 = A_{hi}\frac{\omega_0 - 1}{1+\omega_0/R}, \qquad a_1 = \frac{\omega_0/R - 1}{1+\omega_0/R} \tag{7.3-24}$$

where:

$$\omega_0 = tan\left(\frac{\pi}{2} f_0\right) \tag{7.3-25}$$

The loop delays are eventually adjusted to reflect the T60 filter order.

Subsequently, coloration filter gains are computed for each bin $k = 0, \ldots, N_{FFT}/2$ and left (L) and right (R) channel as follows:

$$g_L^{color}[k] = \frac{\sqrt{DSR'[k] \cdot ef_{rev} \cdot \overline{P}_L^{HRTF}[k]}}{\sqrt{en_{rev}[k]}}, \qquad g_R^{color}[k] = \frac{\sqrt{DSR'[k] \cdot ef_{rev} \cdot \overline{P}_R^{HRTF}[k]}}{\sqrt{en_{rev}[k]}} \tag{7.3-26}$$

where $ef_{rev} = 1$ is reverb energy factor, $\overline{P}_{L/R}^{HRTF}[k]$ is a pre-computed average HRTF power for a bin *k*, and

$$en_{rev}[k] = \frac{10^{\frac{-6 \cdot min(delay)}{T60_{est}[k] \cdot f_s}}}{2^{\frac{log(1e^{-3})}{T60_{est}[k]}}} \tag{7.3-27}$$

in which $min(delay)$ is the shortest delay of the feedback-delay network and $T60_{est}[k]$ is computed as the sum of the frequency response of all the loops *l* computing the shelf filter coefficients:

$$T60_{est}[k] = \frac{1}{8}\sum_{l=0}^{7} T60_l[k] \tag{7.3-28}$$

with:

$$t60_l[k] = \frac{-3.0 \cdot d_{loop}[l]}{f_s \cdot log_{10}(H_l[k])} \tag{7.3-29}$$

where the frequency response is calculated as:

$$H_l[k] = \sqrt{\frac{{b_0}^2 + {b_1}^2 + 2 b_0 b_1 cos\left(2\pi\frac{f_c[k]}{f_s}\right)}{1 + {a_1}^2 + 2 a_1 cos\left(2\pi\frac{f_c[k]}{f_s}\right)}} \tag{7.3-30}$$

The coloration gains are smoothed across the frequency bins to limit gain differences to 4dB for frequencies up to 1 kHz, and to 1.5dB for high frequencies.

To prepare for processing in the FFT domain, a half Hann window is precalculated and relevant filter instances are initialized.

Subsequently, the correlation filter coefficients for each frequency bin $k = 0, \ldots, N_{FFT}/2$ are set as follows:

$$U_{re}[k] = \sqrt{\frac{1 + IAC[k]}{2}}, \qquad U_{im} = 0, \qquad V_{re}[k] = \sqrt{\frac{1 - IAC[k]}{2}}, \qquad V_{im} = 0 \tag{7.3-31}$$

where $IAC[k]$ is a pre-computed inter-aural coherence for a bin *k*.

The minimum phase filter coefficients are computed eventually to avoid aliasing while applying in the FFT domain. The pre-computed time window (half Hann) is applied to the correlation filter coefficients converted to the time domain, the final correlation filters are computed by converting the obtained signal back to FFT domain.

Coloration filter coefficients are computed similarly to the case of correlation filter coefficients, including minimum phase filter coefficient computation and application of the pre-computed time window.

Finally, the reverberator pre-delay line and feedback-delay network branches are initialized. These steps include setting up the reverb delay lines, first order IIR filter coefficients, and feedback gains.

7.3.4.3 Processing

The high-level rendering block diagram using Jot reverb is provided in Figure 7.3-3.

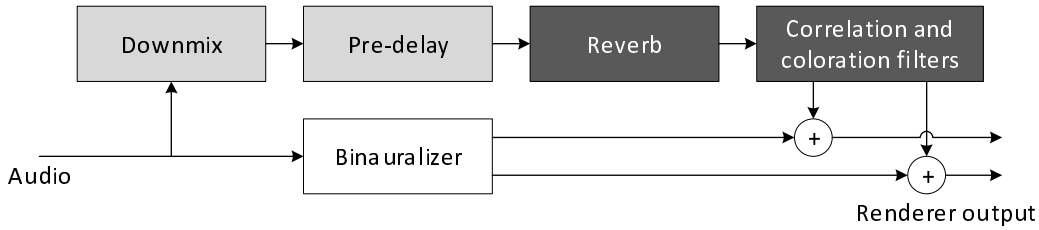


Figure 7.3-3: Rendering block diagram with Jot reverberator

First, the mono downmix signal is computed from input audio and pre-delay is applied. Subsequently, reverb signals are computed for left and right channel. Processing inside reverb block is performed in subblocks of 80 samples for efficient delay line and IIR filter processing.

First, for each branch the samples from the delay lines are gathered and the first order IIR filtering is applied accordingly, utilizing pre-computed T60 filter coefficients, according to the formula:

$$y[n] = b_0 x[n] + b_1 x[n-1] - a_1 y[n-1] \quad (7.3\text{-}32)$$

Once the feedback-delay network output is computed, the incoming samples are fed into the delay lines.

Finally, the correlation and coloration filters are applied to the output signal in the FFT domain. The block diagram of this processing stage is provided in 7.3-6.

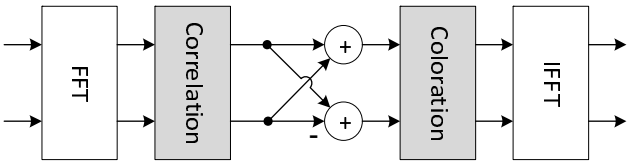


Figure 7.3-4: Reverb output filtering stage block diagram

7.3.5 Early-reflection synthesis

7.3.5.1 General

The early reflections part of the room acoustics immersive effect can be enabled when using any multichannel input with binaural output modes. The reflection parameters described in clause 7.4.8.3 define a virtual 3D room with absorption coefficients representing the average broadband acoustic reflection characteristics of each surface in a geometric rectangular room model. The “shoebox” model computes first order early reflections for each source in the multichannel configuration using the image source method, first described in [16]. This is done by considering the cartesian position of the listener probe and the relative position of the multichannel emitter array within the virtual room. The listener position defaults to the centre of the room (at a standard height) but it can optionally be defined in the renderer parameters to be in a different location than the centre. Once computed, the resulting reflection gains and delay times are used by a process loop to create the reflection signals, totalling to six reflections per source. Each reflection signal is then mixed into the channel buffer that is closest to the direction of arrival, according to the

configuration layout. The resulting mix of early and direct sound is thus jointly sent to the orientation rotation processing creating an interactive directional sound effect.

7.3.5.2 Coordinate System

The virtual room is defined by three dimensions, respectively representing the length, width, and height of a rectangular room, in meter units. The resulting room model of dimensions $R(x, y, z)$ is placed at a centre of a 3D coordinate system where the origin is the centre of the room floor. As seen in the figure below, the room can extend in the positive and negative directions of the x and y axis, but only in the positive direction for the z axis. Each wall surface is referenced by an index following the order of the axis from positive to negative (e.g. $W: 5$ and $W: 6$ reference the surfaces along the z axis, in this case *ceiling* and *floor*), this permits the definition of independent absorption coefficients for each surface. Figure 7.3-5 illustrates the coordinate system.

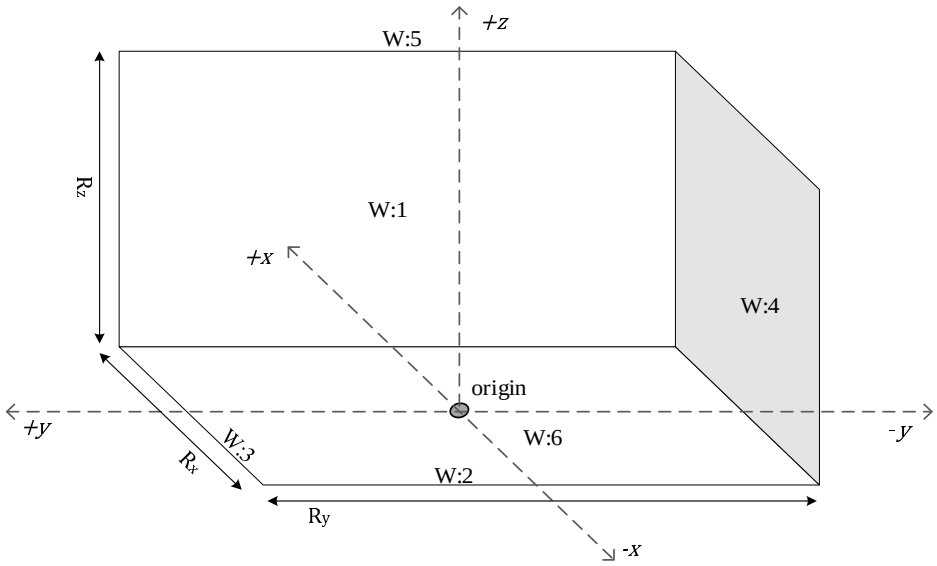


Figure 7.3-5: Coordinate system used by the Early Reflections model

See also Table 7.4-5 for a reference map between the model absorption coefficients and the corresponding surface.

7.3.5.3 Source-receiver location correction

A boundary check executes at initialization stage to make sure the given source/receiver 3D coordinates at point $P(x, y, z)$ are within the virtual room boundaries. A minimum distance from each wall is defined internally to create enough space for reflections to happen with every surface. The boundary limit reference coordinates $B(x, y, z)$ are calculated using the room dimensions and the minimum distance:

$$\begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} R_x/2 \\ R_y/2 \\ R_z \end{bmatrix} - w_d \tag{7.3-33}$$

where R are the room dimensions and w_d is the minimum distance margin allowed for a point from a surface (default 0.1 meters). Each source/receiver point is then checked against the boundary conditions, separately for each axis. Whenever the check fails, the reference coordinate is overwritten to be at the boundary, before feeding the shoebox model:

$$P_x = \begin{cases} \min(P_x, B_x) \text{ if } P_x \geq 0 \\ \max(P_x, -B_x) \text{ if } P_x < 0 \end{cases}, \quad P_y = \begin{cases} \min(P_y, B_y) \text{ if } P_y \geq 0 \\ \max(P_y, -B_y) \text{ if } P_y < 0 \end{cases}, \quad P_z = \begin{cases} \min(P_z, B_z) \text{ if } P_z \geq w_d \\ w_d \text{ if } P_z < w_d \end{cases}$$

(7.3-34)

The order of operation bounds firstly the listener point, then each source in the given configuration using relative channel positioning.

7.3.5.4 Reflections calculations

Each emitter source $E_{(n)}$, in a multichannel configuration made of n channels (LFE channel excluded), possesses a set of six image sources representing reflections originating from every surface w in the rectangular model. For each image source I of index (n, w) the model calculates the broadband gain factor using the specified surface absorption rates α_w and the travel path distance, the time of arrival of the reflection to the receiver, and the direction of arrival in spherical coordinate form.

To calculate the reflection gains, the model first calculates the distance vector of each emitter point source E_n and image source $I_{(n,w)}$ to the listener location P in 3D space.

$$d_{(n)} = \begin{bmatrix} E_{(n)_x} \\ E_{(n)_y} \\ E_{(n)_x} \end{bmatrix} - \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}, \quad v_{(n,w)} = \begin{bmatrix} I_{(n,w)_x} \\ I_{(n,w)_y} \\ I_{(n,w)_x} \end{bmatrix} - \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \tag{7.3-35}$$

The gain is taken as follows:

$$g_{(n,w)} = (1 - \alpha_w)\left(\frac{\|d_{(n)}\|}{\|v_{(n,w)}\|}\right) - \left(\|v_{(n,w)}\| * \alpha_{AIR}\right) \tag{7.3-36}$$

where α_{AIR} is the air absorption coefficient set to $1.37e^{-3}$, $\|v_{(n,w)}\|$ is the norm, or the Euclidean distance, from 3D image source position $I_{n,w}(x, y, z)$ to the receiver probe position $P(x, y, z)$, and $\|d_{(n)}\|$ is the distance from the emitter source $E_{(n)}(x, y, z)$ to the receiver.

The time of arrival for each reflection is calculated by using a standard *speed of sound* coefficient and the total travel path distance from the image source to the receiver probe.

$$TOA_{(n,w)}(s) = \frac{\|v_{(n,w)}\|}{c} \tag{7.3-37}$$

where c is the speed of sound set as 343 m/s. These time indexes are converted into samples by multiplication by the current system sample rate.

$$\tau_{(n,w)} = TOA_{(n,w)}(s) \times (\text{sample rate}) \tag{7.3-38}$$

The direction of arrival of each reflection is broken down into azimuth angle θ and vertical angle ϕ.

$$\theta_{(n,w)} = \text{atan2}\left(\frac{v_{(n,w)_y}}{v_{(n,w)_x}}\right) \text{ rad} \tag{7.3-39}$$

$$\phi_{(n,w)} = \text{asin}\left(\frac{v_{(n,w)_z}}{\|v_{(n,w)}\|}\right) \text{ rad} \tag{7.3-40}$$

7.3.5.5 Process Loop

The coefficients calculated by the shoebox model are used to compute early reflection signals through a delay line architecture that scales and delays copies of the input signal. This is followed by a panning process that mixes each reflection signal with the spatially closest channel in the original multichannel layout, to provide a spatial direction to reflections that approximately matches the image source direction of arrival. The reflections rendering loop runs within the main processing loop prior to the computation of late reverb, rotation processing, and binauralization. The total number of reflection signals equals to the number of channels in the input multichannel configuration, composed by N channels (excluding the LFE), times the number of reflection surfaces used by the shoebox model.

Each input channel signal S_n is fed to six delay lines (one per shoebox surface) using the gain scaling multipliers calculated as per equation (7.3-36) and delayed by a number of samples corresponding to the reflection arrival times as per equation (7.3-38). Each resulting reflection signal $S_{R(n,w)}$ is mixed back with the original input channel that is geometrically closest to the direction of arrival of the reflection as computed by the shoebox, $S_{k\ (n,w)}$.

To find the closest channel, the Euclidean distance is calculated between the cartesian coordinates of the spherical projection on the unit sphere of each reflection signal $S_{R(n,w)}$ based on its direction of arrival determined by equations (7.3-39) and (7.3-40), and each emitter source of the input layout configuration. The cartesian coordinates are taken as follows:

$$\begin{bmatrix} R_{(n,w)_x} \\ R_{(n,w)_y} \\ R_{(n,w)_z} \end{bmatrix} = r \begin{bmatrix} \cos(\phi_{n,w})\cos(\theta_{n,w}) \\ \cos(\phi_{n,w})\sin(\theta_{n,w}) \\ \sin(\phi_{n,w}) \end{bmatrix} \quad (7.3\text{-}41)$$

Where $r = 1$ is the reference projection distance for all reflections.

The index of the most spatially close emitter, indicated by index $k_{(n,w)}$ is chosen as the one producing the smallest Euclidean distance between each emitter coordinates $E_{(m)}$ of the multichannel input (excluding LFE), and the reflection projection coordinates $R_{(n,w)}$.

$$k_{(n,w)} = \underset{m}{\operatorname{argmin}} \left(\sqrt{\left(E_{(m)_x} - R_{(n,w)_x}\right)^2 - \left(E_{(m)_y} - R_{(n,w)_y}\right)^2 - \left(E_{(m)_z} - R_{(n,w)_z}\right)^2} \right) \quad (7.3\text{-}42)$$

where $1 \leq m \leq N$

Index $k_{(n,w)}$ is thus used to select the input channel signal S_k into which the reflection signal is summed in. The process is repeated for each of the $(N \times 6)$ reflections. Figure 7.3-6, illustrates the signal flow architecture.

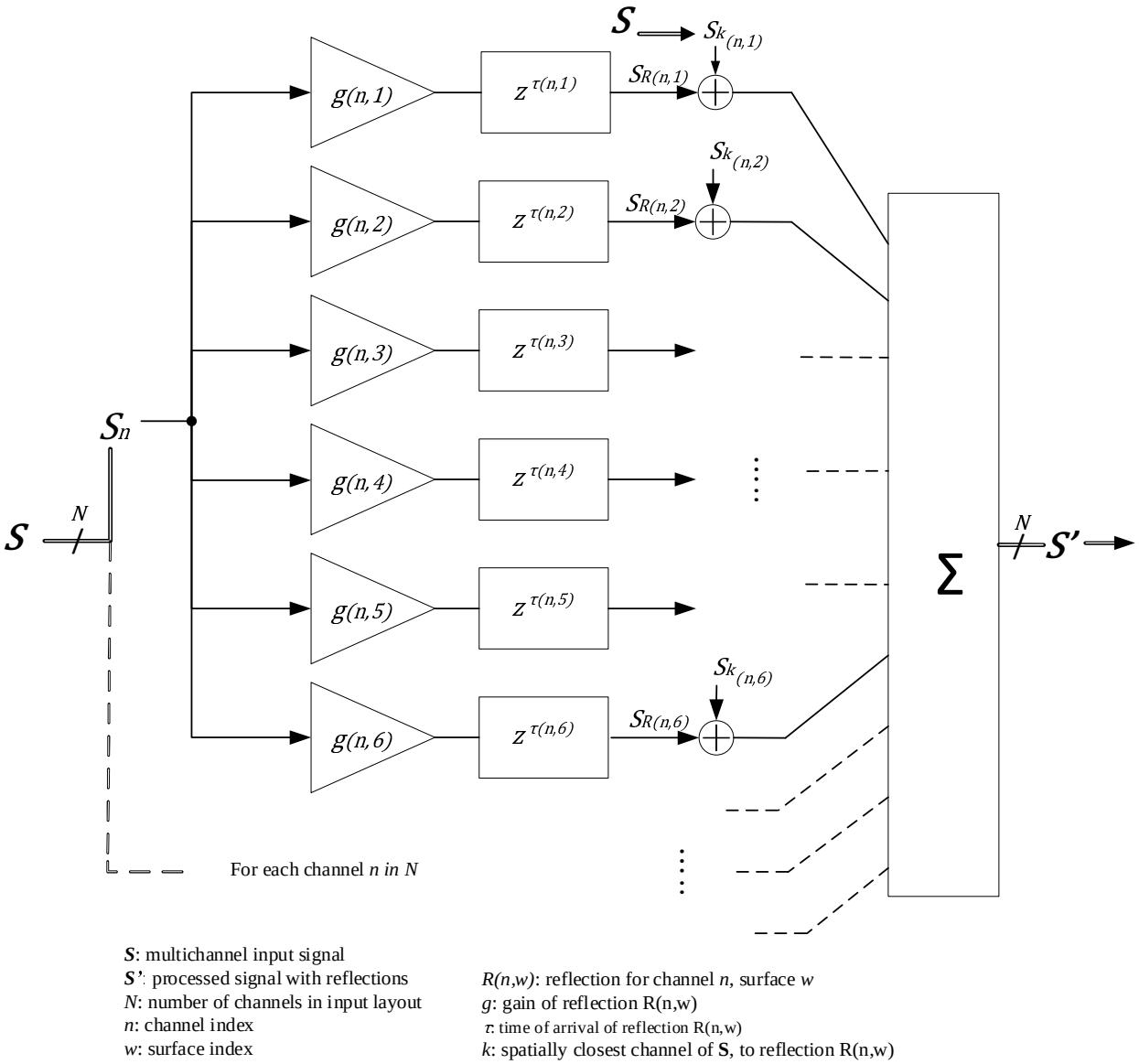


Figure 7.3-6: Delay line architecture for Early Reflections processing using parameters computed by the shoebox model

7.3.5.6 Low-complexity mode

To save on the computational load of rendering early reflections, it is possible to activate a Low-complexity mode option. This mode significantly reduces the number of operations needed to render early reflections, at a minimal cost in perceptual quality. This is achieved through the computation of reflections on a channel layout container mix of a lower density than the original input layout.

Table 7.3-3: Low-Complexity mode channel mix orders for available MC layouts

Original Layout	Low Complexity Layout	Channel mix order (no LFE)	Approximate cost saving
5.1	3.0	L, R, C, L, R	37%
5.1.2	5.0	L, R, C, Ls, Rs, L, R	25%
5.1.4	5.0	L, R, C, Ls, Rs, L, R, Ls, Rs	40%
7.1	5.0	L, R, C, Ls, Rs, Ls, Rs	25%
7.1.4	5.0	L, R, C, Ls, Rs, Ls, Rs, L, R, Ls, Rs	48%

For this process, a local copy of the original multichannel audio is first mixed into the lower density layout container using the downmixing schemes indicated in Table 7.3-3, excluding the LFE channels since no reflections are computed for that channel. The reflections coefficients, calculated by the shoebox model on the lower density layout, are then used to compute reflections as described in clause 7.3.5.5, on the content of the lower density layout container mix. The resulting reflections signals are panned back in the original layout, to the channel closest to the direction of arrival computed by the shoebox, and mixed with the original input audio, including LFE. In practice, this process results in a less dense acoustic reflection field that saves up to ~50% of computational cost, while preserving the directional resolution determined by the original layout.

A summary of the process is depicted in Fig. 7.3-7, showing the high-level process pipeline of downmixing, early reflections shoebox calculations and ER processing using the low-complexity layout, followed by panning and mixing back in the original layout.

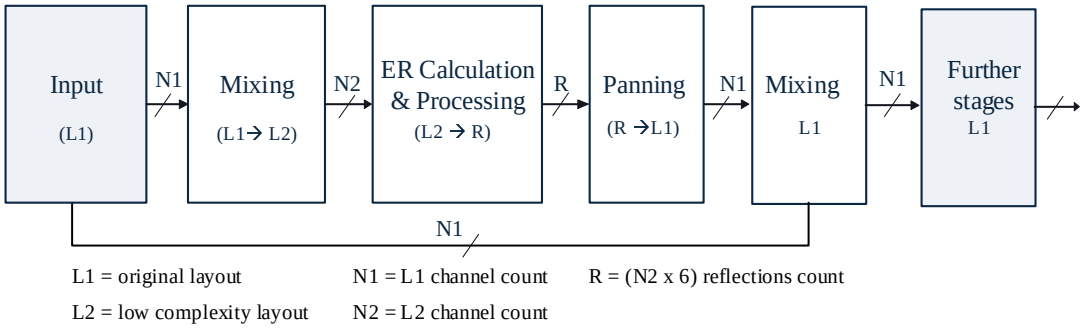


Figure 7.3-7: Low-complexity mode processing flow

7.4 Rendering control

7.4.1 Rendering control overview

Rendering control means allow for real-time control of rendered audio and for customizing the immersive experience. This clause discusses on two groups of rendering control means: scene and listener orientation control and rendering customization control. The basic aspects of scene and listener orientation are provided in clause 7.4.2. The scene and listener orientation control include means to track, compensate, modify, and process scene orientation, listener pose handling and capture device orientation handling, including several orientation tracking or compensation mechanisms, and combining the above. These aspects are discussed in clauses 7.4.3 through 7.4.6. The customization control mechanisms include HRTF and BRIR sets handling for binauralization (clause 7.4.7) and room acoustics parameters handling for room acoustics synthesis (clause 7.4.8). All the above are applicable to binaural rendering to support immersive experience. Next to that, custom loudspeaker layout control is discussed in clause 7.4.9.

7.4.2 Scene and listener orientation

7.4.2.1 Scene orientation

The spherical coordinates θ and ϕ correspond to the azimuth and elevation of a point on the surface of the unit sphere, respectively. Values for azimuth θ are positive left and are expected to be in the range from -180° (exclusive) to 180°. Values for elevation ϕ are positive up and are expected to lie in the range from -90° to 90°. Any values outside these ranges will be wrapped to within these ranges. Loudspeakers are positioned on the surface of the unit sphere, audio objects may in addition have a radius and directivity (cf. clauses 5.6.4.3 and 7.2.2.2.7).

Scene orientation control can be used for modifying the scene, e.g., in response to information about the default scene orientation or in response to potential capture device orientation changes. If available at the renderer, such orientation parameters can be used to rotate the rendered immersive scene. This can for example be done to compensate for a change of the capture device orientation in case such a compensation has not already been carried out in a preceding processing step, e.g., during capture at the sending device. This control can also be used to undo the compensation of a capture device orientation change, e.g., in case the application requires that capture device orientation changes have a corresponding effect in the rendered immersive scene. Note that compensation of a capture device orientation change means that the immersive scene is displaced in the same rotation sense as the capture device. Undoing a compensation means that the scene is displaced by the inverse rotation of capture device. Details of scene orientation control with such kind of external orientation input are described in clause 7.4.5. When used in combination with head tracking, the corresponding details are described in clause 7.4.6.

7.4.2.2 Listener orientation

The listener orientation is defined with respect to the coordinate system described in clause 7.4.2.1. By default the listener's head is positioned at the origin (0,0,0) and faces towards the positive x-axis (1, 0, 0). This can be described by a vector $p_{forward}$ which is therefore $[1, 0, 0]$. The listener's head only rotates around the origin and may not be repositioned for three degrees of freedom cases. IVAS codec also supports six degrees of freedom where the listener can move away from the origin over three axes in Cartesian coordinates as explained in clause 7.4.3.1. Listener orientation may be modified by the orientation tracking module which is further described in clause 7.4.4.

7.4.3 Head tracking

7.4.3.1 Head tracking via scene displacement

Head tracking in the IVAS codec is implemented via scene displacement. The coordinate system is depicted in Figure 7.4-1 the immersive scene. To track head movement of the user, the scene is displaced by the **inverse** rotation of the listener's head. For example, if the listener orientation points towards the front center speaker of a 5.1 layout and is updated to point towards the front-right loudspeaker (cf. clause 4.3.2) at -30° azimuth, the scene must rotate +30° in yaw around the z-axis to reposition the speaker, as if it were the front center loudspeaker directly in front$\hat{\boldsymbol{\imath}} + y\hat{\boldsymbol{\jmath}} + z\hat{\boldsymbol{k}}$

where $\mathbf{1}, \hat{\boldsymbol{\imath}}, \hat{\boldsymbol{\jmath}}$ and $\hat{\boldsymbol{k}}$ are basis vectors. The IVAS_QUATERNION structure used for this data is:

```
typedef struct
{
    float w, x, y, z;

} IVAS_QUATERNION;
```

A special value of -3 (which cannot occur for unit quaternions) for *w* can be used to indicate that the values x, y and z correspond to Euler angles of yaw, pitch and roll respectively.

IVAS codec supports six degrees of freedom with optional listener position parameters *pos.x, pos.y* and *pos.z* in the head tracking structure. The default values for the listener position parameters are set to 'zero' placing the listener at the origin. Listener position parameters are defined in Cartesian coordinates and no limit is set for the movement of the listener.

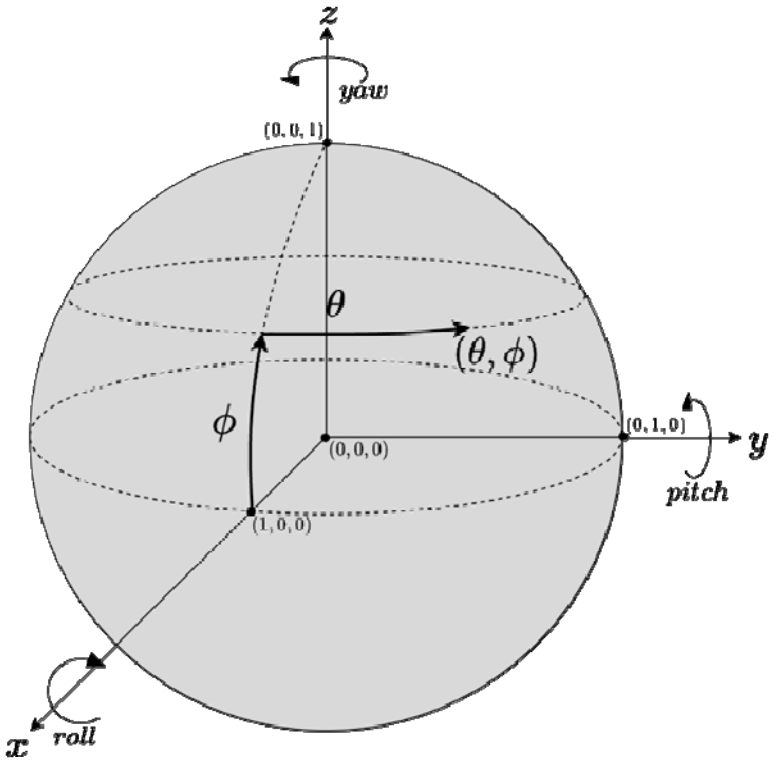


Figure 7.4-1: IVAS Coordinate system

7.4.3.2 Conversion from Euler angles to quaternions

The Euler angles of yaw, pitch and roll are converted to a quaternion representation, with the elements w, x, y and z according to the Equations (7.4-1) to (7.4-4).

$$w = \cos\left(\frac{yaw}{2}\right)\cos\left(\frac{pitch}{2}\right)\cos\left(\frac{roll}{2}\right) + \sin\left(\frac{yaw}{2}\right)\sin\left(\frac{pitch}{2}\right)\sin\left(\frac{roll}{2}\right) \quad (7.4\text{-}1)$$

$$x = \cos\left(\frac{yaw}{2}\right)\cos\left(\frac{pitch}{2}\right)\sin\left(\frac{roll}{2}\right) - \sin\left(\frac{yaw}{2}\right)\sin\left(\frac{pitch}{2}\right)\cos\left(\frac{roll}{2}\right) \quad (7.4\text{-}2)$$

$$y = \sin\left(\frac{yaw}{2}\right)\cos\left(\frac{pitch}{2}\right)\sin\left(\frac{roll}{2}\right) + \cos\left(\frac{yaw}{2}\right)\sin\left(\frac{pitch}{2}\right)\cos\left(\frac{roll}{2}\right) \quad (7.4\text{-}3)$$

$$z = \sin\left(\frac{yaw}{2}\right)\cos\left(\frac{pitch}{2}\right)\cos\left(\frac{roll}{2}\right) - \cos\left(\frac{yaw}{2}\right)\sin\left(\frac{pitch}{2}\right)\sin\left(\frac{roll}{2}\right) \quad (7.4\text{-}4)$$

7.4.3.3 Rotation matrix from quaternions

Using the quaternion elements w, x, y and z, a rotation matrix for the real cartesian 3D space may be computed as described in Equation (7.4-5).

$$\begin{matrix} w^2 + x^2 + y^2 - z^2 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & w^2 - x^2 - y^2 + z^2 \end{matrix} \quad (7.4\text{-}5)$$

7.4.3.4 Application of rotations

7.4.3.4.1 Rotation in the spatial domain

Rotation in the spatial domain is used for channel-based and object-based audio signals. For object-based audio, the related metadata positions are rotated, whereas for channel-based audio the loudspeakers are treated as virtual objects and re-panned onto the existing layout.

Given a coordinate pair θ and ϕ for the unit sphere to which a rotation must be applied, the cartesian position vector is computed:

$$x = cos(\theta)\, cos(\phi)$$

$$y = sin(\theta)\, cos(\phi)$$

$$z = sin(\phi)$$

Then the rotation matrix is computed from the head tracking data (Equation (7.4-5)) and applied to the position vector to obtain a rotated position. This position is then converted back to spherical coordinates yielding the rotated spherical coordinates.

For object-based audio, updating the metadata with the rotated object position is sufficient and may be consumed by further rendering steps as necessary. For channel-based audio, each individual loudspeaker position is rotated using this method and gains for the rotated position of the loudspeaker are computed using EFAP (cf. clause 7.2.1.3). The individual loudspeaker gains together are used to construct a rotation matrix for the corresponding loudspeaker layout, which is applied to perform rotation of the audio signal.

7.4.3.4.2 Rotation in the spherical harmonic domain

The rotation matrices for the spherical harmonic domain are obtained from the real-space rotation matrices (Equation (7.4-5)) using the recurrence relations described in [19] based on the order of ambisonics. With this a rotation matrix of dimension $(N+1)^2 \times (N+1)^2$ is obtained for a given ambisonic order N. This spherical harmonic rotation matrix is then applied via matrix multiplication to the ambisonics channels to produce a rotated representation.

7.4.4 Orientation tracking

7.4.4.1 Orientation tracking introduction

In addition to supporting head rotation processing, the IVAS renderer supports a number of modes for listener orientation tracking. The listener orientation tracking refers to a set of methods used to provide or estimate listener's frontal orientation (being listener's head neutral position or torso position). Such a listener's frontal orientation is further referred to as reference orientation. The following orientation tracking modes are supported:

- External reference orientation,
- External reference vector orientation,
- External reference levelled vector orientation,
- Adaptive long-term average reference orientation.

The externally received or computed average orientation is eventually combined with the head rotation data. Such combined rotation is applied in the rendering to create the final rotated output. The relationship between head rotation and torso orientation is shown in Figure 7.4-2. Note that for the sake of readability, the view from the top is depicted. In actual implementations rotation in all dimensions is considered.

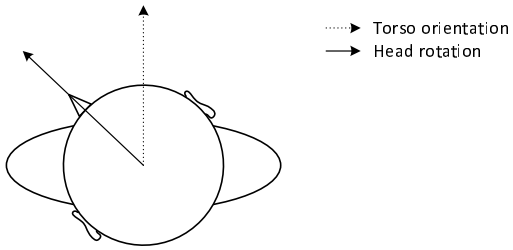


Figure 7.4-2: Head rotation vs torso orientation

7.4.4.2 External reference orientation

In case of external reference orientation mode (identified as HEAD_ORIENT_TRK_REF) the IVAS decoder/renderer is provided with two input orientations: head rotation $Rot_{in}[m]$ from the head tracking device and reference rotation $Rot_{ref}[m]$. The reference rotation can for instance originate from a sensor attached to the listener torso or a mobile phone in the listener's pocket. The reference rotation is subtracted from the input head rotation.

$$Rot_{rel}[m] = Rot_{in}[m] - Rot_{ref}[m] \quad (7.4\text{-}6)$$

The relative rotation quaternion $Rot_{rel}[m]$ is eventually used as the control input for the binaural renderer.

7.4.4.3 External reference vector orientation

The listener orientation in the IVAS coordinate system is described in clause 7.4.2.2 which defines the vector $p_{forward}$. The The input parameters to the reference vector orientation (identified as HEAD_ORIENT_TRK_REF_VEC) orientation tracking modes are:

- The absolute position of the listeners head in Cartesian 3D coordinates ($p_{listenerabs}$).
- The absolute position of an acoustic reference (p_{refabs}). In case of camera-based head tracking by a UE, where the UE should act as the acoustic reference direction, this would be the position of the phone in Cartesian 3D coordinates.
- The absolute head orientation of the listener ($r_{listenerabs}$).

The position of the listener and the reference must refer to a common coordinate system, e.g., an Earth-fixed coordinate system.

Using the positions of the listeners head and the reference, a vector spanning from the listener's head to the reference position is determined ($p_{acousticfront}$).

$$p_{acousticfront} = p_{refabs} - p_{listenerabs} \quad (7.4\text{-}7)$$

The absolute head orientation of the user must be transformed such that the resulting head orientation (r_{result}) causes the binaural rendering step, using this processed head orientation, will produce an audio signal, where an audio object with 0-degree azimuth and 0-degree elevation would get rendered in the direction of the reference position. Therefore, this vector from the listeners head to the reference acts as the acoustic front of the system in HEAD_ORIENT_TRK_REF_VEC mode.

This shall be be implemented by normalizing $p_{acousticfront}$ and $p_{ivasforward}$:

$$p_{acousticfront\_norm} = \frac{p_{acousticfront}}{|p_{acousticfront}|} \quad (7.4\text{-}8)$$

and

$$p_{forward\_norm} = \frac{p_{forward}}{|p_{forward}|} \quad (7.4\text{-}9)$$

And then building the cross product and the dot product of both vectors:

$$cross\_product = p_{acousticfront\_norm} \times p_{ivasforward\_norm} \quad (7.4\text{-}10)$$

and

$$dot\_product = p_{acousticfront\_norm} \cdot p_{ivasforward\_norm} \quad (7.4\text{-}11)$$

When the dot product is close to -1 (i.e., *dot_product* < *-0.999999*), the two vectors are almost directly opposite to each other. In such cases, the cross product becomes near-zero, leading to an ambiguous rotational axis. To handle this, a predefined quaternion representing a 180-degree rotation is used:

$$r_{refrot} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{bmatrix} \quad (7.4\text{-}12)$$

Otherwise, the reference rotation quaternion (r_{refrot}) is calculated as:

$$r_{refrot} = \begin{bmatrix} \sqrt{\left(\left|p_{acousticfront\_norm}\right|^2\right)\left(\left|p_{forward\_norm}\right|^2\right)} + dot\_product \\ cross\_product_x \\ cross\_product_y \\ cross\_product_z \end{bmatrix} \quad (7.4\text{-}13)$$

The $r_{refrot}.w$ part of the quaternion represents the combined magnitudes of the vectors and their dot product. Since the vectors are normalized, the magnitudes are both 1, resulting in:

$$r_{refrot}.w = 1 + dot\_product \quad (7.4\text{-}14)$$

The resulting Quaternion is normalized:

$$r_{refrot\_norm} = \frac{r_{refrot}}{|r_{refrot}|} \quad (7.4\text{-}15)$$

The inverse of the resulting normalized reference orientation (r_{refrot_norm}) shall then be used to rotate the absolute listener orientation:

$$r_{result} = {r_{refrot\_norm}}^{-1} \times r_{listenerabs} \quad (7.4\text{-}16)$$

where the $\times$ operator denotes quaternion product.

7.4.4.4 External reference levelled vector orientation

In the reference vector level mode (identified by `HEAD_ORIENT_TRK_REF_VEC_LEV`) orientation tracking mode, the processing is identical to reference vector mode (identified by `HEAD_ORIENT_TRK_REF_VEC`) as specified in clause 7.4.4.3, except that the difference in height between the head of the user and the reference is ignored. This is achieved by aligning the up/down axis input values of listener and reference position to the same value. Therefore, the $p_{acousticfront}$ vector in this orientation tracking mode is defined as:

$$p_{acousticfront} = \begin{bmatrix} {p_{refabs}}_x \\ {p_{refabs}}_y \\ {p_{listenerabs}}_z \end{bmatrix} - \begin{bmatrix} {p_{listenerabs}}_x \\ {p_{listenerabs}}_y \\ {p_{listenerabs}}_z \end{bmatrix}. \quad (7.4\text{-}17)$$

7.4.4.5 Adaptive long-term average reference orientation

The adaptive long-term average reference orientation mode (identified by `HEAD_ORIENT_TRK_AVG`) is applicable in the cases that no external reference for a pose is provided directly. This can be especially efficient to compensate for involuntary rotation, such as might occur while a user is travelling (e.g., taking a turn on a train, while walking, etc.). As a result, a user perceives that the virtual sound sources stay consistent in the scene in relation to the frontal (torso) orientation, also in the case of involuntary rotation.

A general illustration of such adaptation of the reference (torso) orientation is provided in Figure 7.4-3. For the sake of readability, the view from the top is depicted. Also, change of reference orientation in limited to horizontal plane, while in the implementation all dimensions are considered.

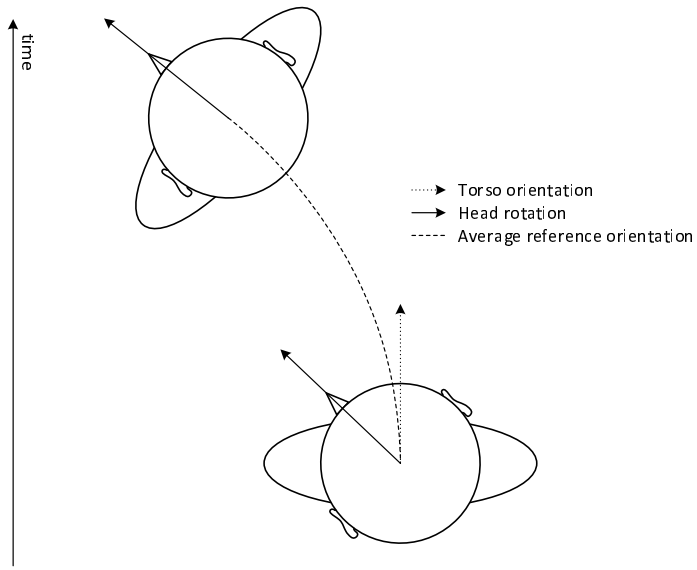


Figure 7.4-3: Adaptive long-term average reference orientation in a function of time

Input head rotation $Rot_{in}[m]$ is provided by a head tracker with a sensor that captures the motion of the head. A rotation angle relative to a reference direction is calculated, where the reference direction is dependent on the movement of the head.The head rotation relative to the reference $Rot_{rel}[m]$ for frame m is computed by applying high-pass filter to the input head rotation $Rot_{in}[m]$. This is achieved by subtracting long term average rotation from the input head rotation. The reference rotation estimate is depending on the movement detected by the head rotation, as measured by the sensor of the head tracker:

$$Rot_{rel}[m] = Rot_{in}[m] - Rot_{avg}[m] \quad (7.4\text{-}18)$$

where the long-term average orientation is obtained by low pass filtering of the input head rotations $Rot_{in}(t)$. Adaptive averaging is applied to adjust to larger rotation deviations faster than for small ones, making the averaging algorithm less sensitive to small, quick head movements. Such adaptive average rotation is computed by applying an adaptive IIR lowpass filter:

$$Rot_{avg}[m] = \alpha Rot_{in}[m] + (1-\alpha)Rot_{avg}[m-1] \quad (7.4\text{-}19)$$

where the adaptive parameter α:

$$\alpha = \sin\left(2\pi \frac{f_c}{f_s}\right) \quad (7.4\text{-}20)$$

with the cutoff frequency f_c is linearly interpolated between its minimum and maximum values:

$$f_c = f_{c,min} + RDR\left(f_{c,max} - f_{c,min}\right) \quad (7.4\text{-}21)$$

and using relative direction ratio RDR:

$$RDR = \min\left(\frac{|Rot_{rel}[m-1]|}{Rot_{max}}, 1\right) \quad (7.4\text{-}22)$$

and f_s equals the audio processing sample rate.

The threshold values of adaptation parameters are set as follows:

$$f_{c,min} = \frac{1}{30} Hz,$$

$$f_{c,max} = \frac{1}{8} Hz,$$

$$r_{max} = 90°.$$

The processing is performed in the quaternion domain. Firstly, the adaptive average rotation quaternion $Rot_{avg}[m]$ is computed using spherical linear interpolation (slerp) between the input rotation $Rot_{in}[m]$ and previous average rotation $Rot_{avg}[m-1]$:

$$Rot_{avg}[m] = \frac{Rot_{in}[m]\sin(\alpha\varphi) + Rot_{avg}[m-1]\sin\big((1-\alpha)\varphi\big)}{sin(\varphi)} \quad (7.4\text{-}23)$$

where the angle

$$\varphi = \cos^{-1}\big(Rot_{in}[m] \cdot Rot_{avg}[m-1]\big) \quad (7.4\text{-}24)$$

and the $\cdot$ operator denotes a dot product.

The adaptive average rotation quaternion Rot_{avg} is then normalized. The resulting relative rotation quaternion Rot_{rel} is computed as:

$$Rot_{rel}[m] = \frac{1}{\widehat{Rot}_{avg[m]}} \times Rot_{in}[m] \quad (7.4\text{-}25)$$

where the $\times$ operator denotes quaternion product. The adaptive parameter α is eventually updated as shown in equation (7.4-27). The relative rotation quaternion $Rot_{rel}[m]$ is eventually used as the control input for the binaural renderer.

7.4.5 External orientation input handling

7.4.5.1 Overview

The external orientation input provides to the IVAS renderer any orientation information separate from the head orientation (rotation) data of the listener. The external orientation data, when available, is therefore processed and applied in addition to the head-tracking data, which is described in clause 7.4.3.

Figure 7.4-4 presents a block diagram describing the external orientation inputs at the IVAS renderer.

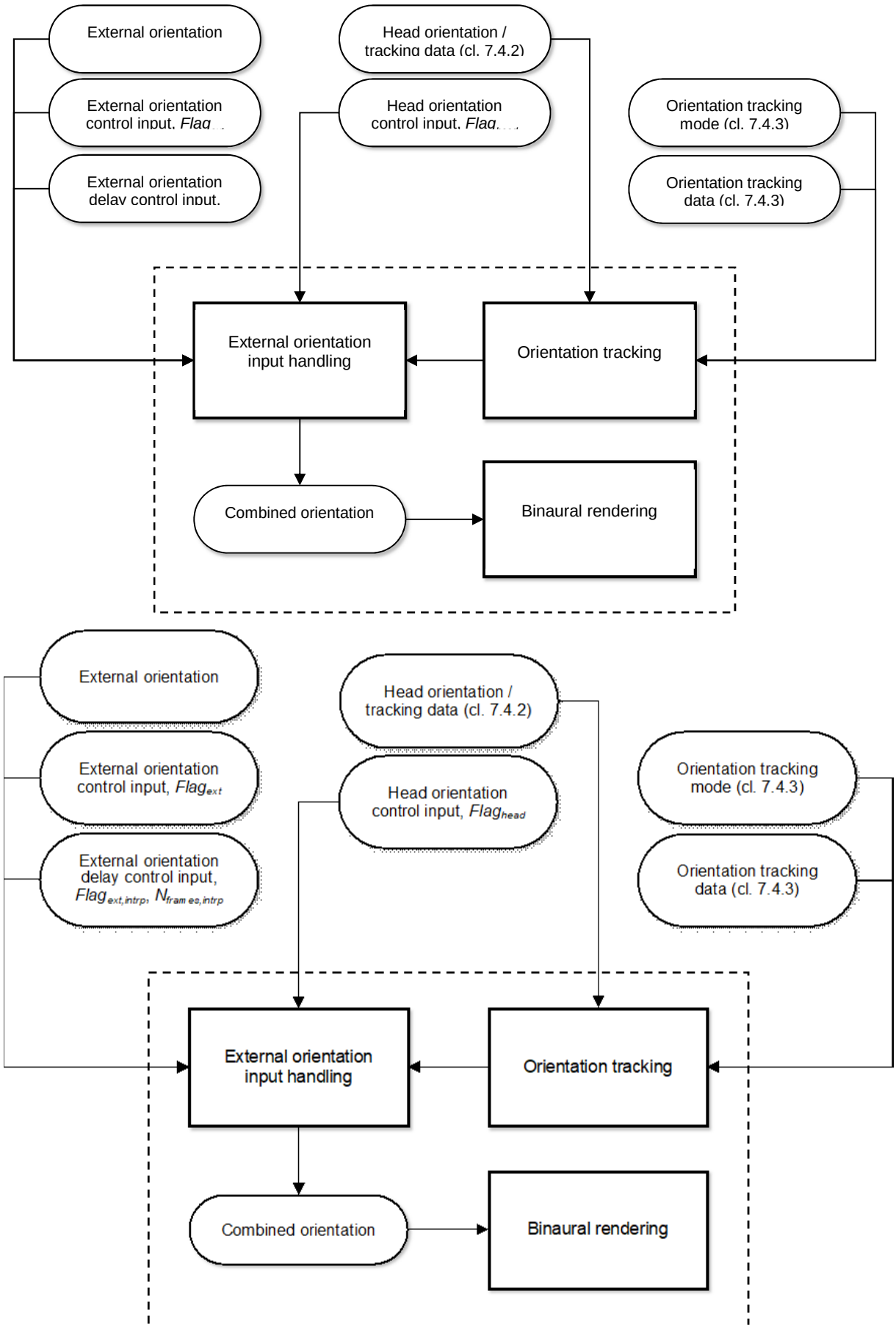


Figure 7.4-4: Block diagram of external orientation input handling for binaural rendering

For example, a capture device (e.g., sending UE) that can comprise, e.g., a microphone or a microphone array captures or otherwise obtains one or more (spatial) audio signals part of a spatial audio scene. In addition, the sending UE may determine or otherwise obtain orientation information that relates to this spatial audio scene. The spatial audio scene, represented by one or more audio signals, is encoded by the IVAS encoder according to input format specific operation, as described in clause 5 (e.g., according to clause 5.4 for SBA operation or clause 5.5 for MASA format operation), and transmitted to the receiving UE (e.g., using IVAS RTP payload, defined in Annex A). The UE receives the transmitted IVAS stream comprising the encoded audio signal(s) part of the spatial audio scene. Additionally, the receiver may obtain the encoded orientation information which is associated with the capture device (e.g., sending UE). The receiver then decodes the encoded audio signal(s), as described in clause 6, and the encoded orientation information. The orientation information may comprise information associated with, e.g., a default scene orientation, orientation of the capture device (e.g., sending UE), and orientation compensation. This decoded orientation information is then provided to the IVAS renderer as external orientation input data, and the renderer uses it to process the audio signal(s) as part of the rendering.

The external orientation information may, e.g., include (intended or default) scene orientations, device orientations, orientation compensations, orientations of a user operating a capture device (e.g., a sending UE), or other such reference orientations that may be used to guide or modify the rendering processing. The (intended or default) scene orientations may comprise the orientation of the scene at the sending UE or the local scene orientation at the receiving UE. A default scene orientation or an intended scene orientation may comprise, e.g., an orientation determined by the sending UE, which is transmitted as a guidance for rendering orientation of the decoded audio scene. The scene orientation data is thus intended to modify the rendering orientation of the spatial audio. The orientation compensation may comprise, e.g., orientation data for removing from, or adding to, rendering the orientation (rotational) information of the capture device (e.g., a sending UE). The orientation compensation may also comprise a reference orientation (e.g., the viewing direction) of the capture device relative to the scene. A reference orientation, related to a sender UE, e.g., may also be used to define a scene orientation. A reference orientation may also identify a global orientation reference. The device orientations may comprise the orientation of the capture device (on send side) or the orientation of the playback device (on receive side).

For example, an intended scene orientation may be provided to renderer to rotate the spatial audio scene such that a desired direction in the spatial audio scene is to the front of the listener (assuming no head-tracking data corresponding to a listener's head turn is being provided).

For example, orientation of a capturing device may be decoupled from the spatial audio scene before encoding (i.e., orientation change of capture device is being compensated at the time of audio capture) to generate modified spatial audio signals for encoding and thus for the receiving UE for presentation of the aural scene. The corresponding device orientation can also be transmitted to the receiving UE. The modified signals may then be presented (i.e., rendered) to the listener in the receiver space by modifying (or compensating) the transmitted spatial audio scene with the captured direction data.

The external orientation input to the IVAS renderer may be transmitted to the receiving UE or, e.g., determined at the receiving UE. Concerning the rendering, the external orientation input thus provides augmentation control parameters that are configured to control in part the rendering of the spatial audio scene by the renderer obtaining the corresponding spatial audio signals and augmentation control parameters. Typically, the rendering is in part controlled also by the head-tracking data. The same data can also be made available to (e.g., by transmitting, storing, or relaying through the external renderer interface), and applied at, any external renderer in addition to the internal IVAS renderer and the external IVAS renderer. If there exist multiple external orientations that are to be applied at the renderer, the multiple external orientations should be combined into a single external orientation to provide the external orientation data to the IVAS renderer.

7.4.5.2 Processing of the external orientation data

The external orientation data comprises of the external orientation information and the external orientation control information. The external orientation information (w, x, y, z) shall be provided as quaternions. The external orientation control information may comprise of signalling (e.g., a flag, a switch) to enable, disable, or freeze the application of the external orientation information and/or head orientation (rotation). Additionally, the external orientation control information may include activation signalling and time for indicating that the external orientation information shall be applied at a current frame or at a future frame.

The control signalling is indicated with $Flag_{head}(m)$ for head rotations and with $Flag_{ext}(m)$ for external rotations. A flag value of 0 disables the respective rotation in the rendering process, a flag value of 1 enables the rotation. A flag value of 2 freezes the respective rotation to a last rendered rotation. Based on the control flags, a rendering rotation of the head and/or external rotation is accordingly modified and then applied in the rendering of the spatial audio. If a

rotation flag is not present in the external orientation control information, the associated rotation shall be enabled by default (i.e., as if the flag would have a value of 1). This processing is described in more detail in clause 7.4.6.1.

For future frame appliance indication, the receiver may receive an orientation change data set as part of the external orientation data. The data set comprises orientation change value specifying a change to an orientation (e.g., a change to an orientation of a capture device with respect of the spatial audio scene comprising the audio signal) and an orientation change delay time for the change to the orientation (e.g., to the device orientation). In the external orientation data, the future appliance of external rotation is signalled with $Flag_{ext,intrp}(m)$, where a value of 1 indicates that the external rotation is changed after a period of time specified by an external orientation change delay time $N_{frames,intrp}(m)$. A value of 0 for $Flag_{ext,intrp}(m)$ indicates that the future appliance of external rotation is disabled, and that the external rotation should be applied immediately. The orientation change delay time is signalled or expressed with $N_{frames,intrp}(m)$ which indicates the number or units of audio frames (20 ms) of the audio signal after the associated external rotation is applied. During the transition period $N_{frames,intrp}(m)$ the external rotation is interpolated from the previously applied external rotation to the target external rotation. If the future appliance flag is not present in the external orientation control information, the future appliance of external rotation shall be disabled by default (i.e., as if the future appliance flag would have a value of 0). If the orientation change delay time is not present in the external orientation control information, the change delay time shall be set to zero (0). The interpolation processing is described in more detail in clause 7.4.6.

7.4.6 Combined rotations for rendering

7.4.6.1 Combining head and external rotations

A received external orientation $O_{ext}(m)$ is combined with head-tracking data (head rotation $Rot_{head}(m)$) in a rotation combiner to determine the rotation for rendering. Both the external orientation and the head orientation (rotation) are processed as quaternions in the rotation combiner. Before the combination process, the received external orientation $O_{ext}(m)$ shall be inverted to operate as a rotation, $Rot_{ext}(m) = {O_{ext}}^{-1}(m)$.

In case there is no external rotation present, or the external rotation signalling flag disables the external rotation $(Flag_{ext}(m) == 0)$, and there is head rotation present, the combined rotation $Rot_{com}(m)$ shall consist of the head rotation only, i.e., $Rot_{com}(m) = Rot_{head}(m)$.

If the external orientation control information indicates freezing of the head rotation $(Flag_{head}(m) == 2)$ and the freezing indicator is the first in a series (i.e., $Flag_{head,frozen} == 0$), the head rotation shall be frozen to the received head rotation value $Rot_{head,frozen} = Rot_{head}(m)$, the indicator flag for frozen head rotation shall be activated as $Flag_{head,frozen} = 1$ and the frozen head rotation value shall be used in the processing, $Rot_{head,applied}(m) = Rot_{head,frozen}$. In the subsequent processing subframes $m + d, d \in \mathbb{N}$ with $Flag_{head}(m + d) == 2$ and $Flag_{head,frozen} == 1$, the frozen head rotation value shall be used as the head rotation in the processing, $Rot_{head,applied}(m + d) = Rot_{head,frozen}$. If a head rotation signalling flag of $Flag_{head}(m) \neq 2$ is noticed, the frozen head rotation indicator flag shall be deactivated as $Flag_{head,frozen} = 0$ and the frozen head rotation shall be set to identity as $Rot_{head,frozen} = (1, 0, 0, 0)$.

If the external orientation control information indicates disabling of the head rotation $(Flag_{head}(m) == 0)$, the head rotation shall be disabled for the processing. If the external orientation control information indicates enabling of the head rotation $(Flag_{head}(m) == 1)$ or the head rotation signalling flag is missing, the received head rotation shall be used in the processing, $Rot_{head,applied}(m) = Rot_{head}(m)$.

If the external orientation control information indicates freezing of the external rotation $(Flag_{ext}(m) == 2)$ and the freezing indicator is the first in a series (i.e., $Flag_{ext,frozen} == 0$), and the future frame application is disabled $\left(Flag_{ext,intrp}(m) == 0\right)$, the external rotation shall be frozen to the received external rotation value $Rot_{ext,frozen} = Rot_{ext}(m)$, the indicator flag for frozen external rotation shall be activated as $Flag_{ext,frozen} = 1$ and the frozen external rotation value shall be used in the processing, $Rot_{ext,applied}(m) = Rot_{ext,frozen}$.. In the subsequent processing subframes $m + d, d \in \mathbb{N}$ with $Flag_{ext}(m + d) == 2$ and $Flag_{ext,frozen} == 1$, the frozen external rotation value shall be used as the external rotation in the processing, $Rot_{ext,applied}(m + d) = Rot_{ext,frozen}$. If an external rotation signalling flag of $Flag_{ext}(m) \neq 2$ is noticed, the frozen external rotation indicator flag shall be deactivated as $Flag_{ext,frozen} = 0$ and the frozen external rotation shall be set to identity as $Rot_{ext,frozen} = (1, 0, 0, 0)$.

If the external orientation control information indicates disabling of the external rotation $(Flag_{ext}(m) == 0)$, the external rotation shall be disabled for the processing. If the external orientation control information indicates enabling

of the external rotation ($Flag_{ext}(m) == 1$) or the external orientation signalling flag is missing, the received external rotation shall be used in the processing, $Rot_{ext,applied}(m) = Rot_{ext}(m)$.

If the external orientation future frame application is enabled $\left(Flag_{ext,intrp}(m) == 1\right)$, the external rotation is processed as described in clause 7.4.6.2.

If both the head rotation and the external rotation are present and indicated to be used in the processing, the rotations $Rot_{head,applied}(m)$ and $Rot_{ext,applied}(m)$ shall be combined into a single rotation before applying them in the processing. The combination of the two rotations are given as:

$$Rot_{com}(m) = Rot_{ext,applied}(m) \times Rot_{head,applied}(m) \quad (7.4\text{-}26)$$

where the × operator denotes quaternion product.

In case there is no head rotation present or the head rotation signalling flag disables the head rotation ($Flag_{head}(m) == 0$) and there is external rotation present and the external rotation is enabled ($Flag_{ext}(m) == 1\ or\ Flag_{ext}(m) == 2$), the combined rotation shall consist of the external rotation only, $Rot_{com}(m) = Rot_{ext,applied}(m)$.

The combined rotation $Rot_{com}(m)$ in quaternions is transformed into a rotation matrix $R_{com}(m)$ as shown in equation 7.4-5.

If there are no head rotation and no external orientation present in a subframe, the combined rotation variables shall be set to their initial values as presented in Table 7.4-1.

$Flag_{com}(m)$ indicates if the combined rotation matrix $R_{com}(m)$ is enabled or disabled for the processing. A value of $Flag_{com}(m) == 1$ indicates that $R_{com}(m)$ shall be used in the processing and a value of $Flag_{com}(m) == 0$ indicates that $R_{com}(m)$ shall not be used in the processing. If there is head rotation present and no external rotation present for a subframe m, the $Flag_{com}(m)$ shall be set to 1. If there is external rotation present and no head rotation present for a subframe m and the external orientation control information indicates enabling the external rotation ($Flag_{ext}(m) == 1$ or $Flag_{ext}(m) == 2$), the $Flag_{com}(m)$ shall be set to 1. If there is both head rotation and external rotation present for a subframe m and the external orientation control information indicates enabling either one or both of the rotations (($Flag_{ext}(m) == 1$ or $Flag_{ext}(m) == 2$) and/or ($Flag_{head}(m) == 1$ or $Flag_{head}(m) == 2$)), the $Flag_{com}(m)$ shall be set to 1. Otherwise, the $Flag_{com}(m)$ shall be set to 0. $Flag_{com}(m)$ is used in the various rendering processes to check if the combined rotations shall be applied or not.

The applied listener position for subframe m $\left(Pos_{listener,applied}(m)\right)$ is also controlled in the rotation combiner. If head tracking data is available for subframe m, the applied listener position shall be set as the received listener position, $Pos_{listener,applied}(m) = Pos_{listener}(m)$. If both head tracking and external rotation data is not available for subframe m, the applied listener position shall be set to origo, $Pos_{listener,applied}(m) = (0, 0, 0)$. The listener position is processed as (x, y, z) coordinates.

7.4.6.2 External rotation interpolation

If the external orientation future frame application is enabled $\left(Flag_{ext,intrp}(m) == 1\right)$, the received external rotation $Rot_{ext}(m)$ should be applied in the future. The time of appliance is indicated by $N_{frames,intrp}(m)$ which indicates that after $N_{frames,intrp}(m)$ frames (20 ms) the received external rotation $Rot_{ext}(m)$ is reached. The time of appliance in subframes is $4 * N_{frames,intrp}(m)$. During the transition time $\left[m, m + 4 * N_{frames,intrp}(m)\right]$, the external rotation is interpolated from a start rotation $Rot_{ext,start}$ to a target rotation $Rot_{ext,target}$. The start rotation depends on the state of the rendering process. The target rotation is set as the received external rotation at the beginning of the interpolation process $Rot_{ext,target} = Rot_{ext}(m)$.

The progress of an interpolation process is observed with an interpolation increment $Intrp_{incr}$, an interpolation coefficient $Intrp_{coeff}(m)$ and an interpolation progress flag $Flag_{intrp,ongoing}$. The interpolation increment is a linear increment based on to the ratio of the external orientation change delay time $N_{frames,intrp}(m)$, calculated as:

$$Intrp_{incr} = \frac{1}{4 * N_{frames,intrp}(m)} \quad (7.4\text{-}27)$$

At the beginning of a new interpolation process, the interpolation coefficient is set as $Intrp_{coeff}(m) = Intrp_{incr}$ and the interpolation progress flag is activated as $Flag_{intrp,ongoing} = 1$. After each interpolation step, the interpolation coefficient is incremented as $Intrp_{coeff}(m+1) = Intrp_{coeff}(m) + Intrp_{incr}$ for the next subframe.

At each subframe m with the future frame application enabled $\left(Flag_{ext,intrp}(m) == 1\right)$, a check for a current ongoing external rotation interpolation is performed. If the interpolation progress flag is active ($Flag_{intrp,ongoing} == 1$), the interpolation coefficient $Intrp_{coeff}(m) \leq 1.0$ and the set target rotation is approximately the same as the received external rotation at the current subframe $Rot_{ext,target} \approx Rot_{ext}(m)$, a current interpolation process is continued. The rotations $Rot_{ext,start}$ and $Rot_{ext,target}$ are interpolated (as in Equation (7.4-23)):

$$Rot_{ext,intrp}(m) = \frac{Rot_{ext,start} * \sin\left(\left(1 - Intrp_{coeff}(m)\right) * \varphi\right) + Rot_{ext,target} * \sin\left(Intrp_{coeff}(m) * \varphi\right)}{sin(\varphi)} \quad (7.4\text{-}28)$$

where φ is the angle between the start and target rotations given as:

$$\varphi = \arccos\left(Rot_{ext,start} \cdot Rot_{ext,target}\right) \quad (7.4\text{-}29)$$

where the $(\cdot)$ operator denotes quaternion dot product. The interpolated rotation is normalized before applying the rotation in the processing:

$$Rot_{ext,applied}(m) = \widehat{Rot}_{ext,intrp}(m) \quad (7.4\text{-}30)$$

In the interpolation process, an increment of rotation change is applied to the starting rotation by audio subframe basis for the period of time specified by the external orientation change delay time $N_{frames,intrp}(m)$. The increment of rotation change is determined linearly as the rotation of a ratio of the rotation change to the external change orientation delay time. The rotation change is the change in rotation from the starting rotation to the target rotation. I.e., the interpolation coefficient $Intrp_{coeff}(m)$ determines how many increments of rotation change are applied to the starting rotation $Rot_{ext,start}$ to determine the interpolated rotation for a subframe m.

After the interpolation step, the interpolation coefficient is incremented as $Intrp_{coeff}(m+1) = Intrp_{coeff}(m) + Intrp_{incr}$. Two rotations are approximated to be the same if the following conditions are all fulfilled:

$$Rot_1 \approx Rot_2, \quad if \begin{cases} |w_1 - w_2| \leq 0.05 \\ |x_1 - x_2| \leq 0.05 \\ |y_1 - y_2| \leq 0.05 \\ |z_1 - z_2| \leq 0.05 \end{cases} \quad (7.4\text{-}31)$$

where $(w_1,\ x_1,\ y_1, z_1)$ represents the first rotation Rot_1 in quaternions and $(w_2,\ x_2,\ y_2, z_2)$ represents the second rotation Rot_2 in quaternions.

If a current interpolation is not in progress $\left(Flag_{intrp,ongoing} == 0\right)$ or the current interpolation process is completed $\left(Intrp_{coeff}(m) > 1.0\right)$ or a new external target rotation is set $\left(Rot_{ext,target} \not\approx Rot_{ext}(m)\right)$, the interpolation variables are set as $Flag_{intrp,ongoing} = 0$, $Intrp_{coeff}(m) = 1.0$ and $Intrp_{incr} = 1.0$ and a new interpolation process can be started.

At the beginning of the interpolation process, the time of appliance $N_{frames,intrp}(m)$ is forced to a range of $[0, 500]$ frames (20 ms) (i.e., to a range of $[0, 4 * 500]$ subframes (5 ms)). For $N_{frames,intrp}(m) < 0$ the time of appliance is set to zero (0), and for $N_{frames,intrp}(m) > 500$ the time of appliance is set to 500. If $N_{frames,intrp}(m) == 0$, the target rotation is applied immediately $\left(Rot_{ext,applied} = Rot_{ext,target}\right)$, the interpolation process is skipped and the interpolation variables are set as $Flag_{intrp,ongoing} = 0$, $Intrp_{coeff}(m) = 1.0$ and $Intrp_{incr} = 1.0$.

If $N_{frames,intrp}(m) > 0$, a new interpolation process is started. If the current target rotation and the received external rotation are not approximately the same $\left(Rot_{ext,target} \not\approx Rot_{ext}(m)\right)$, new interpolation values are calculated and set. The target rotation is set as the received external rotation, $Rot_{ext,target} = Rot_{ext}(m)$. The starting rotation depends on the codec state. If the external rotation was disabled in the previous subframe, the starting rotation shall be set to identity, i.e., $Rot_{ext,start} = (1, 0, 0, 0), if\ Flag_{ext,intrp}(m-1) == 0$. If the external rotation was frozen in the previous subframe $\left(Flag_{ext,intrp}(m-1) == 2\right)$ or the current external rotation is frozen $\left(Flag_{ext,intrp}(m) == 2\right)$, the starting rotation shall be set to the frozen external rotation, $Rot_{ext,start} = Rot_{ext,frozen}$. Otherwise, the starting

rotation shall be set to the external rotation from the previous subframe, $Rot_{ext,start} = Rot_{ext}(m-1)$. The interpolation increment shall be calculated as in equation (7.4-27) and the interpolation coefficient shall be set as $Intrp_{coeff}(m) = Intrp_{incr}$. The interpolation progress flag shall be set as $Flag_{intrp,ongoing} == 1$. The rotations $Rot_{ext,start}$ and $Rot_{ext,target}$ shall be interpolated as in equations (7.4-28), (7.4-29) and (7.4-30) and the interpolated rotation shall be used as the applied external rotation $Rot_{ext,applied}(m)$ for the current subframe. The interpolation coefficient shall be incremented as $Intrp_{coeff}(m+1) = Intrp_{coeff}(m) + Intrp_{incr}$.

If in the beginning of a new interpolation process the current target rotation and the received external rotation are approximately the same $\left(Rot_{ext,target} \approx Rot_{ext}(m)\right)$, the calculations for the interpolation values are skipped and the old values are used. The interpolation progress flag shall be set as $Flag_{intrp,ongoing} == 1$. The rotations $Rot_{ext,start}$ and $Rot_{ext,target}$ shall be interpolated as in equations (7.4-28), (7.4-29) and (7.4-30) and the interpolated rotation shall be used as the applied external rotation $Rot_{ext,applied}(m)$ for the current subframe. The interpolation coefficient shall be incremented as $Intrp_{coeff}(m+1) = Intrp_{coeff}(m) + Intrp_{incr}$.

7.4.6.3 Initial values for combined rotation variables

The variables in the rotator combiner shall be initialized as given in Table 7.4-1.

Table 7.4-1: Initial values for combined rotations

Variable	Initial value
$Intrp_{coeff}$	1.0
$Intrp_{incr}$	1.0
$Flag_{intrp,ongoing}$	0
$Rot_{ext,start}$	(1, 0, 0, 0)
$Rot_{ext,target}$	(1, 0, 0, 0)
Rot_{com}	(1, 0, 0, 0)
R_{com}	Identity
$Flag_{com}$	0
$Pos_{listener,applied}$	(0, 0, 0)

7.4.7 HRTF and BRIR sets

7.4.7.1 HRTF and BRIR latency

In case of multi-channel input with LFE, LFE input is added to the binaural output using a delay line and a gain:

$$O^{l,r} = O^{l,r} + G_{lfe} * delay(I_{lfe}, binraural\_latency\_s)$$

I_{lfe} is the LFE input signal.

The default values:

- $G_{lfe} = 10^{5.5/20} \approx 1.88364911$
- The $binraural\_latency\_s$ includes the HRTF/BRIR group delay and the processing delay. The default values are:
  - For Fastconv binaural renderer:
    - HOA to binaural filters, $binraural_latency_s = 0.000020834$s
    - MC to binaural filters (HRIR), $binraural_{latency_s} = 0.000020834s$
    - MC to binaural filters (BRIR), $binraural\_latency_s = 0.000937500s$
  - For TD binaural binaural renderer:
    - HRIR, $binraural\_latency_s = 0.000020834$
  - For Crend binaural renderer:
    - HOA to binaural filters, $binraural\_latency_s = 0.000020833s$
    - MC to binaural filters (HRIR), $binraural\_latency_s = 0.000666667s$
    - MC to binaural filters (BRIR), $binraural\_latency_s = 0.000145834s$

During conversion HRTF/BRIR conversion process the binaural filter shall be estimated.

7.4.7.2 Parametrization of Binaural renderers using binary file

Head related filters for the binaural rendering may be provided to the decoder or the renderer by using dynamic loading of external binary file. The way to generate such a binary file from a set of SOFA file is described in [9].

The decoder program should be called with option -hrtf <binary_file>. This option can be used with the output configurations BINAURAL, BINAURAL_ROOM_IR and BINAURAL_ROOM_REVERB.

A binary file has a specific container format with a header and a sequence of entries.

The header of a binary file is defined according to table 7.4-2 as follows:

Table 7.4-2: Binary file header

Offset	Format	Length (in bytes)	Description
0	string	8	File identifier: “IVASHRTF”
8	integer	4	Size of file in bytes (header of file included)
12	integer	2	Number of entries (HR filters)
14	integer	4	Max size of raw data (HR filter in binary format)

Every entry contains a header followed by the related raw data which is the binary representation of the HR filter.

The header of each entry is defined as given in table 7.4-3:

Table 7.4-3: Entry headers

Offset	Format	Length (in bytes)	Description
0	integer	4	Renderer type The renderer type is defined according to the enumeration RENDERER_TYPE among the following values : - HRTF_READER_RENDERER_BINAURAL_FASTCONV - HRTF_READER_RENDERER_BINAURAL_FASTCONV_ROOM - HRTF_READER_RENDERER_BINAURAL_PARAMETRIC - HRTF_READER_RENDERER_BINAURAL_PARAMETRIC_ROOM - HRTF_READER_RENDERER_BINAURAL_OBJECTS_TD - HRTF_READER_RENDERER_BINAURAL_MIXER_CONV - HRTF_READER_RENDERER_BINAURAL_MIXER_CONV_ROOM - HRTF_READER_RENDERER_REVERB_ALL
4	integer	4	Input audio configuration The input audio configuration is defined according to the enumeration BINAURAL_INPUT_AUDIO_CONFIG among the following values : - BINAURAL_INPUT_AUDIO_CONFIG_COMBINED - BINAURAL_INPUT_AUDIO_CONFIG_HOA3 - BINAURAL_INPUT_AUDIO_CONFIG_HOA2 - BINAURAL_INPUT_AUDIO_CONFIG_FOA - BINAURAL_INPUT_AUDIO_CONFIG_UNDEFINED
8	integer	4	Sampling frequency (16000, 32000, 48000)
12	integer	4	Raw data size in bytes

The format of the raw data depends on the rendering and the HR filters are represented in floating point.

Note:
- Both renderer types RENDERER_BINAURAL_PARAMETRIC and RENDERER_BINAURAL_FASTCONV_ROOM always have only one entry regardless of input audio configuration and sampling frequency.
- The HR filters for the renderer types RENDERER_BINAURAL_FASTCONV and RENDERER_BINAURAL_FASTCONV_ROOM are fully defined at 48kHz.

- For the renderer type `RENDERER_BINAURAL_OBJECTS_TD` the input audio configuration is always `BINAURAL_INPUT_AUDIO_CONFIG_UNDEFINED`.
- renderer type `HRTF_READER_RENDERER_REVERB_ALL` should be associated with `HRTF_READER_RENDERER_BINAURAL_OBJECTS_TD` and/or `HRTF_READER_RENDERER_BINAURAL_CREND` to specify the binaural reverberation parameters jointly with new HRIR parameters. They shall be computed on the same HRIR set.
- The binary file does not have to contain all data for all renderers. The following minimal configurations are accepted or any combination of those

7.4.7.3 HRTFs and BRIR conversion methods

7.4.7.3.1 Conversion from spatial domain to spherical harmonics domain

The conversion from a spatial-domain head-related impulse response (SD HRIR) set to a spherical-harmonics-domain head-related impose response (SHD HRIR) set consists of 4 main steps:

1. Determine the bulk delays of the HRIRs in the SD HRIR set and compute a delay-less HRIR set from the SD HRIR set by removing the bulk delay computed previously.
2. Determine a set of all-pass filters that are dependent on the previously calculated bulk-delays.
3. Apply the all-pass filters to the delay-less HRTFs.
4. Form the SHD HRIR set by linearly combining the now delay-less sampled non-sparse HRTF set.

These main steps can be further elaborated into the following intermediate steps:

Step 1 – Determine bulk delay and compute delay-less HRIRs

Interpolate the existing HRIRs onto a dense sphere of known positions.

$$H_{interp}(k) = interp\big(H_{SD}(n)\big), \tag{7.4-32}$$

where $H_{interp}(k), k = 1..K$ are the K dense positions interpolated HRIRs and $H_{SD}(n), n = 1..N$ are the N original HRIRs.

Compute the frequency domain representation of the interpolated HRIRs using an MDFT.

$$F_k = MDFT(H_{interp}(k)), \tag{7.4-33}$$

where F_k are the K complex frequency responses of the HRIRs.

Find the equivalent minimum-phase versions of the frequency responses, using the Hilbert transform.

$$F_{k,minphase} = mag2minphase(F_k) \tag{7.4-34}$$

Calculate the excess phase by subtracting the minimum-phase frequency responses from the original frequency responses.

$$ExcessPhase_k = angle(F_k) - angle(F_{k,minphase}), \tag{7.4-35}$$

where $ExcessPhase_k$ is a matrix of $K \times nBins$ dimensions.

Find the interaural time delay (ITD) by looking at the excess phase at 1.2 kHz.

$$ITD = ExcessPhase(1..K, i) \tag{7.4-36}$$

Where ITD is the $K \times 1$ vector of delays and i is the index of the 1.2 kHz bin.

Find the maximum delay from all of the ITDs.

$$MaxDel = \max(ITD) \tag{7.4-37}$$

Create delays for the left ear using the ITDs and max delay to ensure causality.

$$EarDel_{left} = (0.5 \times \text{ITD}) + (0.5 \times \text{MaxDel}) \times \left(1 - \frac{2}{\pi} \times \cos\left(\frac{\text{ITD} \times \frac{\pi}{2}}{MaxDel}\right)\right) \quad (7.4\text{-}38)$$

Step 2 – Determine a set of all-pass filters

Create the desired phase response by using the delays calculated in the previous step and pre-defined base polynomials.

$$X = polyval(protoPoly, EarDel_{left}) \quad (7.4\text{-}39)$$

Where X is a matrix of dimensions $6 \times K$ and $protoPoly$ is the stored polynomial coefficients.

Then X is mapped to 6 complex filter poles P_s in the Laplace domain (s-domain)

$$ang_{j,k} = \frac{\text{atan}(X_{j-1,k})}{2} + \frac{\pi}{4} \quad (7.4\text{-}40)$$

$$P_{Sj,k} = -2\pi \times X_{j.k} \times e^{([i, \ -i] \times ang_{j.k})} \quad (7.4\text{-}41)$$

where $j = [5, .. 1]$ indexing the output of the polynomials and for $j = 0$ $P_{S0,k} = -2\pi \times X_{0,k}$. This is performed for all K positions.

Then take the bilinear transform of the poles to convert into z-domain poles P_z.

$$P_Z = bilinear(P_S) \quad (7.4\text{-}42)$$

Compute the first 512 coefficients of the filters' impulse responses:

$$AP_{IR} = filter(\delta(x), \ P_Z) \quad (7.4\text{-}43)$$

Where AP_{IR} is the $K \times 512$ vector of impulse response all-pass filters, $\delta(x)$ is a unit-impulse and P_Z is the z-domain poles.

Step 3 – Apply the all-pass filters

Compute the average magnitude response using the mean of the L ear and R ear frequency responses

$$M_{ave} = mean(abs(F_{k,minphase})) \quad (7.4\text{-}44)$$

Where M_{ave} is the average magnitude response.

Convolve the all-pass filters with the averaged magnitude response.

$$F_{phasecomp} = M_{ave} \times AP \quad (7.4\text{-}45)$$

Find the equivalent minimum-phase version of the convolved FRs.

$$F_{mpc} = mag2minphase(F_{phasecomp}) \quad (7.4\text{-}46)$$

Step 4 – Form the SHD HRIR set

Weigh the frequency responses computed in Step 3 to ensure stability when solving for the final frequency responses.

$$W = \frac{1}{\sqrt[4]{max(abs(F_{mpc}), 0.1)}} \quad (7.4\text{-}47)$$

Where W is a weights matrix of $K \times 512$ size.

Compute the pseudo-inverse matrix G^+ of the basis function panning gains G that map the dense-sampled sphere to SHD, and solve for the SHD FRs. The panning gains G can either be computed on-the-fly or be stored as a set of coefficients, as they depend on the SHD format but not on the SD HRIRs.

First F_{mpc} and G are multiplied by the previously calculated weights.

$$\hat{F}_{mpc} = F_{mpc} \times W \quad (7.4\text{-}48)$$

$$\hat{G} = G \times W \quad (7.4\text{-}49)$$

The relationship between the SD HRTF filter responses F_{mpc} and the SHD HRTF filter responses B is as follows:

$$\hat{F}_{mpc} = \hat{G} \times B \quad (7.4\text{-}50)$$

Obtain B using the pseudoinverse $\hat{G}^{+}$:

$$B = \hat{G}^{+} \times \hat{F}_{mpc} \quad (7.4\text{-}51)$$

The SHD frequency responses are then converted back to impulse responses by taking the inverse MDFT.

$$H_{SHD} = IMDFT(B) \quad (7.4\text{-}52)$$

Create the symmetric right-ear HRIRs by multiplying the left-ear HRIRs by either [-1, 1] depending on the SHD channel.

Finally, ensure the HRIRs have the correct duration by multiplying by a sinusoidal window to create the desired IR length.

$$H_{final} = H_{SHD}(1{:}\,l) \times sin(win) \quad (7.4\text{-}53)$$

7.4.7.3.2 Conversion from Time domain to CLDFB domain for HRIRs (Fast convolution binaural renderer)

Binaural filtering using HRIR (FIR) filters can also be carried out in the CLDFB domain, which is beneficial if the audio is present in the CLDFB domain and if further processing in that domain is required. Therefore, a method is presented here which optimally converts the HRIR to complex-valued CLDFB domain multi-tap FIR filters. The optimization minimizes the squared error between the CLDFB domain processed signal and the time domain HRIR filtering. The optimal FIR coefficients are computed such that the impulse response of the system comprising CLDFB analysis, FIR filtering and CLDFB synthesis best matches the target HRIR, jointly optimized for all relevant multiple impulse positions p. The joint optimization for multiple impulse positions is needed due to the periodically time varying nature of the CLDFB where the period corresponds to the stride S (=60). The method works as follows, where N complex coefficients (or taps, typically N = 3 or N = 4) are computed for each of the L (=60) CLDFB frequency bands:

1. Concatenate S target impulse responses $h_{HRIR}(n-p)$ according to S unit-pulse inputs, each unit-pulse shifted by 1 sample and delayed by the overall CLDFB delay, to a column vector *t* of length S x *M* where *M* is set to the CLDFB domain impulse response length with FIR filtering.
2. For all unit-pulse $\delta(n-p)$inputs do
    a. CLDFB analysis
    b. For all (N x L) real-valued FIR coefficients
        i. Compute elementary signals $u_{p,l,k}(n)$ as contributions of a single, real-valued, filter tap k in frequency band l for pulse position p by setting the investigated filter tap to 1.0 and all other filter taps to zero, filtering and CLDFB synthesis.
        ii. Store the elementary signals of length M in the respective column of the matrix *C* starting at the position corresponding to the start position in the target vector.
    c. For all (N x L) imaginary valued FIR coefficients
        i. Compute elementary signals $v_{p,l,k}(n)$ as contribution of the single imaginary-valued filter tap k in frequency band l for pulse position p by setting the investigated filter tap to the imaginary number *j* and all other taps to zero, filtering and CLDFB synthesis.
        ii. Store the elementary signals of length M in the respective column of the matrix *C* starting at the position corresponding to the start position in the target vector.

3. Solve the matrix equation $Ch \approx t$ for column vector *h* in the least squares error sense where *h* holds the optimized N x 2 x L FIR coefficients and matrix *C* has dimensions [S x M, N x 2 x L].

   a. The least squares error solution may be computed as $h = (C^T C)^{-1}(Ct)$, where $(C^T C)$ is conditioned so that the matrix inversion remains numerically stable.

   b. The least squares error solution provides filter coefficients such that for all p = 0,1,…,S-1 $\sum_{l=0}^{L-1}\sum_{k=0}^{N-1}\left(h_{r,l}(k)u_{p,l,k}(n) + h_{i,l}(k)v_{p,l,k}(n)\right) \approx h_{HRIR}(n-p)$, where $h_{i,l}(k)$ corresponds to the imaginary coefficient for tap k and$h_{r,l}(k)$ to the real coefficient respectively.

   c. Rearrange the real and imaginary coefficients in *h* to an array which is suited to perform the filtering in the CLDFB domain.

The FIR filtering in the CLDFB domain on signal $X_l(k)$ is performed by running each FIR filter corresponding to one frequency band *l* for a particular time slot *k* as

$$V_l(k) = \sum_{m=0}^{N-1}(h_{r,l}(m) + jh_{i,l}(m))\,X_l(k-m) \qquad (7.4\text{-}54)$$

Figure 7.4-5 illustrates how the contribution from a single filter tap is computed.

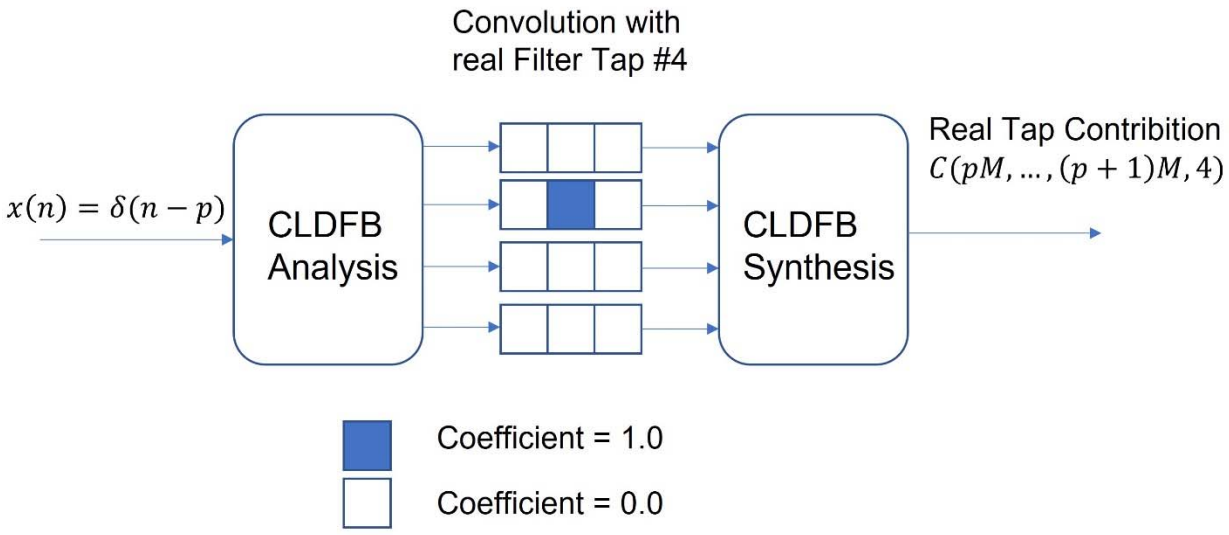


Figure 7.4-5: Computation of a single (real) tap contribution where *M* is the impulse response length in samples for the example of 3 taps FIR CLDFB domain filters.

7.4.7.3.3 Conversion from Time domain to CLDFB domain for BRIRs (Fast convolution binaural renderer)

For the conversion of BRIRs to CLDFB domain for the FastConv renderer, the procedure is described in clause 7.2.2.4.2.2.

7.4.7.3.4 Conversion from Time domain to HRIR/ITD model (Time Domain binaural renderer)

7.4.7.3.4.1 Modelling of head-related filters

Head-related (HR) impulse responses (HRIR) used for binauralization in the Time Domain binaural renderer are efficiently represented by HRIR and ITD model parameters (see clause 7.2.2.2). To obtain model parameters, the error between an HR filter model and the HR filter data set it minimized, given a selected model structure. The HR filter set may or may not be modelled using a separate ITD model, meaning the time difference between the left and right HR filters are extracted and modelled separately or included in a single HR model over azimuth and elevation angles. For the IVAS default HR filter set, the minimum phase (zero-time-delay) filters and the ITD are modelled separately.

At first a model structure has to be selected. As described in clause 7.2.2.2, the HR filters are represented by B-spline basis functions over azimuth and elevation, where azimuth basis functions are periodic (circular) and elevation basis

functions are standard B-spline functions for the minimum phase filters. For ITD, standard B-spline basis functions are used both for elevation and azimuth.

Fewer azimuth basis function may be used towards the poles than at the equator of the sampled HR filter sphere given the angular distance between HR filter sample points is getting smaller towards the poles. For this reason, different knot sequences may be used for different elevations. For the IVAS default HR filter set, a single elevation knot sequence is defined with 15-degree knot interval as

$$Kseq_{hrf}^{elev} = \{-90, -75, \dots, 75, 90\} \quad (7.4\text{-}55)$$

For the poles at $\pm 90\ deg$ there is a single azimuth knot interval as there is no variation over azimuth, i.e. $Kseq_{hrf}^{azim}(0) = Kseq_{hrf}^{azim}(L_K^{elev} - 1) = \{0, 360\}$, where L_K^{elev} is the number of elevation knot points. For the other elevation knot points, the following azimuth knot sequence is used

$$Kseq_{hrf}^{azim}(p) = \{0, 10, \dots, 350, 360\} \quad (7.4\text{-}56)$$

for the elevation knot point index p. This means for the default IVAS HR filter set, the azimuth knot point sequences are the same for all elevations except for the poles at $\pm 90\ deg$.

For the default IVAS HR filter ITD model,

$$Kseq_{itd}^{elev} = \{-90, -78, \dots, 78, 90\},\ Kseq_{itd}^{azim} = \{0, 10, \dots, 180\}. \quad (7.4\text{-}57)$$

To extract the minimum phase HR filters, onset delays $d_{l,r}^{onset}(\vartheta_m, \varphi_m)$ for the left and right filter are estimated. For the IVAS default filter set, this can be done finding the first non-zero sample of the impulse responses for the left and the right filter respectively. Additionally, the latency imposed by the HR filtering is estimated. For the default IVAS filter set, this is estimated finding the index $i_{l,r}^{max}$ where the impulse responses reach their maxima for the left and right ear impulse responses respectively. The HR filter latency is then determined as

$$d_{hrf} = \frac{\min(i_l^{max}, i_r^{max})}{fs} \quad (7.4\text{-}58)$$

where fs is the sample rate of the HR filter impulse responses.

Using a separate ITD model, as is the case for the default IVAS HR filter set, the minimum phase HR filter set to be modelled is extracted from the original HR filter set based on the determined onset delays $d_{l,r}^{onset}(\vartheta, \varphi)$. The extracted set covers the filter tap from the filter onset to the end of each impulse response. Depending on the length of the original HR filter set impulse responses, the latter part of the filters might not contribute significantly to the spatialization. In that case, the extracted HR filter set may comprise truncated HR filters from the original HR filter set, e.g., covering 2 ms. This way the complexity of the rendering can be controlled. A maximum filter length of 256 samples is supported. Longer filters are truncated prior to rendering.

The extracted HR filter set is not modelled directly, but a delay correction procedure is performed to handle potential inconsistencies in the original HR filter set, see clause 7.4.7.3.4.2.

Prior to determining a model of the minimum phase HR filters and the ITD, an initial model of the minimum phase HR filters is determined and used to correct potential inconsistencies in the input HR filter set. Subsequently, model parameters for the minimum phase HR filters are determined according to clause 7.4.7.3.4.4. The ITD model is determined similarly as the minimum phase HR filter model, see clause 7.4.7.3.4.5. However, as can be seen in clause 7.2.2.2, different sets of B-spline functions are used.

7.4.7.3.4.2 Delay correction of HR filter set

The delay correction aims to mitigate inconsistencies of onset delay for the HR filters of an HR filter set. Such inconsistencies in the data, may e.g. be caused by errors introduced in HR filter measurement or by post processing applied to the HR filter set. The delay correction is run on the minimum phase HR filter set extracted from the original HR filter set to model utilizing a B-spline HR filter model. To avoid overfitting for the final HR filter model, a subset of the original data is used for the modelling, leaving some data for validation. However, for the delay correction, it may be beneficial to use the complete original HR filter set generating a model for delay correction, as is done for the default IVAS HR filter set.

To generate a model, the B-spline basis functions need to be computed. For the 4th order standard B-spline model (with cubic spline functions), a multiplicity of 4 is applied to the first and last knot point in $Kseq$. For example, the elevation knot sequence, which including multiplicities becomes

$$Kseq_{hrf}^{elev} := \{-90, -90, -90, -90 - 75, \dots, 75, 90, 90, 90, 90\} \quad (7.4\text{-}59)$$

Based on the knot point sequence, the cubic polynomials can be derived. Similarly for azimuth, the cubic polynomials can be determined from the knot point sequence. However, as periodic B-spline basis functions are used, there is no knot multiplicity, but the knot point sequence is extended beyond the modelling intervals to get the right number of basis functions. For the IVAS default model, it becomes

$$Kseq_{hrf}^{azim}(p) := \{-30, -20, -10, 0, 10, \dots, 350, 360, 370, 380\} \ \forall p \in \{1, \dots, L_K^{elev} - 2\} \quad (7.4\text{-}60)$$

from which the periodic B-spline basis functions can be derived.

With the extracted HR filter set and the obtained B-spline basis functions, an initial model is obtained as given by equation (7.4-71) of clause 7.4.7.3.4.3. Based on this initial model, delay correction for the filters of the original HR filter set is iteratively performed.

For each sample of the HR filter set $\boldsymbol{h}(\vartheta_m, \varphi_m)$, a difference between a time shifted original HR filter and the corresponding HR filter of the initial HR filter model $\widehat{\boldsymbol{h}}\,(\vartheta_m, \varphi_m)$ is computed. Then the time shift giving the minimum difference is determined as

$$\hat{s}_m(\vartheta_m, \varphi_m) = \arg\max \left(\frac{\sum_k \left(\text{shift}(h_k(\vartheta_m, \varphi_m), s_i) - \widehat{h}_k(\vartheta_m, \varphi_m) \right)^2}{\sum_k \left(\text{shift}(h_k(\vartheta_m, \varphi_m), s_i) \right)^2} \right) \quad (7.4\text{-}61)$$

for shifts of $s_i = \{-maxDel, -maxDel + 1, \dots, 0, \dots, maxDel - 1, maxDel\}$, where $\text{shift}(h_k(\vartheta_m, \varphi_m), s_i)$ is shifting $h_k(\vartheta_m, \varphi_m)$ by s_i samples. Depending on the HR filter set and the sample frequency, shifts by fractions of samples may be used. For the default IVAS HR filter set, $maxDel = 5$.

If $\hat{s}_m(\vartheta_m, \varphi_m) \neq 0$, the original filter is updated to the corrected filter $h_k(\vartheta_m, \varphi_m) := \text{shift}\left(h_k(\vartheta_m, \varphi_m), \hat{s}_m(\vartheta_m, \varphi_m)\right)$ and a new model is determined for the corrected HR filter set $h_k(\vartheta_m, \varphi_m)$. The iterative correction may be done until there are no corrected filters in an iteration anymore or until N_{iter} iterations are reached. For the IVAS default HR filter model, $N_{iter} = 20$ is used. Once the delay correction is done, the onset delays $d_{l,r}^{onset}(\vartheta_m, \varphi_m)$ are updated corresponding to the time shifts $\hat{s}_m(\vartheta_m, \varphi_m)$ obtained from the delay correction.

The ITD to be modelled can then be determined as

$$ITD(\vartheta, \varphi) = \tilde{d}_l^{onset}(\vartheta_m, \varphi_m) - \tilde{d}_r^{onset}(\vartheta_m, \varphi_m) \quad (7.4\text{-}62)$$

using the updated $\tilde{d}_{l,r}^{onset}(\vartheta_m, \varphi_m)$ for the sampled elevation angle ϑ_m and an azimuth angle φ_m.

7.4.7.3.4.3 Determination of model parameters for delay correction

To obtain model parameters, a least squares minimization is performed. The following error criterion is set up for the modelling parameters $\boldsymbol{\alpha}$

$$J(\boldsymbol{\alpha}) = \sum_{m=1}^{M} \left\| \boldsymbol{h}(\vartheta_m, \varphi_m) - \widehat{\boldsymbol{h}}\,(\vartheta_m, \varphi_m) \right\|^2 \quad (7.4\text{-}63)$$

where

$$\boldsymbol{h}(\vartheta_m, \varphi_m) = \begin{pmatrix} h_1(\vartheta_m, \varphi_m) \\ \vdots \\ h_K(\vartheta_m, \varphi_m) \end{pmatrix} \text{ and} \quad (7.4\text{-}64)$$

$$\widehat{\boldsymbol{h}}(\vartheta_m, \phi_m) = \begin{pmatrix} \hat{h}_1(\vartheta_m, \varphi_m) \\ \vdots \\ \hat{h}_K(\vartheta_m, \varphi_m) \end{pmatrix} = \sum_{p=1}^{P} \sum_{q=1}^{Q_p} \sum_{k=1}^{K} \alpha_{p,q,k} \Theta_p(\vartheta_m) \Phi_{p,q}(\varphi_m) \boldsymbol{e}_k. \quad (7.4\text{-}65)$$

The number of parameters estimated is $\left(\sum_{p=1}^{P} Q_p\right) \cdot K$, which should be much fewer than the number of available data samples $M \cdot N$ to avoid an underdetermined system.

As the canonical orthonormal basis vectors $\boldsymbol{e}_k$ are orthonormal vectors, the parameters $\boldsymbol{\alpha}_k = \{\alpha_{p,q,k} : p = 1, \dots, P;\ q = 1, \dots, Q_p\}$ can be solved independently for each sample k. For each sample k the linear minimization criterion becomes

$$J(\boldsymbol{\alpha_k}) = \sum_{m=1}^{M} \left| h_k(\vartheta_m, \varphi_m) - \sum_{p=1}^{P} \sum_{q=1}^{Q_p} \alpha_{p,q,k} \Theta_p(\vartheta_m) \Phi_{p,q}(\varphi_m) \right|^2. \quad (7.4\text{-}66)$$

In matrix form, it can be expressed as

$$J(\boldsymbol{\alpha_k}) = \left\| \underbrace{\begin{pmatrix} h_k(\vartheta_1, \varphi_1) \\ \vdots \\ h_k(\vartheta_M, \varphi_M) \end{pmatrix}}_{\boldsymbol{h_k}} - \underbrace{\begin{pmatrix} \boldsymbol{b}(\vartheta_1, \varphi_1) \\ \vdots \\ \boldsymbol{b}(\vartheta_M, \varphi_M) \end{pmatrix}}_{\boldsymbol{B}} \boldsymbol{\alpha}_k \right\|^2 = \|\boldsymbol{h}_k - \boldsymbol{B}\boldsymbol{\alpha}_k\|^2, \quad (7.4\text{-}67)$$

where

$$\begin{aligned} \boldsymbol{b}(\vartheta_m, \varphi_m) &= \left(\Theta_p(\vartheta_m)\Phi_{p,q}(\varphi_m)\colon\ p = 1,\cdots, P;\ q = 1,\cdots, Q_p\right)_{row\ vector} \\ \boldsymbol{\alpha}_k &= \left(\alpha_{p,q,k} : p = 1,\cdots, P;\ q = 1,\cdots, Q_p\right)_{column\ vector} \end{aligned}. \quad (7.4\text{-}68)$$

The solution that minimizes $J(\boldsymbol{\alpha}_k)$ is obtained by solving the normal equation $\boldsymbol{\alpha}_k = (\boldsymbol{B}^T\boldsymbol{B})^{-1}\boldsymbol{B}^T\boldsymbol{h}_k$. However, as the solution may be sensitive to noise in the data $\boldsymbol{h}_k$, Tikhonov regularization is applied as follows

$$\bar{\bar{J}}(\boldsymbol{\alpha}_k) = \|\boldsymbol{h}_k - \boldsymbol{B}\boldsymbol{\alpha}_k\|^2 + \lambda^2\|\boldsymbol{\alpha}_k\|^2 = \left\| \underbrace{\begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{0} \end{bmatrix}}_{\bar{\bar{\boldsymbol{h}}}_k} - \underbrace{\begin{bmatrix} \boldsymbol{B} \\ \lambda \boldsymbol{I} \end{bmatrix}}_{\bar{\bar{\boldsymbol{B}}}} \boldsymbol{\alpha}_k \right\|^2 = \left\|\bar{\bar{\boldsymbol{h}}}_k - \bar{\bar{\boldsymbol{B}}}\boldsymbol{\alpha}_k\right\|^2. \quad (7.4\text{-}69)$$

where $\boldsymbol{I}$ is the identity matrix of size $\sum_{p=1}^{P} Q_p$ and $\boldsymbol{0}$ is a column vector of zeros with $\sum_{p=1}^{P} Q_p$ elements and λ is the regularization factor.

The solution that minimizes $\bar{\bar{J}}(\boldsymbol{\alpha}_k)$ is similarly obtained by solving the normal equation $\boldsymbol{\alpha}_k = \left(\bar{\bar{\boldsymbol{B}}}^T\bar{\bar{\boldsymbol{B}}}\right)^{-1}\bar{\bar{\boldsymbol{B}}}^T\bar{\bar{\boldsymbol{h}}}_k$. The value of λ may be determined such that the condition number of the matrix $\bar{\bar{\boldsymbol{B}}}$ is less than 10 or some other value that leads to good model accuracy. For the default IVAS HR filter set, a regularization factor of $\lambda = 0.5$ is used.

For better numerical accuracy, $\boldsymbol{\alpha}_k$ is solved by using singular value decomposition (SVD) of $\bar{\bar{\boldsymbol{B}}}$,

$$\bar{\bar{\boldsymbol{B}}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \quad (7.4\text{-}70)$$

The columns of $\bar{\bar{\boldsymbol{B}}}$ and $\boldsymbol{U}$ span the same subspace. The projection of $\bar{\bar{\boldsymbol{h}}}_k$ onto the orthonormal matrix $\boldsymbol{U}$ is given by $\boldsymbol{U}\boldsymbol{U}^T\bar{\bar{\boldsymbol{h}}}_k$ and equals $\bar{\bar{\boldsymbol{B}}}\boldsymbol{\alpha}_k = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{\alpha}_k$. This gives $\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{\alpha}_k = \boldsymbol{U}^T\bar{\bar{\boldsymbol{h}}}_k$, which then gives the solution for the model parameters

$$\boldsymbol{\alpha}_k = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^T\bar{\bar{\boldsymbol{h}}}_k. \quad (7.4\text{-}71)$$

This estimation becomes very efficient as it only requires one SVD of $\bar{\bar{\boldsymbol{B}}}$ that is a matrix of relatively small dimension, which can be done in parallel for $k = 1, \ldots, K$.

7.4.7.3.4.4 Determination of model parameters for minimum phase HR filters

Using the delay corrected HR filter set, the data is split into a training and test set. The split is based on spatial areas determined by the density of sample points within an area encompassing each sample point in the elevation-azimuth plane, corresponding to an expansion of the sphere of HR filter samples into a flat surface. Figure 7.4-6 illustrates the spatial areas (grey boxes) for the default IVAS HR filter set. The areas are larger for sample point with a lower density of neighbouring sample point and smaller for a higher density of sample points. The rectangular areas are determined by calculating the distance in elevation between sample points, splitting the distance in half between adjacent elevation levels. Then, the other dimension of the rectangle is similarly given by calculating the distance between neighbouring sample points in azimuth and splitting the distance in half. A special case is at the poles $\pm 90\ deg$ where there is no neighbouring sample. In this case the spatial area is covering the complete range of 360 degrees.

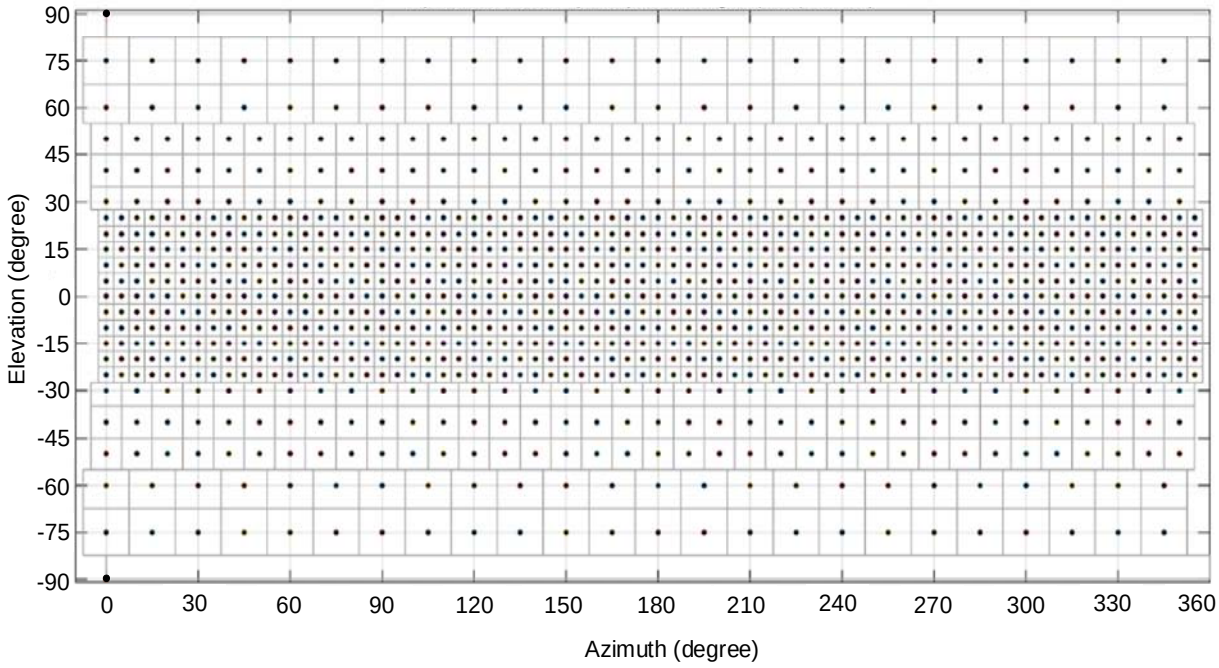


Figure 7.4-6: Spatial areas for sample points of the default IVAS HR filter set.

The size of each spatial area is computed, and a cumulative count of the spatial areas is determined as shown in figure 7.4-7. The training set is selected as all sample points with a spatial area larger than the size of the spatial are corresponding to half of the total spatial area counts, and 50 % of the samples, evenly distributed over azimuth, for spatial areas smaller or equal to this size of a spatial area. For the default IVAS HR filter set, as depicted in 7.4-7, this means 50% of the sample points with the smallest spatial area and 100% of the other sample points are used for the modelling.

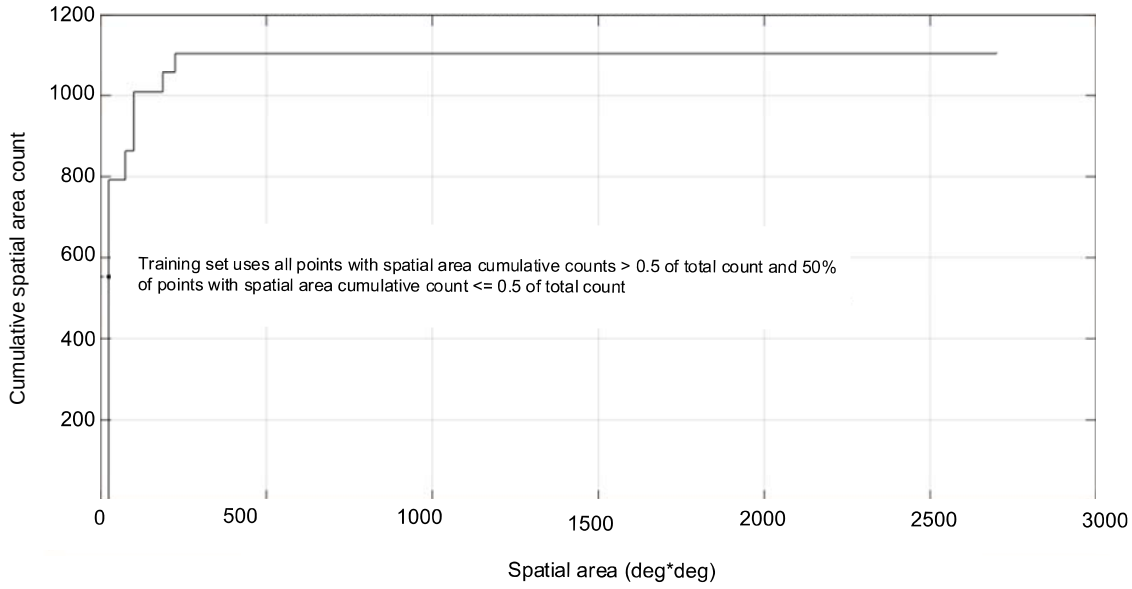


Figure 7.4-7: Cumulative spatial area count for the default IVAS HR filter set.

Further, the spatial areas are used to weight sample points corresponding to a larger spatial area higher than sample point corresponding to a smaller spatial area for the modelling. This improves the modelling for areas closer to the poles.

The HR filter model for the minimum phase HR filter set is determined by minimizing

$$J_w(\boldsymbol{\alpha}) = \sum_{m=1}^{N_T} w_m \left\| \boldsymbol{h}(\vartheta_m, \varphi_m) - \widehat{\boldsymbol{h}}\,(\vartheta_m, \varphi_m) \right\|^2 \quad (7.4\text{-}72)$$

where $J_w(\boldsymbol{\alpha})$ is the weighted modelling error, N_T is the number of HR filters included in the training set, and w_m is weight value for an *m*-th HR filter in the training set. The weights are determined based on the spatial areas as

$$\begin{pmatrix} w_1 \\ \vdots \\ w_{N_T} \end{pmatrix} = \begin{pmatrix} \sqrt{N_T \frac{a_1}{a_T}} \\ \vdots \\ \sqrt{N_T \frac{a_{N_T}}{a_T}} \end{pmatrix} \tag{7.4-73}$$

where $\left(a_1, \ldots a_{N_T}\right)$ are the sizes of the spatial areas corresponding to the sample points $m = \{1, \ldots, N_T\}$. Figure 7.4-8 illustrates the relative weights for sample points based on the size of the spatial areas.

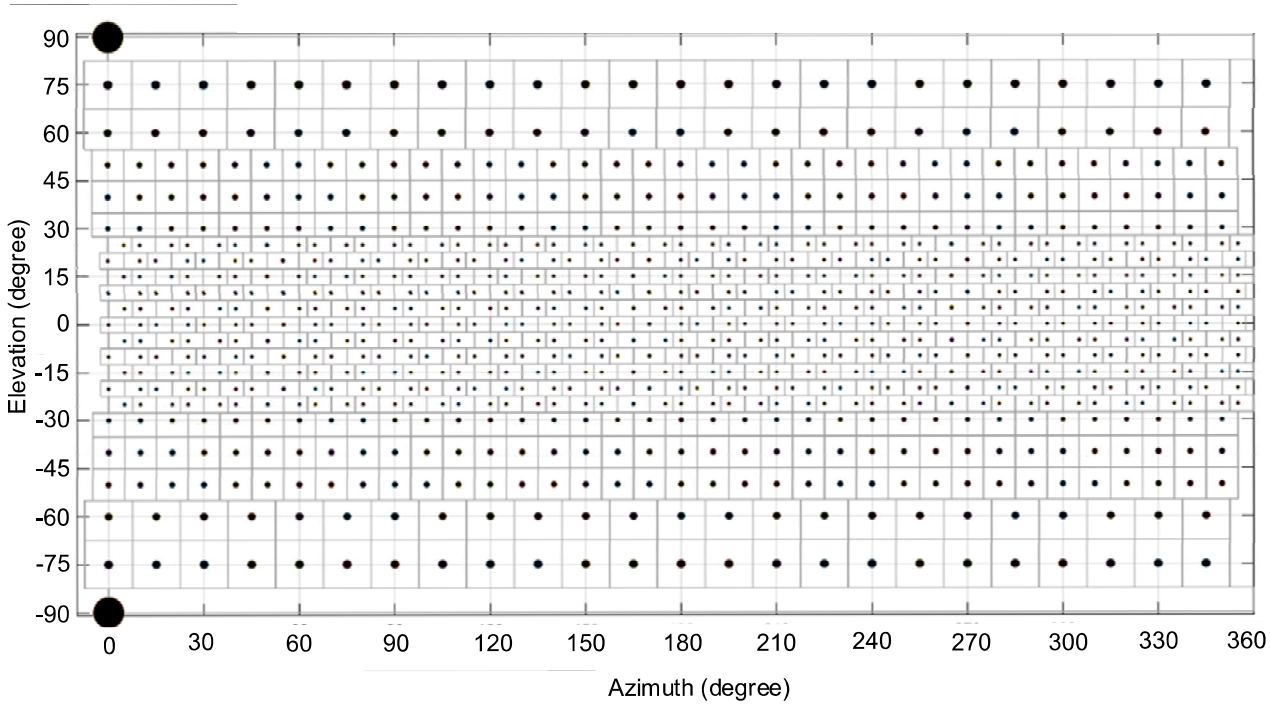


Figure 7.4-8: Weights (proportional to marker diameters) for HR filter samples of the default IVAS filter set.

In accordance with clause 7.4.7.3.4.3, including Tikhonov regularization, the error criterion can be expressed in matrix form as

$$\bar{\bar{J}}(\boldsymbol{\alpha}_k) = \|\boldsymbol{h}_k - \boldsymbol{W}\boldsymbol{B}\boldsymbol{\alpha}_k\|^2 + \lambda^2\|\boldsymbol{\alpha}_k\|^2 = \left\| \underbrace{\begin{bmatrix} \boldsymbol{W}\boldsymbol{h}_k \\ \boldsymbol{0} \end{bmatrix}}_{\bar{\bar{\boldsymbol{h}}}_k} - \underbrace{\begin{bmatrix} \boldsymbol{W}\boldsymbol{B} \\ \lambda \boldsymbol{I} \end{bmatrix}}_{\bar{\bar{\boldsymbol{B}}}} \boldsymbol{\alpha}_k \right\|^2 = \left\| \bar{\bar{\boldsymbol{h}}}_k - \bar{\bar{\boldsymbol{B}}}\boldsymbol{\alpha}_k \right\|^2. \tag{7.4-74}$$

where $\boldsymbol{I}$ is the identity matrix of size $\sum_{p=1}^{P} Q_p$ and $\boldsymbol{0}$ is a column vector of zeros with $\sum_{p=1}^{P} Q_p$ elements, $\boldsymbol{W}$ is a square diagonal matrix of weight values w_m on the diagonal, and λ is the regularization factor. For the default IVAS HR filter set, a regularization factor of $\lambda = 0.5$ is used.

The solution minimizing $\bar{\bar{J}}(\boldsymbol{\alpha}_k)$ is obtained by solving the normal equation $\boldsymbol{\alpha}_k = \left(\bar{\bar{\boldsymbol{B}}}^T\bar{\bar{\boldsymbol{B}}}\right)^{-1}\bar{\bar{\boldsymbol{B}}}^T\bar{\bar{\boldsymbol{h}}}_k$, using singular value decomposition (SVD) of $\bar{\bar{\boldsymbol{B}}}$ in accordance with clause 7.4.7.3.4.3, which gives

$$\boldsymbol{\alpha}_k = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^T\bar{\bar{\boldsymbol{h}}}_k. \tag{7.4-75}$$

7.4.7.3.4.5 Determination of model parameters for ITD

For the default IVAS HR filter set, the model for the ITD is determined in a similar manner as the minimum phase HR filters, see clause 7.4.7.3.4.4, with the same training set but not utilizing the weighting by size of spatial area. This means the sample points get the same weights as was the case for the model used for delay correction, see clause 7.4.7.3.4.3. A regularization factor of $\lambda = 0.2$ is used.

7.4.7.3.5 Conversion from HRTF/BRIR to Crend binaural renderer convolver parameters

The following description explaining how to compute Crend binaural renderer convolver parameters is informative. It describes how the default rom values are computed from default HRTF and BRIR set stored in SOFA files [21] (HRIR_128_Meth5_IRC_53_Q10_symL_Itrp1_48000.sofa [9], IIS_BRIR_officialMPEG_Combined.sofa [9]).

The energy of a single-channel signal $x(n)$ is defined by:

$$E(x) = \sum_{n=0}^{+\infty} x^2(n)$$

The forward cumulative energy is defined by :

$$e_{forward}(x,n) = \sum_{i=0}^{n} x^2(i)$$

The backward cumulative energy is defined by :

$$e_{backward}(x,n) = \sum_{i=n}^{+\infty} x^2(i)$$

Crend binaural renderer convolver parameters are the ones needed to implement equation (7.2-82) from clause 7.2.2.5.3.

For each $HRIR/BRIR_k^{l,r}$, we need to estimate:

- a starting time $To_k^{l,r}$,
- a mixing time $Tm_k^{l,r}$
- an ending time $Tend_k^{l,r}$

to be able to compute:

- $To = \min_{k,l,r}\left(To_k^{l,r}\right)$, the minimal value of the initial time for the HRIR/BRIR set.
- $Tm = \max_{k,l,r}\left(Tm_k^{l,r}\right)$, the maximal value of the mixing time for the HRIR/BRIR set.
- $Tend = \max_{k,l,r}\left(Tend_k^{l,r}\right)$, the maximal value of the ending time for the HRIR/BRIR set .

And deduce:

- m_{tm}, the number of direct blocks
- m_{end}, the number of diffuse blocks

We need also to compute the cut-off frequencies for each direct blocks $fc_k^{l,r}(m)$, the cut-off frequencies for each diffuse blocks and the weighting gain $G_k^{l,r} = \frac{G(k)}{W_k^{l,r}}$. $G(k)$ are a predeterminate gain (by default set to 1) allowing to adjust the level of reverberation for each input. It can be useful to adjust the level of late reverberation depending on the type of input (voice, musique, cinema, VR, AR, …). For example, if inputs are very correlated the level of late reverberation may be perceived louder compared to the original BRIR. Indeed, when convolving with original BRIR the late reverberation filters applied to each input are decorrelated. It is not the case with converted BRIR. In case of this case an attenuation of -3dB or -4dB can be applied for example to input channels known as correlated.

Parametrization of HRIR from default SOFA file

Starting from a set of HRIRs which maximum length is shorter than the size the DFT (MDFT) used (5ms) like the ones sored in HRIR_128_Meth5_IRC_53_Q10_symL_Itrp1_48000.sofa file the Crend parameters will be the following:

- $To = 0$s

- $Tm = Tend$ s
- $Tend = \frac{128}{4800} s$
- $m_{tm} = 1$
- $m_{end} = 0$
- $\boldsymbol{b}_k^{l,r}(0) = BRIR_k^{l,r}(0, Tend), k \in [1,16]$, 16 nearest directions found in SOFA file corresponding to standard loudspeaker directions for output format 5.1, 7.1, 5.1+2, 5.1+4, and 7.1+4
- $\boldsymbol{B}_k^{l,r}(0) = MDFT\left(\boldsymbol{b}_k^{l,r}(0)\right)$
- $fc_k^{l,r}(0) = Nyquist = F_s/2$, F_s being the sampling rate.

Equation (7.2-82) becomes:

$$O^{l,r} = \sum_{k=1}^{N} B_k^{l,r}(0) *_{[0;\ldots;F_S/2]} I_k$$

If input audio format is 5.1 for example N = 5. The low frequency effect (LFE) channel is added after the binaural downmix.

Parametrization of BRIR from default SOFA file

Starting from a set of BRIRs, first step is to find propagation time of each BRIRs.

Propagation time computation

Starting from a set of original BRIRs, the starting or propagation time $To_k^{l,r}$, is computed to remove audio samples with near-zero energy at the beginning of all BRIRs. For each channel $k \in [1, N = 16]$ and each ear $[l, r]$, starting time is estimated using the following steps:

- Compute BRIRs energies $\boldsymbol{E}_k^{l,r} = E(BRIR_k^{l,r}), k \in [1,16]$
- Compute max value of $\boldsymbol{E}_k^{l,r}$, $E_{max} = \max_{k,l,r}\left(\boldsymbol{E}_k^{l,r}\right)$
- Compute $To_k^{l,r} = find(t)\ such\ as\ e_{forward}\left(BRIR_k^{l,r}(t), t\right) = E_{max} * 10^{-BeginEnergyThreshold/10}\right)$
- Compute $To = \min_{k,l,r}\left(To_k^{l,r}\right)$

To compute BRIR default starting times values, $BeginEnergyThreshold = -50$.

Mixing time computation

Starting from a set of original BRIRs, the mixing time $Tm_k^{l,r}$, is computed to split the BRIRs into direct blocks and diffuse blocks. Direct blocks filters depend on inputs, while diffuse blocks filters are common to all inputs. For each channel $k \in [1, N = 16]$ and each ear $[l, r]$, mixing time is estimated using the following steps:

- Compute BRIRs energies $\boldsymbol{E}_k^{l,r} = E(BRIR_k^{l,r}), k \in [1,16]$
- Compute max value of $\boldsymbol{E}_k^{l,r}$, $E_{max} = \max_{k,l,r}\left(\boldsymbol{E}_k^{l,r}\right)$
- Compute $Tm_k^{l,r} = find(t)\ such\ as\ e_{backward}\left(BRIR_k^{l,r}(t), t\right) = E_{max} * 10^{-DirectEnergyThreshold/10}\right)$
- Compute $Tm = \min_{k,l,r}\left(Tm_k^{l,r}\right)$
- $m_{tm}{}_k^{l,r}$ is defined as the number of direct blocks for $BRIR_k^{l,r}$

- compute $m_{tm_k}^{l,r}$ such as $m_{tm_k}^{l,r} * L \leq (Tm_k^{l,r} - To) * F < m_{tm_k}^{l,r} + 1$
- compute $m_{tm} = \max_{k,l,r}(m_{tm_k}^{l,r})$ or or such as $m_{tm} * L \leq (Tm - To) * F < m_{tm} + 1$

To compute BRIR default values, $DirectEnergyThreshold = -25$. The resulted number of direct blocks is $m_{tm} =$ 22 (Default value).

Ending time computation

Starting from a set of original BRIRs, the ending time $Tend_k^{l,r}$, is estimated to compute the number of diffuse blocks. Diffuse filter blocks with very low energies are removed. For each channel $k \in [1, N = 16]$ and each ear $[l, r]$, ending time has been estimated using the following steps:

- Compute BRIRs energies $\boldsymbol{E}_k^{l,r} = e_{backward}(BRIR_k^{l,r}, Tm), k \in [1,16]$
- Compute max value of $\boldsymbol{E}_k^{l,r}$, $E_{max} = \max_{k,l,r}(\boldsymbol{E}_k^{l,r})$
- Compute $Tend_k^{l,r} = find(t)\ such\ as\ e_{backward}(BRIR_k^{l,r}(t), t) = E_{max} * 10^{-DiffuseEnergyThreshold/10})$
- Compute $Tend = \max_{k,l,r}(Tend_k^{l,r})$
- compute $m_{end_k}^{l,r}$ such as $m_{end_k}^{l,r} * L \leq (Tend_k^{l,r} - To) * F < m_{end_k}^{l,r} + 1$
- compute $m_{end} = \max_{k,l,r}(m_{end_k}^{l,r})$ or such as $m_{end} * L \leq (Tend - To) * F < m_{end} + 1$

To compute BRIR default value, $DiffuseEnergyThreshold = -25$. The resulted number of direct blocks is $m_{end} =$ 40 (Default value).

Frequency blocks computation

Direct frequency blocks are obtained by taking the MDFT of the BRIR corresponding blocks.

$$\boldsymbol{b}_k^{l,r}(m) = BRIR_k^{l,r}(To + m * L, To + (m+1).L - 1)$$

$$\boldsymbol{B}_k^{l,r}(m) = MDFT\left(\boldsymbol{b}_k^{l,r}(m)\right)$$

$$k \in [1, N], 0 \leq m < m_{tm}$$

Diffuse frequency blocks are obtained using equations (7.2-80) and (7.2-81). .

Cut-off frequencies computation

Default values for direct and diffuse blocks cut-off frequencies are extracted for direct and diffuse block filters.

For direct blocks:

- Compute direct block energy $\boldsymbol{E}_k^{l,r}(m) = e_{forward}(\boldsymbol{B}_k^{l,r}(m), Nyquits), k \in [1,16]$
- Compute
$$fc_k^{l,r}(m) = find(f)\ such\ as\ e_{backward}(\boldsymbol{B}_k^{l,r}(m), f) = \boldsymbol{E}_k^{l,r}(m) * 10^{-DirectFcThreshold/10})$$

To compute BRIR default value of $DiffuseFcThreshold = -20$

For diffuse blocks:

- Compute diffuse block energy $\boldsymbol{E}^{l,r}(m) = e_{forward}(\boldsymbol{B}_{diff}^{l,r}(m), Nyquits)$
- Compute
$$fc^{l,r}(m) = find(f)\ such\ as\ e_{backward}(\boldsymbol{B}_{diff}^{l,r}(m), f) = \boldsymbol{E}^{l,r}(m) * 10^{-DiffuseFcThreshold/10})$$

To compute BRIR default value of $DiffuseFcThreshold = -20$

For default values:

- $fc_k^{l,r}(0) = Nyquist = F_s/2$, no cut-off frequency for the first block
- $fc_k^{l,r}(m) = \max_{l,r}\left(fc_k^{l,r}(m)\right)$, same value for left, right filters.
- $fc^{l,r}(m) = \max_{l,r}\left(fc^{l,r}(m)\right)$, same value for left, right filters.

7.4.7.3.6 Conversion from Time domain to SH CLDFB domain for parametric binauralizer

7.4.7.3.6.1 Overview

The conversion of the time domain head-related impulse response HRIR and BRIR data to CLDFB-domain spherical harmonic binaural rendering matrices is described here. The data is assumed to be at 48 kHz sampling rate.

7.4.7.3.6.2 HRIR conversion

The HRIR database consists of time domain binaural responses corresponding to a multitude of directions in anechoic conditions. The responses are assumed to be time-aligned. If not, time-alignment can be achieved by measuring the average group delay of each response pair, and temporally aligning all HRIR pairs in the dataset so that they have the same average group delay.

As the first step, a common group delay in samples is formulated as a median group delay value in samples over all directions and both left and right response channels, and this value is rounded to the nearest integer value.

Next, a reference impulse is generated that is zeros otherwise, but one at the sample corresponding to the common group delay.

The reference impulse and the HRIRs are then converted to a time-frequency representation using the CLDFB. Prior to this conversion, the responses are zero-padded at the end so that all time domain data is represented at the converted CLDFB domain responses.

Let us denote the CLDFB domain responses as $H'(b,n,j,s)$ where $b = 1,2 \ldots,60$ is the bin index, $n = 1,2,\ldots,N$ is the temporal slot index, $j = 1,2$ is the channel index, and $s = 1,2,\ldots,S$ is the source direction index in the HRIR database. $R(b,n)$ is the reference impulse CLDFB domain response. Non-equalized HRTFs $H_{noneq}(b,j,s)$ for each direction is formulated by

$$\alpha(b,j,s) = angle\left(\sum_{n=1}^{N} H'(b,n,j,s)R^H(b,n)\right)$$

$$H_{noneq}(b,j,s) = e^{i\alpha(b,j,s)}\sqrt{\frac{\sum_{t=1}^{T}|H'(b,n,j,s)|^2}{\sum_{t=1}^{T}|R(b,n,j)|^2}}$$

where i is the imaginary value.

Then, diffuse field equalization is performed by first determining a spatially even set of 240 directions in 3D, and then selecting for each of these directions the closest data point at the HRIR dataset. The indices of these data points are $s_{nearest}$, and the total number of these indices is denoted $S_{nearest} = 240$. A single data point of the HRIR set can be represented multiple times in the set $s_{nearest}$.

The HRTF set is then equalized by

$$E_{eq}(b) = \frac{1}{S_{nearest}}\sum_{j=1}^{2}\sum_{s_{nearest}}\left|H_{noneq}(b,j,s_{nearest})\right|^2$$

$$g'_{eq}(b) = \sqrt{\frac{1}{E_{eq}(b)}}$$

$$g_{eq}(b) = \min(g'_{eq}(b), 5g'_{eq,median})$$

where $g'_{eq,median}$ is the median of $g'_{eq}(b)$, and finally,

$$H(b,j,s) = H_{noneq}(b,j,s)g_{eq}(b)$$

The HRTFs are then transformed to the spherical harmonic representation. Let us denote matrix **S** as a matrix of size $(16, S)$ containing at its columns the third order spherical harmonic encoding gains for each of the S source directions. Let us denote matrix $\mathbf{H}(b)$ as a $(2, S)$ matrix having the HRTF gains $H(b,j,s)$ as its elements, where index j populates the first axis and s the second axis. For bins b where the bin center frequency exceeds 2kHz, then the elements of $\mathbf{H}(b)$ are $|H(b,j,s)|$.

The spherical harmonic domain rendering matrix for bin $b$is then

$$\mathbf{H}_{sh}(b) = \mathbf{S}^{-1}\mathbf{H}(b)$$

where $\mathbf{S}^{-1}$ is the pseudoinverse of **S**. These matrices can then be used in determining HRTFs as described in clause 7.2.2.3.6 where $\mathbf{H}_{sh}$ is used as $\mathbf{M}_{HOA3bin}$.

7.4.7.3.6.3 BRIR conversion

Typical HRTF datasets are obtained in a high number of spatial positions. For example, by use of simulations or moving loudspeakers arcs at the anechoic chamber, one can obtain up to hundreds or thousands of measurement positions, resulting in a high spatial precision.

On the contrary, typical BRIR datasets are obtained in listening rooms with lesser number of loudspeakers. Typical surround loudspeaker arrangements have in the range of 6-12 loudspeakers, whereas some listening rooms are equipped with extensive arrangements over 20 loudspeakers.

However, even the setups with 20 to 30 loudspeakers are not close to the spatial fidelity of the typical HRTF sets. Therefore, the parametric renderer selects a hybrid approach: The directional sound is rendered with HRTF-based data, but the rendering is modified so that it attains characteristics of the given BRIR dataset. The spatial rendering in such a scheme is described in clause 7.2.2.3. Here, it is described how these relevant characteristics are estimated from a given BRIR dataset.

First, from the given BRIR set, five responses are selected, which are the responses closest to reference positions 30°, 0°, -30°, 110°, -110° at the horizontal plane. Other BRIR responses are discarded. Then, the BRIR set is high-pass filtered with a 5th order Butterworth high-pass filter with the cut-off frequency at 80 Hz.

Next, the maximum energy sample indices of the BRIRs are formulated. The energy response of each of the five BRIR pairs $s = 1,..,5$ is formulated as a $e(t,s) = |r(t,s,1)|^2 + |r(t,s,2)|^2$ where $r(t,s,j)$ is the BRIR of index s, t is the sample index, and j is the left or right ear channel index.

A 120-sample length lowpass filter window is determined by

$$w_h(t) = \sin\left(\frac{\pi t}{120}\right)$$

Then, $e(t,s)$ is convolved with $w_h(t)$ to find temporally smoothed energies, and the maximum energy position of the BRIR response is determined based on the maximum energy of the temporally smoothed response, for each s independently.

Then, each of the BRIR responses are independently truncated from the beginning so that the maximum energy position corresponds to the 61st sample of the truncated response. This procedure aligns the obtained BRIR responses so that the direct sound portion is aligned to a first frame ranging from sample indices 1 to 120.

Next, the BRIR set is transformed to a time-frequency representation via STFT, using 120-sample length FFT operation, 60-sample length hop size and a complex 120-length window that is defined by

$$w(t) = e^{-\frac{i\pi t}{120}}w_h(t)$$

In the above STFT procedure, no zero-padding is applied at the beginning. In other words, the first frame temporal indices range from 1 to 120. Only the first 60 bins of the STFT operation are preserved. These bins have the same centre frequencies as the CLDFB frequency domain data, and the same temporal resolution.

The energies of the STFT operation results are formulated by

$$E(b,n) = \sum_{s=1}^{5}\sum_{j=1}^{2}|S(b,n,s,j)|^2$$

where $S(b,n,s,j)$ is the STFT domain (as described in the foregoing) BRIR where b is the bin index and n is the temporal index. The energy data is also converted to decibels by

$$E_{dB}(b,n) = 10\log_{10}E(b,n)$$

The BRIR analysis is performed based on this energy data for each frequency bin b independently. First, for each bin b, the start index of the late reverberation $n_{late}(b)$ is determined as the one that has the maximum energy $E_{db}(b,n)$ at range $n = 5, \dots, 15$.

Next, a line is fitted in the least squares sense to all sequences of $E_{db}(b,n)$ that have the starting point $n_{late}(b)$ and ending point $n_{end}(b,l)$. The value $n_{end}(b,l)$ is tested for each $l > 0$ so that $n_{end}(b,l) = n_{late}(b) + 4 + l$, up until $n_{end}(b,l)$ is the temporal length of $E_{db}(b,n)$.

Then the best line $l_{opt}(b)$ is selected. First, an estimation error value is defined by

$$E_{dB,rmse}(b,l) = \sqrt{\frac{\sum_{n=n_{late}(b)}^{n_{end}(b,l)}\left|E_{db}(b,n) - \hat{E}_{db}(b,n,l)\right|^2}{n_{end}(b,l) - n_{late}(b)}}$$

Then, $l_{opt}(b)$ is selected as that l which has the largest $n_{end}(b,l)$ that satisfies the condition $E_{dB,rmse}(b,l) < E_{dB,rmse,min}(b) + 1$, where $E_{dB,rmse,min}(b)$ is the minimum of $E_{dB,rmse}(b,l)$ over all l. The purpose of this procedure is to find the longest sequence that does not contain the noise floor.

The selected values are then collected by $n_{sel}(b) = n_{end}(b, l_{opt}(b))$. These values are then median-filtered by a 5-length median filter, truncated at the edges (e.g., 2nd bin has median filter from 1st bin to 4th bin), resulting in $n_{sel,med}(b)$.

Then, the reverberation time $T_{60}(b)$, for each bin b is determined based on fitting a linear line to $E_{db}(b,n)$ for indices $n = n_{late}(b), \dots, n_{sel,med}(b)$, and the $T_{60}(b)$ is the time when that line reduces by 60 dB.

For highest bins the $T_{60}(b)$ estimates may be noisy. Therefore, a linear fit is performed for $T_{60}(b)$ in range $b = 30, \dots, 50$, resulting in $T_{60,linear}(b)$. Then the $T_{60}(b)$ values for $b \geq 30$ are modified so that the resulting $T_{60}(b)$ estimates linearly interpolate from $T_{60}(b)$ to $T_{60,linear}(b)$ at range $b = 30, \dots, 50$ and then after it remain at $T_{60,linear}(b)$. The result is the estimated set of reverberation times.

Then, early and late part energies are formulated by

$$E_{early}(b) = \sum_{n=1}^{25}E(b,n)$$

$$E_{late}(b) = \sqrt{2}\left(\left(\sum_{n}E(b,n)\right) - E_{early}(b)\right)$$

Then, the reference HRTF energies are formulated from HRTFs as determined in clause 7.4.7.3.6.2, which are stored in spherical harmonic representation, denoted $\mathbf{H}_{sh}(b)$. From the spherical harmonic representation HRTF pairs are determined by $\mathbf{h}(b,s) = \mathbf{H}_{sh}(b)\boldsymbol{v}(s)$ where $\boldsymbol{v}(s)$ are the spherical harmonic encoding gains for the five directions s (i.e., 30°, 0°, -30°, 110°, -110°). Energy $E_{early,ref}(b)$ is the mean energy of these five HRTF pairs, i.e.

$$E_{early,ref}(b) = \frac{1}{5}\sum_{s=1}^{5}\mathbf{h}(b,s)\mathbf{h}^H(b,s)$$

These reference HRTF pairs are further modified by $E_{early}(b) \coloneqq \max\left(E_{early}(b), 0.2E_{early,median}\right)$ where $E_{early,median}$ is the median of $E_{early}(b)$ over all b.

Then, early part energy correction gains are formulated by

$$g_{ene,early}(b) = \frac{E_{early}(b)}{E_{early,ref}(b)}$$

An exception is for $b = 1$ where

$$g_{ene,early}(1) = \frac{E_{early}(1)10^{1/10}}{E_{early,ref}(1)}$$

which adds 1 decibel to compensate for the effect of the applied high-pass filter.

7.4.8 Room acoustics parameters

7.4.8.1 Overview

The late reverb is driven by the set of parameters comprising of:

- RT60 – indicating the time that it takes for the reflections to drop 60 dB in energy level,
- DSR – diffuse to source signal energy ratio,
- Pre-delay – delay at which the computation of DSR values was made. Can be interpreted as the threshold between early reflections and late reverberation phase.

Spatialized, rotation-responsive, first-order early reflections can be added when using multichannel input (any configuration accepted). The early reflections rendering is determined by several parameters that drive a shoebox model using the image-source method. The set of parameters consists of:

- 3D rectangular virtual room dimensions,
- Broadband energy absorption coefficient per wall,
- Listener origin coordinates within room (optional),
- Low-complexity mode (optional) – favours efficient early reflection rendering over spatial accuracy.

Figure 7.4-9 illustrates the main reverberation properties with relevant reverberation synthesis control parameters indicated.

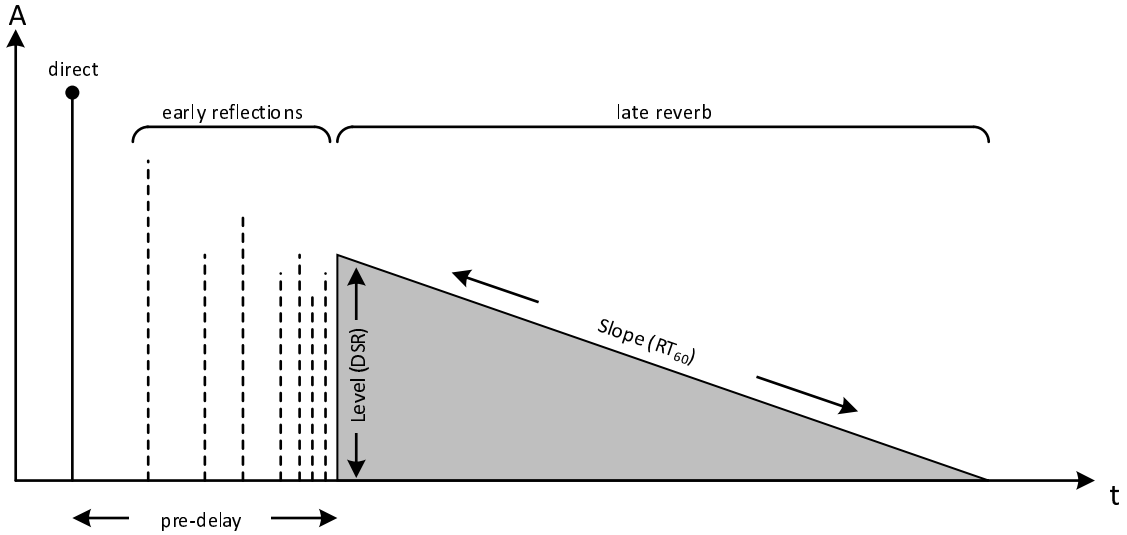


Figure 7.4-9: Simple representation of the main reverberation properties

Room acoustics parameters are provided to the renderer as metadata. Two metadata formats are supported in the IVAS decoder/renderer implementation: binary renderer config metadata format, and text renderer config metadata format (see [12], clause 5.14.1 and 5.14.2 respectively). Regardless of the metadata format, the general metadata processing is

shared, as discussed in 7.4.8.2 for late reverb, and in 7.4.8.3 for early reflections. Both metadata formats support multiple acoustic environment datasets, allowing for selecting between such acoustic environments.

7.4.8.2 Late reverb parameters

The reverberation characteristics of a typical room are strongly frequency dependent. Therefore, RT_{60} and DSR parameters are specified per frequency band. For different room characteristics different frequency grids to specify reverb parameters might be suitable. For instance, in some cases octave or even coarser girds could provide sufficient frequency resolution, while in the other much finer resolution is required (e.g., in order to model strong local resonances in a room). Hence, reverb metadata precedes with data describing such frequency grid. This allows for reusing frequency grids across multiple room acoustics environments. The frequency grid metadata can be represented as:

- default grid identifier – indicating a predefined frequency grid to be used,
- set of individual frequencies – forming a frequency grid,
- start-hop-amount data – indicating start frequency, frequency hop factor, and the number of frequency bands in the grid.

The default frequency grids are provided in Table 7.4-4. It is possible to select a subsection of a default frequency grid by specifying an offset and number of frequency bands to be used.

Table 7.4-4: Default frequency grids for late reverb parameters

ID	Centre frequencies [Hz]	# bands	Description
0	31.5, 63, 125, 250, 500, 1000, 2000, 4000, 8000, 16000	10	Octave – ISO
1	25, 50, 100, 200, 400, 800, 1600, 3150, 6300, 12500	10	Octave - alternative
2	20, 25, 31.5, 40, 50, 63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000, 20000	31	1/3 octave – ISO
3	25, 100, 400, 1600, 6400	5	2 octave – ISO
4	125, 250, 500, 1000, 2000, 4000	6	Octave subset
5	250, 250, 2500	3	
6	27, 56, 89, 126, 168, 214, 265, 323, 387, 459, 539, 628, 727, 839, 963, 1101,1256, 1429, 1621, 1836, 2077, 2345, 2644, 2978, 3351, 3767, 4232, 4750, 5329, 5975, 6697, 7502, 8401, 9405, 10525, 11775, 13171, 14729, 16468, 18410, 20577	41	1 ERB scale
7	27, 89, 168, 265, 387, 539, 727, 963, 1256, 1621, 2077, 2644, 3351, 4232, 5329, 6697, 8401, 10525, 13171, 16468, 20577	21	2 ERB scale
8	50, 150, 250, 350, 450, 570, 700, 840, 1000, 1170, 1370, 1600, 1850, 2150, 2150, 2500, 2900, 3400, 4000, 4800, 5800, 7000, 8500, 10500, 13500	25	Bark scale

Each room acoustic environment contains a frequency grid identifier, a set of RT_{60} and DSR parameters per frequency band, and a single pre-delay parameter. The pre-delay parameter indicates the delay at which the DSR values were calculated.

The pre-delay parameter provided for DSR values does not necessarily correspond to the actual late reverb render delay, regardless of the synthesis domain and algorithm used (see clauses 7.3.3 and 7.3.4, respectively). Therefore, the reverb output energy needs to be adjusted to maintain the room acoustics characteristics. The correction factor is applied to the DSR values corresponding to the relevant RT_{60} as derived from exponential decay model:

$$\mathrm{DSR}_{rend}(f) = DSR_{in}(f)e^{-2\alpha(f)(t_{rend}-t_{in})} \tag{7.4-76}$$

where

$$\alpha(f) = \frac{\ln\left(10^{-\frac{60}{20}}\right)}{RT_{60}(f)} \tag{7.4-77}$$

t_{rend} denotes late reverb render delay, and t_{in} pre-delay parameter used for computing DSR values.

7.4.8.3 Early reflections parameters

The required parameters for using early reflections are the room dimensions, and the surface absorption coefficients. The room dimension parameters are provided as a set of three, that refer respectively to “length”, “width” and “height” of a rectangular room geometry.

The surface absorption parameters need to come as a set of six, and consist in a number between 0 and 1, where 0 means total reflectivity and 1 means total absorption. These coefficients are applied as broadband parameters even though absorption rates usually vary with sound frequency. This is to reduce the number of processing steps needed to compute and process early reflections. The six absorption rates coefficients correspond to the shoebox surfaces according to the axis order used by the coordinate system (see Fig. 7.3-5). For quick reference, Table 7.4-5 shows the order assignation:

Table 7.4-5: Shoebox surface reference for absorption coefficients

Coefficient #	ID	Surface Description	Axis
0	**W1**	Front Wall	$+x$
1	**W2**	Back Wall	$-x$
2	**W3**	Left Wall	$+y$
3	**W4**	Right Wall	$-y$
4	**W5**	Ceiling	$+z$
5	**W6**	Floor	$-z$

Optionally, two additional fields can be provided. The first one is the listener origin position (the point in the shoebox where the geometrical reflections patterns are collected, defaulting to the middle point of the room model at a height of 1.5mt). This field consists of three parameters denoting the cartesian coordinate of placement of the listener probe (x, y, z). The coordinate system used for the probe positioning is illustrated in clause 7.3.5.2, showing the direction of the dimension axes, and the zero-point as the middle of the “floor” surface. While the x (length-axis) and y (width-axis) cartesian coordinates can be either positive or negative, the z coordinate (height-axis) can only be positive.

The second optional parameter field is the Low Complexity mode flag (default is off). The low-complexity mode flag can be activated if a less-intensive early reflection computation is desired. By activating this flag, the processing cost of computing early reflections is decreased by up to 50%, at a minimal perceptual impact. Details on this functionality are given in clause 7.3.5.6.

7.4.9 Use of custom loudspeaker layouts

7.4.9.1 General

The multi-channel loudspeaker rendering supports using custom loudspeaker layouts up to 16 output channels (inclusive of LFE channels). The custom loudspeaker layouts are described by azimuth $\theta_{LS}(i)$ and elevation $\phi_{LS}(i)$ angles, where i is the loudspeaker channel. Any LFE channels are specified separately and are not associated with a position. IVAS only supports 1 output LFE channel. This information must be fed to the decoder by the user via the API function `IVAS_DEC_FeedCustomLsData()` which configures the decoder with the structure `IVAS_CUSTOM_LS_DATA`:

```
typedef struct _IVAS_LS_CUSTOM_LAYOUT
{
    int16_t num_spk;
    float azimuth[IVAS_MAX_OUTPUT_CHANNELS];
    float elevation[IVAS_MAX_OUTPUT_CHANNELS];
    int16_t num_lfe;
    int16_t lfe_idx[IVAS_MAX_OUTPUT_CHANNELS];

} IVAS_CUSTOM_LS_DATA;
```

The input custom loudspeaker data first undergoes sanity checks and validation. A check is performed to verify that the number of input values does not exceed the supported number of output loudspeaker channels or LFE channels. Next, the input azimuth and elevation values are wrapped to the following bounds:

$$-180° < \theta_{LS} \leq 180°$$

$$-90° \leq \phi_{LS} \leq 90°$$

Finally, any duplicate values present after wrapping are removed and the decoder is configured with the updated data. Once this is complete, rendering is handled based on the input format as described below.

7.4.9.2 MASA, OMASA, and McMASA

For MASA, OMASA, and multi-channel (in McMASA mode) formats, the multi-channel loudspeaker rendering is performed using the multi-channel renderer described in clauses 6.5.7.2, 6.7.2.4.2, and 6.9.7.3. For the direct sound part, the multi-channel renderer determines the panning gains using VBAP (see clause 7.2.1.2), which supports custom loudspeaker layouts. For the non-directional sound part, the multi-channel renderer determines the rendering gains using the diffuse response gains (see clause 6.5.7.2.4), which also supports custom loudspeaker layouts.

7.4.9.3 ParamMC and Parametric Upmix MC

For multi-channel in the ParamMC and Parametric Upmix modes, the loudspeaker conversion processing uses the mixing matrix obtained from the MC format conversion (cf. clause 6.7.7.2) for custom layouts using EFAP (cf. clause 7.2.1.3).

7.4.9.4 Remaining formats

For rendering of remaining formats to custom loudspeaker output EFAP (cf. clause 7.2.1.3) is used to compute a rendering matrix. In all cases an EFAP instance is initialized with the output custom loudspeaker setup data configured by the user.

For multi-channel to custom loudspeaker output, EFAP is used to pan each individual loudspeaker of the input format as a virtual object on to the output custom loudspeaker layout. A rendering matrix can thus be constructed mapping each input loudspeaker to the output layout. Any input LFE channels are considered separately from the input loudspeaker channels and are either routed to the output LFE channel if present or mixed to all output channels with a factor of $\frac{1}{N_{LS}}$ where is the number of output loudspeakers *excluding* the number of LFE channels. Since the input loudspeaker positions are fixed, such a rendering matrix must only be computed once for a given combination of input and output configurations.

For object-based audio, the objects are treated as virtual objects like for multi-channel input, however since the object positions vary, a precomputed matrix cannot be used. Instead, the input object positions (azimuth, elevation) are used from the metadata to input to EFAP as the panning position. A similar rendering matrix may be constructed with each input channel associated to one object at the position specified by the metadata in the current audio frame.

For scene-based audio, the EFAP module is used to compute the AllRAD decoding matrix (cf. clause 7.2.1.4) which is used to render the input audio to the output custom loudspeaker layout.

7.5 Pre-rendering

Pre-rendering in IVAS is generally the operation to transform one or more audio inputs into another encoder input format supported by the IVAS codec. Pre-rendering is possible using the rendering tools available in the codec framework. Pre-rendering can be desirable when several audio inputs are to be encoded using a single encoder instance.

IVAS pre-rendering methods allow pre-rendering into scene-based audio (SBA) format, multichannel (MC) format, binaural formats (cf. clause 7.2.2.1). Pre-rendering to metadata-assisted spatial audio (MASA) format using the IVAS pre-rendering methods may only be performed if one or more of the audio inputs to the pre-rendering is a MASA format input. The latter option allows to avoid a pre-rendering delay of 5 ms, if one or more of the audio inputs to the pre-rendering is a MASA format input. Such delay would be incurred when rendering MASA into another format, such as SBA. Therefore, IVAS pre-rendering does not increase the overall codec delay.

Table 7.5-1 summarises the available prerendering combinations. Channel based formats in the table refers to mono, stereo, supported MC layouts or custom loudspeaker layouts.

Table 7.5-1: Overview of Pre-rendering from/to supported formats

Input format	Output Format				
		Channel based	**SBA**	**MASA**	**BINAURAL***
	Channel based	Clause 7.5.5.4	Clause 7.5.2.2	Clause 7.5.3.3	Clause 7.5.4.1
	SBA	Clause 7.5.5.2	N/A	Clause 7.5.3.4	Clause 7.5.4.1
	MASA	Clause 7.5.5.3	Clause 7.5.2.3	Clause 7.5.3.5.1	Clause 7.5.4.3
	ISM	Clause 7.5.5.1	Clause 7.5.2.1	Clause 7.5.3.2	Clause 7.5.4.2

7.5.2 Pre-rendering into SBA format

7.5.2.1 ISM to SBA rendering

For rendering of ISM input to SBA, the real spherical harmonic response $\boldsymbol{Y}_{lm}(\theta, \phi)$ of the input ISM objects is derived using the position metadata of azimuth θ and elevation ϕ for a given frame. A rendering to SBA is then performed via multiplication of the object audio with the spherical harmonic gains to obtain the ambisonics signal for the given order:

$$x_{\mathrm{SBA},(\mathrm{l}^2+\mathrm{l}+\mathrm{m})}(i) = \sum_{n=1}^{N_{ISM}} x_{ISM,n}(i)\boldsymbol{Y}_{lm,n}(\theta_n, \phi_n)$$

Where N_{ISM} is the number of ISM objects, n is the object index, l is the degree and m is the order of the spherical harmonic.

7.5.2.2 Channel-based audio to SBA rendering

For rendering of channel-based audio, including mono, stereo, supported multichannel layouts (clause 4.3.2) and custom loudspeaker layouts (clause 7.4.9) input to SBA, the real spherical harmonic response $\boldsymbol{Y}_{lm}(\theta, \phi)$ of the input loudspeaker channels is derived using the azimuth θ and elevation ϕ for the respective loudspeaker positions. A rendering to SBA is then performed via multiplication of the loudspeaker signals with the spherical harmonic gains to obtain the ambisonics signal for the given order:

$$x_{\mathrm{SBA},(l^2+l+m)}(i) = \sum_{n=1}^{N_{LS}} x_{MC,n}(i)\boldsymbol{Y}_{lm,n}(\theta_n, \phi_n)$$

Where N_{LS} is the number of input loudspeakers, n is the loudspeaker index, l is the degree and m is the order of the spherical harmonic.

7.5.2.3 MASA to SBA rendering

For rendering input MASA signals to SBA, the MASA spatial metadata is used with the transport channels to synthesise the Ambisonics channels according to clause 6.5.7.2.

7.5.3 Pre-rendering into MASA format

7.5.3.1 Overview

7.5.3.2 ISM to MASA rendering

The ISM format input audio signal is converted into MASA spatial audio format. The MASA spatial metadata parameters for the input ISM object signals are determined from the input audio object signals as described in clause 5.9.3.1. The MASA spatial audio format transport audio signal is determined from the input audio object signals as described in clause 5.9.3.3. The signal energy metadata parameter for the input audio signal is determined as described in clause 5.5.2.3 from the determined MASA spatial audio format transport audio signal.

7.5.3.3 Multi-channel to MASA rendering

The multi-channel format input audio signal is converted into MASA spatial audio format. The MASA spatial metadata parameters for the input multi-channel audio signal are determined from the input multi-channel signal as described in clause 5.7.3.3. The MASA spatial audio format transport audio signal is determined from the input multi-channel

signals as described in clause 5.7.3.5. The signal energy metadata parameter for the input multi-channel signal is determined as described in clause 5.5.2.3 from the determined MASA spatial audio format transport audio signal.

7.5.3.4 SBA to MASA rendering

The SBA format input audio signal is converted into MASA spatial audio format. The MASA spatial metadata parameters for the SBA audio signal are determined from the SBA input signals as described in clause 5.9.3.1 with the following change: Instead of determining the first order spherical harmonic signals from the object signals, the first four channels $s(n,i), i \in [0,\ 3]$ of the SBA format input audio signal are used. This is done for all SBA format input signals regardless of if they are first, second, or third order Ambisonics (SBA) signals.

The MASA spatial audio format transport audio signal is determined from these signals $s(n,i)$. If the number of transport audio signal channels in the target MASA format is 1, the transport audio signal is $s(n,0)$. If the number of transport audio signal channels in the target MASA format is 2, the transport audio signals $s_{MASA}(n,j), j \in [0,\ 1]$ are determined as $s_{MASA}(n,0) = 0.5\big(s(n,0) + s(n,1)\big)$ and $s_{MASA}(n,1) = 0.5\big(s(n,0) - s(n,1)\big)$.

The signal energy metadata parameter for the SBA input format signal is determined as described in clause 5.5.2.3 from the determined MASA spatial audio format transport audio signal.

7.5.3.5 Merging audio signals rendered into MASA format

7.5.3.5.1 MASA input to MASA output rendering

Here, MASA input format refers to the input of the pre-rendering and MASA output format refers to the output of the pre-rendering and thus input to the IVAS encoder.

MASA input format to MASA output format rendering is a pre-processing adapting the number of MASA spatial audio transport signals if the number of transport signals is different in the input MASA spatial audio format and in the output MASA spatial audio format. The energy metadata parameter of the transport audio signal is determined as described in clause 5.5.2.3. The spatial metadata parameters are sanitized to eliminate invalid input values.

7.5.3.5.2 MASA signal merge

The pre-rending of multiple input audio signals to MASA output format takes two or more input audio signals that are in different audio signal formats (e.g., MASA, SBA, multi-channel, or ISM), determines spatial metadata parameters and transport audio signals for each input signal, and generates combined spatial metadata parameters that are associated with the combined transport audio signal formed based on the transport audio signals that are determined from the input audio signals. The spatial metadata parameters and transport audio signals for an ISM format input audio signal are determined according to clause 7.5.3.2. The spatial metadata parameters and transport audio signals for a multi-channel format input audio signal are determined according to clause 7.5.3.3. The spatial metadata parameters and transport audio signals for an SBA format input audio signal are determined according to clause 7.5.3.4. The spatial metadata parameters for a MASA format input audio signal are determined according to clause 7.5.3.5.1. The spatial metadata parameters determined from the input signals of different formats include MASA spatial audio metadata parameters, such as direction and direct-to-total energy ratio, and a transport audio signal energy metadata parameter.

The combining of multiple input audio signals possibly in different formats operates in steps, each step merging two audio signals.

The method used to combine two spatial audio signals depends on the original format of the input audio signal: When combining an input audio signal of ISM format with an input signal in a different format (regardless of if the second signal is a second input signal to the merge or a result of merging two signals of different input formats), and the another signal has exactly one direction parameter, the method of clause 7.5.3.5.3 is used. For combining audio signals in other formats (e.g., SBA format and multi-channel format, SBA format and MASA format, multi-channel format and MASA format the method of clause 7.5.3.5.4 is used.

The spatial metadata determined by combining the metadata of the two audio signals of possibly different formats is associated with the audio signal formed by combining the audio signals in the two input audio signals. The signal energy metadata parameter of the combined audio signal is determined from the signal energy parameters $E_0(b,m)$ and $E_1(b,m)$ of the two audio signals that are combined with

$$E_{combined}(b,m) = E_0(b,m) + E_1(b,m)$$

7.5.3.5.3 Merging MASA metadata from ISM format input with inputs of other formats

The first input audio signal is originally in a format other than ISM audio format. I.e., it is in SBA format, multi-channel format, or MASA format, or a combination of audio signals of such formats. The second audio signal is originally in ISM format. The MASA format spatial metadata parameters determined from the input signals contain exactly one parameter direction. The merged spatial metadata parameter values are determined as described in clause 5.9.3.2. A combined transport audio signal is formed from the transport signals determined from the input audio signals by summing the first transport audio signal with the second transport audio signal. The combined spatial metadata parameters are associated with this combined transport audio signal.

7.5.3.5.4 Merging MASA metadata from other input formats

This clause describes merging of MASA metadata in the general case, i.e., when both input audio signals are in other format than ISM format, or if the MASA metadata parameters for one or both of the input signals contain two parameter directions.

The spatial metadata parameters in MASA format are determined for both input signals based on their original input format, as described in clause 7.5.3.5.2. This determines also the transport audio signals for both input audio signals and signal energy metadata parameters of the two transport audio signals.

The merging of MASA spatial metadata parameters from two input signals of possibly different formats is done for each parameter coding band b and parameter subframe m independently.

First, weight values w_i for both input audio signals $i \in \{0,1\}$ (w_0 for the first input signal and w_1 for the second input signal) are determined based on multiplying the determined signal energy metadata parameters E_i (E_0 for the first input signal and E_1 for the second input signal) of the transport audio signals by the determined direct-to-total energy ratio metadata parameter $r_{dir;i}$ ($r_{dir;0}$ for the first input signal and $r_{dir;1}$ for the second input signal) with

$$w_i(b,m) = E_i(b,m) r_{dir;i}(b,m,0)$$

for metadata with one parameter direction, and with

$$w_i(b,m) = E_i(b,m) r_{dir;i}(b,m,0) + E_i(b,m) r_{dir;i}(b,m,1)$$

for metadata with two parameter directions.

The weight values w_0 and w_1 for the input audio signals are compared for controlling the combining of the metadata parameters determined from the input audio signals. The output spatial metadata parameters are generated by selecting the parameters associated with the input signal i with the larger weight value w_i. If the weight value w_0 associated with the first input audio signal is larger than the weight value w_1 associated with the second input audio signal, i.e., $w_0 > w_1$, the spatial metadata parameters determined from the first input audio signal are selected, and otherwise the spatial metadata parameters determined from the second input audio signal are selected. The resulting combined spatial metadata parameters include a direction parameter (possibly defined as azimuth and elevation and/or direction index), a direct-to-total energy ratio parameter, a diffuse-to-total energy ratio parameter, and possibly also surround and spread coherence parameters.

If either of the metadata determined for the input audio signals has two parameter directions, the output combined spatial metadata shall have two parameter directions.

A combined transport audio signal is formed from the transport signals determined from the input audio signals by summing the transport audio signal of the first input audio signal with the transport audio signal of the second input audio signal. The combined MASA spatial metadata parameters resulting from the merging described in this clause are associated with this combined transport audio signal.

7.5.4 Pre-rendering into Binaural format

7.5.4.1 SBA and MC to Binaural rendering

For rendering of MC formats 5.1 and 7.1 with headtracking enabled, the Time Domain binaural renderer is used together with the loudspeaker positions as fixed object positions as described in clause 7.2.2.2. Otherwise Crend binaural renderer is used (7.2.2.5).

7.5.4.2 ISM to Binaural rendering

For rendering of ISM input to binaural, the Time Domain binaural renderer is used together with the input ISM metadata as described in clause 7.2.2.2.

7.5.4.3 MASA to Binaural rendering

For rendering input MASA signals to binaural, the MASA spatial metadata is used with the transport channels to synthesise the binaural channels according to clauses 6.5.7.1 and 7.2.2.3.

7.5.5 Pre-rendering into MC format

7.5.5.1 ISM to MC rendering

For rendering of ISM input to MC, panning gains for the given loudspeaker layout $\boldsymbol{G}_{EFAP}(\theta, \phi)$ of the input ISM objects are computed using EFAP (clause 7.2.1.3) by supplying the position metadata of azimuth θ and elevation ϕ for a given frame. A rendering to MC is then performed via multiplication of the object audio with the EFAP panning gains to obtain the output loudspeaker signals:

$$X_{MC(i)} = \sum_{n=1}^{N_{ISM}} \mathrm{X}_{\mathrm{ISM,n}}(\mathrm{i})\mathbf{G}_{\mathrm{EFAP,n}}(\theta_{\mathrm{n}}, \phi_{\mathrm{n}})$$

Where N_{ISM} is the number of ISM objects and n is the object index.

7.5.5.2 SBA to MC rendering

Rendering from SBA to MC is performed using ALLRAD as described in clause 7.2.1.4.

7.5.5.3 MASA to MC rendering

For rendering input MASA signals to MC, the MASA spatial metadata is used with the transport channels to synthesise the loudspeaker channels according to clause 6.5.7.2.

7.5.5.4 MC to MC rendering

In case of format conversion between multichannel formats, the matrix $\boldsymbol{M}_{conv}$ as specified in clause 6.7.7 is used to convert between formats.

7.6 Split rendering

7.6.1 Overview

IVAS supports split rendering wherein the process of binaural rendering and headtracking is split between a main device (pre-renderer) and a light-weight head-worn device (post-renderer). The split-rendering architecture in IVAS is such that the complexity at the post-renderer is substantially less than the complexity of the IVAS decoder and renderer.

There are two architectures of split rendering supported in IVAS. The first architecture, described in Figure 7.6-1, extends IVAS decoder and internal renderer to perform the pre-rendering part of split renderer, whereas the post rendering is done using a separate post-renderer. In the second architecture, described in Figure 7.6-2, the IVAS decoder runs in pass-through mode and the IVAS external renderer is extended to perform the pre-rendering part of split renderer.

Split rendering in IVAS is supported with all immersive input formats and 48kHz sampling rate. For configuration of split rendering, a degree of freedom (DOF) may be specified at the main device (pre-renderer), which determines the rotation axes for which pose correction can be performed at the light-weight device (post-renderer). The DOF value specifies the number of axes compensated, and a special value of 0 DOF means no pose correction capability is available at the post-renderer. The pre-rendering configuration is further described in 7.6.2.

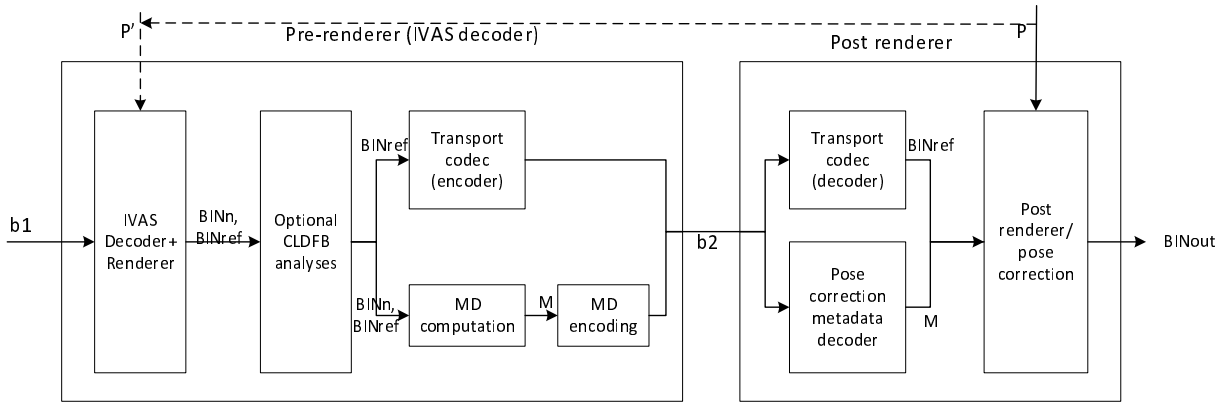


Figure 7.6-1: Split rendering architecture with IVAS decoder

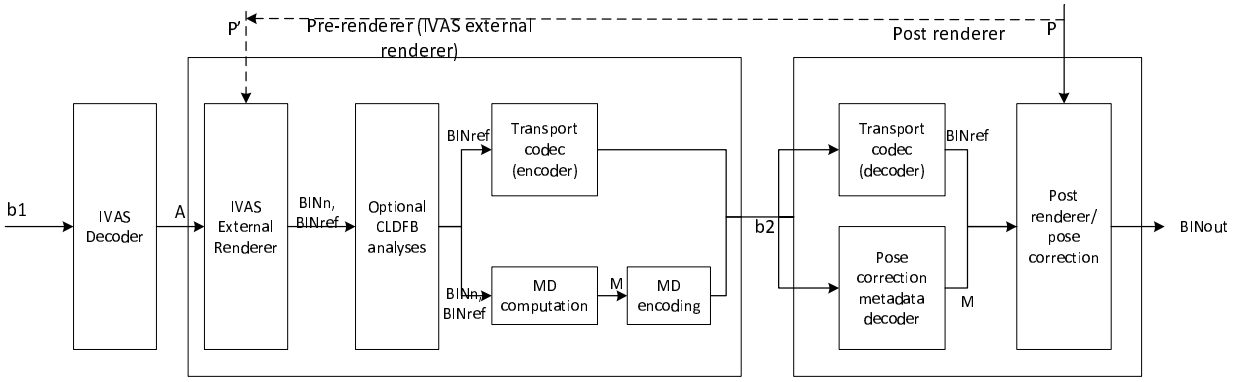


Figure 7.6-2: Split rendering architecture with IVAS external

7.6.2 Split pre-rendering

7.6.2.1 Overview

As shown in Figure 7.6-1, the pre-renderer takes IVAS bitstream b1 as input and runs IVAS decoder to generate the immersive audio A. The pre-renderer also obtains a reference head Pose P', associated with the user of the light-weight device or post-renderer, via a backchannel from post-renderer to pre-renderer. The post-renderer may encode the reference head pose into a pose bitstream bp, in which case P' shall be obtained by decoding bp. Alternatively, the reference pose can also be a static value (the default is forward-looking pose, with rotations along yaw, pitch and roll axis set to 0 degrees). Once the reference head pose is obtained, a reference binaural signal is generated with the reference head pose P' and the default binaural renderer for a given format. Next if pose correction is enabled, one or more binaural pre-renditions are computed with the same default binaural renderer, corresponding to one or more probing head poses P_n , where $n > 1$ and P_n is computed by adding fixed offsets to the reference pose P' such that each probing pose differs from the reference pose along one rotation axis only, as described in Table 7.6-1. If pose correction is not enabled, then the additional pre-renditions are not generated.

If the rendering happens in time domain, then all binaural signals are converted to CLDFB domain and the CLDFB domain binaural signals are fed to the metadata computation block, which computes metadata M such that all binaural pre-renditions can be reconstructed using the reference binaural signal and metadata M.

The reference binaural signal is either coded with the Low Complexity Low Delay (LCLD) codec or the Low Complexity Communication Codec plus (LC3plus) depending on the default transport codec settings, as described in Table 7.6-2. Alternatively, the codec can be set explicitly via the split rendering interface and either LCLD, LC3plus or

PCM can be used with any IVAS immersive input format. The reconstruction metadata M is quantized and coded by the metadata encoder block using the symmetries in probing poses and metadata. The encoded reference binaural signal and encoded metadata are multiplexed to generate the split rendering bitstream $b2$ that is transmitted to the post-renderer.

The reference head pose P' is coded in the bitstream $b2$ as part of metadata M, whereas the offsets to compute probing poses from the reference pose are static and known a priori to both pre-renderer and post-renderer.

Table 7.6-1: Probing poses configuration

1.1.1 DOF	1.1.2 Default Axes	1.1.3 Total number of probing poses	1.1.4 Offsets from reference pose {yaw_offset, pitch_offset, roll_offset} in degrees
1.1.5 0 DOF	1.1.6 N/A	1.1.7 0	1.1.8 N/A
1.1.9 1 DOF	1.1.10 Yaw	1.1.11 2	1.1.12 {15, 0, 0}, {-15, 0, 0}
1.1.13 2 DOF	1.1.14 Yaw, Pitch	1.1.15 4	1.1.16 {15, 0, 0}, {-15, 0, 0}, {0, 15, 0}, {0, -15, 0}
1.1.17 3 DOF, HQMODE OFF	1.1.18 Yaw, Pitch, Roll	1.1.19 4	1.1.20 {15, 0, 0}, {-15, 0, 0}, {0, 10, 0}, {0, 0, 10}
1.1.21 3 DOF, HQMODE ON	1.1.22 Yaw, Pitch, Roll	1.1.23 6	1.1.24 {15, 0, 0}, {-15, 0, 0}, {0, 15, 0}, {0, -15, 0}, {0, 0, 15}, {0, 0, -15}

Table 7.6-2: Default split rendering transport codec configuration

1.1.25 IVAS Renderer domain (CLDFB domain OR Time (TD) domain)	1.1.26 Default split rendering transport codec
1.1.27 CLDFB	1.1.28 LCLD
1.1.29 TD	1.1.30 LC3plus
1.1.31 CLDFB + TD (for combined formats like OSBA and OMASA)	1.1.32 LCLD

7.6.2.2 Supported Split Rendering bitrates with LCLD or LC3plus codec

Table 7.6-3: Supported Split Rendering bitrates with the LC3plus codec

1.1.33 LC3plus Configuration	1.1.34 DoF	1.1.35 Split rendering total bitrate (kbps)	1.1.36 Split rendering frame size (ms)
1.1.37 5ms or 10ms frame duration	1.1.38 0	1.1.39 256, 384, 512	1.1.40 5 or 10
1.1.41 5ms or 10ms frame duration	1.1.42 1-3	1.1.43 384, 512, 768	1.1.44 20

Table 7.6-4: Supported Split Rendering bitrates with the LCLD codec

1.1.45 LCLD configuration	1.1.46 DoF	1.1.47 Split rendering total bitrate (kbps)	1.1.48 Split rendering frame size (ms)
1.1.49 5ms or 10ms or 20ms frame duration	1.1.50 0	1.1.51 256, 384, 512	1.1.52 5 or 10 or 20
1.1.53 20ms frame duration	1.1.54 1-3	1.1.55 384, 512, 768	1.1.56 20

7.6.2.3 Supported Split Rendering bitrates with PCM output

In this mode the pre-renderers output PCM binaural data and a separate metadata bitstream. The split rendering bitrate provided via API is the bitrate of pose correction metadata. This mode is only supported with 1 to 3 DOF configurations. 0 DOF is the equivalent of the IVAS binaural output.

Table 7.6-5: Supported Split Rendering bitrates with the PCM binaural output

1.1.57 DoF	1.1.58 Supported bitrate	1.1.59 Split rendering frame size (ms)
1.1.60 1	1.1.61 Greater than or equal to 50kbps	1.1.62 20
1.1.63 2	1.1.64 Greater than or equal to 66kbps	1.1.65 20
1.1.66 3	1.1.67 Greater than or equal to 128kbps	1.1.68 20

7.6.2.4 Split pre-rendering of SBA

When the IVAS input format is SBA, the IVAS decoder runs the default ambisonics to binaural internal renderer $n + 1$ times including one binaural rendition with reference head pose P' and n binaural renditions with n probing head poses, respectively, as described in Figure 7.6-1. SBA rendering is always in CLDFB domain and the $n + 1$ binaural signals in CLDFB domain are then fed to the metadata generator, as described in clause 7.6.3. The reference binaural signal is coded with the split rendering transport codec and the coded bits are combined with coded metadata bits to form split rendering bitstream.

When the external renderer is running in split pre-rendering mode then the IVAS decoder decodes the IVAS bitstream and outputs SBA-format audio. The external renderer then performs $n + 1$ ambisonics to binaural renditions, as described in Figure 7.6-2.

7.6.2.5 Split pre-rendering of MASA

When the IVAS input format is MASA, $n + 1$ MASA to binaural renditions are done by the default IVAS MASA renderer.

When the external renderer is running in split pre-rendering mode then the IVAS decoder decodes the IVAS bitstream and outputs MASA format audio. The external renderer then performs $n + 1$ MASA to binaural renditions, as described in Figure 7.6-2.

7.6.2.6 Split pre-rendering of CBA

When the IVAS input format is CBA, $n + 1$ multi-channel to binaural renditions are done by the default IVAS multi-channel renderer.

When the external renderer is running in split pre-rendering mode then the IVAS decoder decodes the IVAS bitstream and outputs channel-based format audio. The external renderer then performs $n + 1$ multi-channel to binaural renditions, as described in Figure 7.6-2.

7.6.2.7 Split pre-rendering of ISM

When the IVAS input format is ISM, $n + 1$ ISM to binaural renditions are done by the default IVAS ISM renderer. In this format, an ILD flag is set and used in split-rendering metadata computation as described in clause 7.6.3.

When the external renderer is running in split pre-rendering mode then the IVAS decoder decodes the IVAS bitstream and outputs ISM format audio. The external renderer then performs $n + 1$ ISM to binaural renditions, as described in Figure 7.6-2.

7.6.2.8 Split pre-rendering of OSBA

When the IVAS input format is OSMA, the objects and SBA signals are rendered to binaural separately and then added together to form the final binaural signal. The $n + 1$ ISM to binaural renditions are done using the default ISM renderer and the $n + 1$ SBA to binaural renditions are done using the default SBA renderer. In this format, an ILD flag is set and used in split-rendering metadata computation as described in clause 7.6.3.

7.6.2.9 Split pre-rendering of OMASA

When the IVAS input format is OMASA, objects and MASA signals are rendered to binaural separately and then added together to form the final binaural signal. The $n + 1$ ISM to binaural renditions are done by the default ISM renderer and the $n + 1$ MASA to binaural renditions are done by the default MASA renderer. In this format, an ILD flag is set and used in split-rendering metadata computation as described in clause 7.6.3.

7.6.3 Intermediate split renderer metadata format

7.6.3.1 Overview

Split rendering metadata is computed to perform pose correction on the reference binaural signal from head pose P' at the pre-renderer to the latest head pose P at the post-renderer. The metadata is computed in CLDFB domain, wherein the 60 CLDFB bands are grouped into 20 metadata bands with the mapping shown in Table 7.6-6.

Table 7.6-6**: Metadata banding**

1.1.69 Metadata bands	1.1.70 CLDFB bands
1.1.71 1	1.1.72 1
1.1.73 2	1.1.74 2
1.1.75 3	1.1.76 3
1.1.77 4	1.1.78 4
1.1.79 5	1.1.80 5
1.1.81 6	1.1.82 6
1.1.83 7	1.1.84 7
1.1.85 8	1.1.86 8
1.1.87 9	1.1.88 9
1.1.89 10	1.1.90 10
1.1.91 11	1.1.92 11
1.1.93 12	1.1.94 12, 13
1.1.95 13	1.1.96 14,15
1.1.97 14	1.1.98 16, 17, 18, 19, 20
1.1.99 15	1.1.100 21, 22, 23, 24, 25
1.1.101 16	1.1.102 26, 27, 28, 29, 30
1.1.103 17	1.1.104 31, 32, 33, 34, 35
1.1.105 18	1.1.106 36, 37, 38, 39, 40
1.1.107 19	1.1.108 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50
1.1.109 20	1.1.110 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60

The banded covariance computation between a binaural signal corresponding to head pose P_1 and a binaural signal corresponding to head pose P_2 , where P_1 and P_2 are any two poses from the $n+1$ head poses, is done as follows:

$$R_{y,p_1,p_2}(b) = \sum_{m=1}^{m=16} \sum_{k=k_0^b}^{k=k_1^b} S_{P_1}(m,k) S_{P_2}^H(m,k),$$

where, $S_{P_1}(m,k)$ is the [2x1] matrix of the binaural signal corresponding to head pose P_1 at time slot m and CLDFB band index k, $S_{P_2}^H(m,k)$ is the complex conjugate matrix of the binaural signal corresponding to head pose P_2 at time slot m and band index k, b is the metadata band index with range $1 \le b \le 20$, k_0^b and k_1^b are the start and end CLDFB band indices as per Table 7.6-6.

7.6.3.2 Metadata computation, quantization and coding

7.6.3.2.1 Metadata computation for deviations about Yaw axis

The metadata corresponding to probing poses that deviate only along Yaw axis is computed as follows. First, a transformation matrix $\widehat{M}_{p_s}$ in each metadata band is computed as follows:

$$\widehat{M}_{p_s} = R_{y,p_s,p'}\left(R_{y,p',p'} + I\epsilon\right)^{-1}, \quad (7.6\text{-}1)$$

where, $R_{y,p',p'}$ is the 2x2 covariance matrix of the reference binaural signal $Bin_{p'}$ corresponding to reference pose P', $R_{y,p_s,p'}$ is the 2x2 covariance matrix of the reference binaural signal and the binaural signal corresponding to probing pose P_s .

With the above transformation matrix, a post prediction matrix is computed as

$$R_{\hat{y},p_s,p_s} = \widehat{M}_{p_s} R_{y,p',p'} \widehat{M}^*_{p_s} \ . \tag{7.6-2}$$

The enhanced metadata prediction matrix M_{p_s} is computed by multiplying $\widehat{M}_{p_s}$ with a gain matrix G_{ps} to further match the energy in the post-prediction matrix. The gain matrix G_{ps} is computed as follows.

$$G_{ps} = \begin{bmatrix} g_{l,ps} & 0 \\ 0 & g_{r,ps} \end{bmatrix}, \tag{7.6-3}$$

where

$$g_{l,ps} = sqrt(\frac{R_{y,ps,ps}(1,1)}{R_{\hat{y},ps,ps}(1,1)}) \ , \ g_{r,ps} = sqrt(\frac{R_{y,ps,ps}(2,2)}{R_{\hat{y},ps,ps}(2,2)}). \tag{7.6-4}$$

$R_{y,ps,ps}$ is the 2x2 covariance matrix of the binaural pre-rendition corresponding to probing pose P_s, and $R_{\hat{y},ps,ps}$ is the 2x2 covariance matrix of the reconstructed binaural pre-rendition corresponding to probing pose P_s as given in equation (7.6-2).

$$M_{p_s} = G_{p_s} \widehat{M}_{p_s} \tag{7.6-5}$$

The M_{p_s} matrix is quantized and coded in the metadata bitstream.

The yaw metadata in high frequency range is computed differently when ILD flag (as mentioned in clause 7.6.2.5) is set OR when the quantization strategy is Q2 or Q3 as described in Table 7.6-7. If the ILD flag is set or if the quantization strategy is Q3, the first 4 metadata bands, i.e., frequency range 0 to 1600 Hz, M_{p_s} matrix is computed as shown in equation (7.6-5) to enable the reconstruction of both magnitude and phase of the binaural signal corresponding to probing pose P_s from the reference binaural signal. In the remaining metadata bands, i.e., frequencies above 1600 Hz, a real only 2x2 diagonal matrix D is computed to enable the reconstruction of just magnitude of the binaural signal corresponding to probing pose P_s from the reference binaural signal. The computation of D is as follows:

$$D_{P_s} = \begin{bmatrix} d_l & 0 \\ 0 & d_r \end{bmatrix}, \tag{7.6-6}$$

where

$$d_l = sqrt(\frac{R_{y,ps,ps}(1,1)}{R_{y,p',p'}(1,1)}) \ , \ d_r = sqrt(\frac{R_{y,ps,ps}(2,2)}{R_{y,p',p'}(2,2)}). \tag{7.6-7}$$

If the quantization strategy is Q2, the first 10 metadata bands, i.e., frequency range 0 to 4000 Hz, M_{p_s} matrix is computed as shown in equation (7.6-5). In the remaining metadata bands, i.e., frequencies above 4000 Hz, the real only 2x2 diagonal matrix D is computed as per equation (7.6-7).

M_{p_s} and D matrix is quantized and coded in the metadata bitstream which is then combined with coded reference binaural signal bitstream to form split rendering output bitstream.

7.6.3.2.2 Quantization and coding of Yaw metadata

The high frequency elements of matrix M_{p_s} may be converted to absolute values depending on the quantization strategy as described in clause 7.6.3.3. Then, a fix Q_{P_s} matrix is subtracted from the matrix M_{p_s} (and D_{P_s} when ILD flag is enabled) before quantization.

$$M_{q,p_s} = M_{p_s} - Q_{P_s}, D_{q,p_s} = D_{p_s} - Q_{P_s}, \tag{7.6-8}$$

Where, Q_{P_s} is a 2x2 identity matrix for the bands with real only diagonal matrix D coefficients. For the complex M_{p_s} coefficients, Q_{P_s} is computed as follows.

$$Q_{P_s} = \begin{bmatrix} \cos(d_y) & 0 \\ 0 & \cos(d_y) \end{bmatrix} \quad (7.6\text{-}9)$$

and d_y is the deviation between probing pose P_s and reference pose P' about the Yaw axis.

Each element is uniformly quantized and the quantized index is computed as shown below:

$$Y^{index} = round\left(\frac{((qlvl_Y - 1)\, max(Y_{min}, min(Y_{max}, Val)))}{Y_{max} - Y_{min}}\right), \quad (7.6\text{-}10)$$

where $Y_{min} = -1.4$, $Y_{max} = 1.4$, $qlvl_Y$ is the number of quantization points, Val is the unquantized element of M_{q,p_s} and D_{q,p_s} matrix.

The value of $qlvl_Y$ depends on the quantization strategy as described in clause 7.6.3.3. The quantized index Y^{index} is then coded using either a base2 coding scheme or Huffman coding scheme as described in clause 7.6.3.3. If Huffman coding is used then the quantized indices are first differentially coded using the symmetries in the metadata of various probing poses as given below.

$$M_{index,P_s} = \begin{bmatrix} Y^{index}(0,0) & Y^{index}(0,1) \\ Y^{index}(1,0) & Y^{index}(1,1) \end{bmatrix}$$

$$M_{differential,index,P_s} = \begin{bmatrix} Y^{index}(0,0) & Y^{index}(0,1) \\ Y^{index}(1,0) - sY^{index}(0,1) & Y^{index}(1,1) - sY^{index}(0,0) \end{bmatrix}$$

Where, $s = 1$ for real values and $s = -1$ for imaginary values. For the real only diagonal matrix D indices, $s = -1$.

Then modulo $qlvl_y$ is performed to keep the index range as $\{Y_{min}^{index}, Y_{max}^{index}\}$

$$Y_{TD}^{index} = \begin{cases} Y_{TD}^{index} + qlvl_Y & if\ Y_{TD}^{index} < Y_{min}^{index} \\ Y_{TD}^{index} - qlvl_Y & if\ Y_{TD}^{index} > Y_{max}^{index} \\ Y_{TD}^{index} & otherwise \end{cases} \quad (7.6\text{-}11)$$

where $Y_{min}^{index} = round\left(\frac{((qlvl_Y - 1) * Y_{min})}{Y_{max} - Y_{min}}\right)$, $Y_{max}^{index} = round\left(\frac{((qlvl_Y - 1) * Y_{max})}{Y_{max} - Y_{min}}\right)$ and Y_{TD}^{index} is the differentially coded index.

7.6.3.2.3 Metadata computation for deviations about Pitch axis

Metadata computation for pose deviation along pitch axis is done by using the set of probing poses that deviate from reference pose only by rotation around the pitch axis. The metadata is computed to enable the reconstruction of binaural pre-renditions (BINn) corresponding to the pitch deviation probing poses from the reference binaural signal $Bin_{p'}$ corresponding to reference pose P'. The pitch reconstruction metadata, as shown below, includes a diagonal real 2x2 pitch correction matrix (H) for each time-frequency tile as it is assumed that ITD cues do not change with pitch deviation.

$$H_{P_s} = \begin{bmatrix} h_l & 0 \\ 0 & h_r \end{bmatrix} \quad (7.6\text{-}12)$$

$$h_l = sqrt(\frac{R_{y,ps,ps}(1,1)}{R_{y,p',p'}(1,1)}) \ , \ h_r = sqrt(\frac{R_{y,ps,ps}(2,2)}{R_{y,p',p'}(2,2)}) \quad (7.6\text{-}13)$$

where $R_{y,p',p'}$ is the covariance matrix of reference binaural presentation $Bin_{p'}$, and $R_{y,ps,ps}$ is the covariance matrix of a binaural pre-rendition corresponding to pose Ps.

7.6.3.2.4 Quantization and coding of Pitch metadata

The diagonal elements of H_{P_s} matrix are uniformly quantized with 15 quantization points and the quantization range is limited between the values 0.5 and 1.5. The quantized indices are then either coded using a base2 coding scheme or Huffman coding scheme as described in clause 7.6.3.3.

7.6.3.2.5 Metadata computation for deviations about Roll axis

Metadata computation for pose deviation along roll axis is done by using the set of probing poses that deviate from reference pose only by rotation around the roll axis. In the first 4 metadata bands, i.e., frequency range 0 to 1600 Hz, a complex valued M_{p_s} matrix is computed to enable the reconstruction of both magnitude and phase of the binaural signal corresponding to probing pose P_s from the reference binaural signal. In the remaining metadata bands, i.e., frequencies above 1600 Hz, a real only 2x2 diagonal matrix D_{p_s} is computed to enable the reconstruction of just magnitude of the binaural signal corresponding to probing pose P_s from the reference binaural signal. The computation of M_{p_s} and D_{P_s} is similar to the metadata computation for deviation about Yaw axis when ILD flag is enabled.

7.6.3.2.6 Quantization and coding of Roll metadata

A 2x2 identity matrix I_{P_s} is subtracted from the matrix M_{p_s} and D_{P_s} before quantization.

$$M_{q,p_s} = M_{p_s} - I_{P_s} \quad (7.6\text{-}14)$$

$$D_{q,p_s} = D_{p_s} - I_{P_s} \quad (7.6\text{-}15)$$

Then each element is uniformly quantized with 31 quantization points and the quantization range is limited between the values -1.4 and 1.4. The quantized index is then coded using either a base2 coding scheme or Huffman coding scheme as described in clause 7.6.3.3. If Huffman coding is used, then the quantized indices are first differentially coded using the symmetries in the metadata of various probing poses as described in clause 7.6.3.2.2.

7.6.3.3 Common split rendering metadata quantization and coding strategies

Table 7.6-7: Quantization strategies

1.1.111 Quantization strategies	**1.1.112 Yaw**		**1.1.113 Pitch**		**1.1.114 Roll**	
1.1.115	1.1.116 Number of quantization points	1.1.117 Complex valued MD band threshold (b_{th})	1.1.118 Number of quantization points	1.1.119 Complex valued MD band threshold (b_{th})	1.1.120 Number of quantization points	1.1.121 Complex valued MD band threshold (b_{th})
1.1.122 Q0	1.1.123 63	1.1.124 20	1.1.125 15	1.1.126 0	1.1.127 31	1.1.128 4
1.1.129 Q1	1.1.130 31	1.1.131 20	1.1.132 15	1.1.133 0	1.1.134 31	1.1.135 4
1.1.136 Q2	1.1.137 31	1.1.138 10	1.1.139 15	1.1.140 0	1.1.141 31	1.1.142 4
1.1.143 Q3	1.1.144 31	1.1.145 4	1.1.146 15	1.1.147 0	1.1.148 31	1.1.149 4

The metadata is quantized and coded in a quantization and coding loop which includes iterating over 4 quantization strategies in the order Q0, Q1, Q2, Q3. For each quantization strategy, metadata is quantized with the number of quantization points mentioned in Table 7.6-7. The complex valued MD band threshold (b_{th}) indicates the frequency threshold for complex valued metadata matrix. For pitch axis deviation metadata, the elements are already real valued and hence b_{th} is set to 0.

The quantized metadata indices are then coded with Huffman coder. The indices for Yaw and Roll axis metadata are differentially coded with Huffman coder as per equation (7.6-11). If the total Huffman bits are greater than base2 coded bits than metadata indices are coded with base2 coding. If the coded bits are more than the available metadata bit budget, as given in Table 7.6-8, then the metadata is quantized with next quantization strategy and coded with either Huffman coder or base2 coder depending on whichever coding method results in lesser number of coded bits. The quantization strategy is coded in the bitstream with 2 bits and the coding strategy is coded in the bitstream with 1 bit.

Table 7.6-8: Metadata bit budget

1.1.150 Split rendering bitrate (bps)	1.1.151 Available metadata bitrate (bps)
1.1.152 768000	1.1.153 256000
1.1.154 512000	1.1.155 128000
1.1.156 384000	1.1.157 128000
1.1.158 256000	1.1.159 0 (only 0 DOF mode supported)

In addition to rotation axis metadata, reference head pose, DOF, HQ mode, ILD flag and rotation axes are also coded in the bitstream. To code reference head pose, yaw, pitch and roll Euler angles are rounded to nearest integer and coded with 9 bits each. The DOF parameter can take values 0 to 3 and is coded with 2 bits. The ILD flag is coded with 1 bit. The rotation axes indicate the axes for which the metadata is being coded in the bitstream when DOF is set to 1 or 2. For 1DOF, it can take following values {`DEFAULT_AXIS, YAW, PITCH, ROLL` } and for 2DOF it can take {`DEFAULT_AXIS, YAW_PITCH, YAW_ROLL, PITCH_ROLL`}. The rotation axes are coded with 2 bits for 1DOF and 2DOF modes. The default rotation axis for 1DOF is YAW and the default for 2DOF is `YAW_PITCH`.

The remaining bits are then used by either LCLD codec or LC3plus codec to encode the reference binaural signal.

7.6.3.4 Intermediate split renderer metadata decoder

The split renderer metadata decoder decodes the reference head pose P', DOF, HQ mode, ILD flag, rotation axis and the metadata corresponding to the set of probing poses (Pn) from the bitstream. The metadata decoder decodes coding strategy and quantization strategy from the bitstream. Based on the coding strategy, bits are decoded with either base2 decoder or Huffman decoder to obtain the quantized indices. If Huffman decoder is used then for yaw or roll axis related metadata, differential indices are converted to absolute values as follows.

$$M_{differential,index,P_s} = \begin{bmatrix} Y^{index}(0,0) & Y^{index}(0,1) \\ Y^{index}(1,0) & Y^{index}(1,1) \end{bmatrix} \quad (7.6\text{-}16)$$

$$M_{index,P_s} = \begin{bmatrix} Y^{index}(0,0) & Y^{index}(0,1) \\ Y^{index}(1,0) - sY^{index}(0,1) & Y^{index}(1,1) - sY^{index}(0,0) \end{bmatrix} \quad (7.6\text{-}17)$$

Where, $s = -1$ for real values and $s = 1$ for imaginary values, $M_{differential,index,P_s}$ are the Huffman decoded indices for each time-frequency tile and M_{index,P_s} are the absolute indices. For real only diagonal matrix D indices, $s = 1$.

Then modulo $qlvl_y$ is performed, as described in equation (7.6-11), to keep the index range as $\{Y_{min}^{index}, Y_{max}^{index}\}$.

The quantized indices are then converted to quantized values as shown below.

$$Y^{quantized} = Y^{index}\left(\frac{Y_{max} - Y_{min}}{(qlvl_Y - 1)}\right) \quad (7.6\text{-}18)$$

Where, Y_{max} and Y_{min} are the maximum and minimum values of the quantization range for a given type of metadata parameter, quantization range for metadata parameters along yaw, pitch and roll axis are given in clause 7.6.3.2.2, 7.6.3.2.4 and 7.6.3.2.6, $qlvl_Y$ is the number of quantization points as per the quantization strategy given in Table 7.6-4.

The metadata parameters corresponding to yaw and roll axis are stored in 2x2 matrix form M_{yaw,q,p_s}, M_{roll,q,p_s} for each time-frequency tile wherein the matrix M_{q,p_s} is either real valued or complex valued. The metadata parameters corresponding to pitch axis are stored in real valued 2x2 diagonal matrix form H_{pitch,p_s}.

A fix Q_{P_s} matrix is added to the M_{yaw,q,p_s} as shown below.

$$M_{yaw,p_s} = M_{yaw,q,p_s} + Q_{P_s} \quad (7.6\text{-}19)$$

Where Q_{P_s} is given in clause 7.6.3.2.2.

A 2x2 identity matrix I_{P_s} matrix is added to the M_{roll,q,p_s} as shown below.

$$M_{roll,p_s} = M_{roll,q,p_s} + I_{P_s} \quad (7.6\text{-}20)$$

The metadata matrices M_{yaw,p_s}, M_{roll,p_s} and H_{pitch,p_s} are interpolated and extrapolated as shown in clause 7.6.6.2 to obtain final 2x2 pose correction matrix.

7.6.3.5 Intermediate split renderer metadata loss concealment

In case of packet loss identified by raised BFI flag, the most recent set of successfully decoded split rendering metadata is kept and reapplied for the current (lost) frame.

7.6.4 LCLD coded intermediate split renderer binaural audio format

7.6.4.1 LCLD codec overview

The LCLD codec is designed to be a low complexity and low latency codec to transport the oversampled CLDFB (Complex Low Delay Filter Bank) coefficients between devices. Transporting the oversampled CLDFB coefficients allows for post-processing operations to be split between multiple devices, while ensuring no additional latency is incurred due to the conversion between time and filter-bank domains.

The LCLD codec provides support for 1 and 2 channel content sampled at a rate of 48kHz. The LCLD codec also supports multiple frame rates of 5ms, 10ms, and 20ms.

7.6.4.2 LCLD encoder

7.6.4.2.1 Overview

The structure of the LCLD codec is shown in figure 7.6-1. The input to the LCLD codec is blocks of oversampled CLDFB coefficients, where the number of blocks is dependent on the desired frame rate. For a frame length of 20ms there will be 16 blocks of CLDFB coefficients, for a frame length of 10ms there will be 8 blocks of coefficients, and for a frame length of 5ms there will be 4 blocks of coefficients. For signals with 2 channels the first operation of the encoder is joint channel coding, which is detailed in section 7.6.4.2.3. After the joint channel coding process, the blocks of CDLFB coefficients are partitioned into groups of slots that will share RMS envelope information. The merging of the blocks into groups is detailed in section 7.6.4.2.4. The RMS envelope for each group of slots is computed in approximate critical bands. The RMS envelope is then used to normalize the CLDFB coefficients. The specific banding of the CLDFB coefficients into perceptual bands is provided in section 7.6.4.2.2, while the RMS envelope calculation, envelope coding, and the normalization of the CLDFB coefficient are detailed in section 7.6.4.2.5. The RMS envelope is further used to calculate a backward adaptive perceptual model, which is designed for integer bit-exact calculation, such that the perceptual model can be computed in the decoder and get the same result. The calculation of the bit-exact perceptual model is covered in section 7.6.4.2.6. The output of the perceptual model is then used to calculate the bit allocation, which is detailed in section 7.6.4.2.8. The actual quantization of the normalized CLDFB coefficients can be achieved in two possible ways; direct scalar quantization followed by Huffman coding or scalar quantization of prediction residual (DPCM) followed by Huffman coding. The DPCM based quantization uses a forward adaptive complex first order predictive structure. The calculation and coding of the prediction coefficients for the DPCM based quantization are covered in section 7.6.4.2.7, while the details of the quantization procedure are provided in section 7.6.4.2.9.

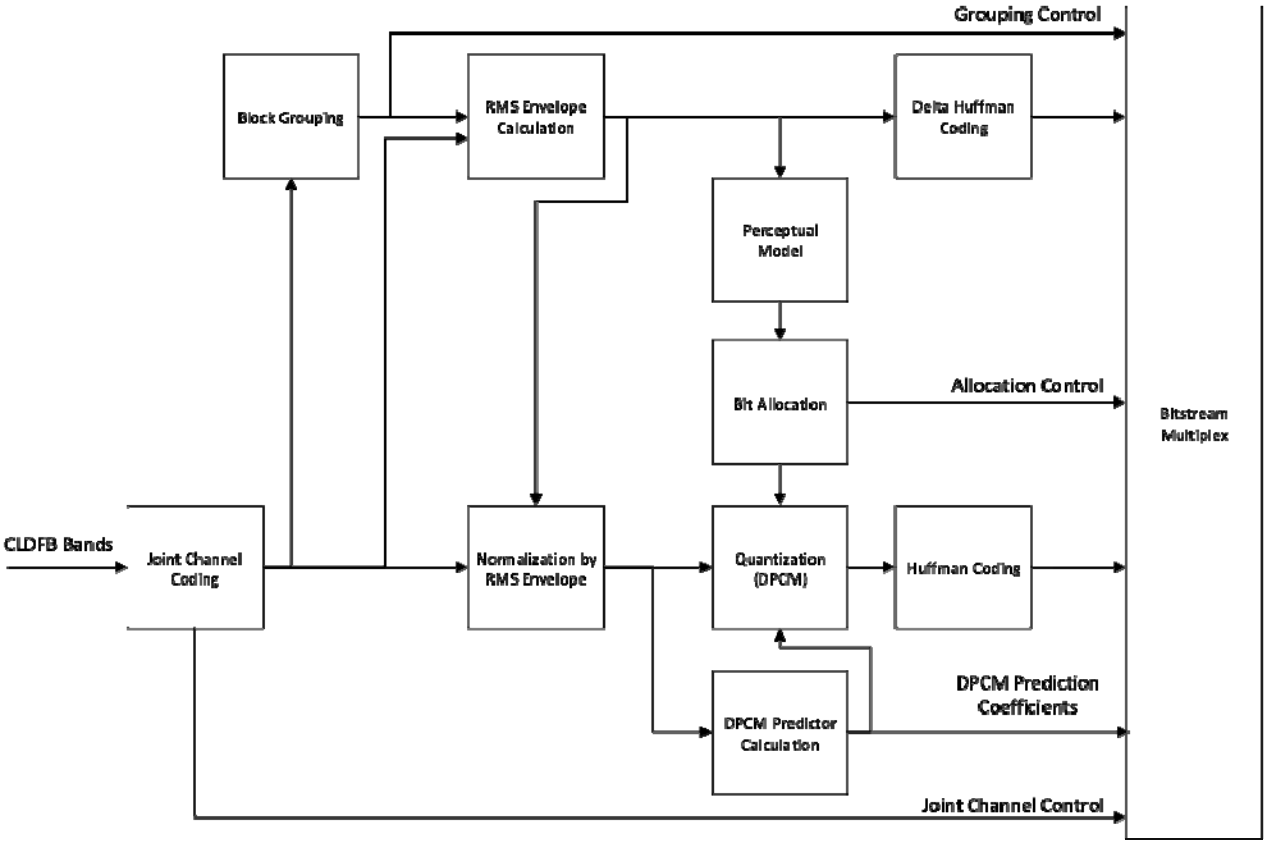


Figure 7.6-1. Schematic of the LCLD Encoder Process

7.6.4.2.2 Perceptual Banding

While the LCLD codec transports 60 CLDFB bands many of the encoder and decoder operations are performed on a courser banding structure that approximates critical bands. The number of CLDFB bands contained in the coarser perceptual bands are provided in table 7.6-9. Where b_m is the number of CLDFB bands in a perceptual band, B_m^{Low} is the CLDFB start band, B_m^{High} is the CDLFB stop band, F_m^{Low} is the start frequency of the band in Hz, and F_m^{High} is the stop frequency of the band in Hz.

Table 7.6-9: Perceptual Banding Structure

1.1.160 Band (*m*)	1.1.161 b_m	1.1.162 B_m^{Low}	1.1.163 B_m^{High}	1.1.164 F_m^{Low} (Hz)	1.1.165 F_m^{High} (Hz)
1.1.166 0	1.1.167 1	1.1.168 0	1.1.169 1	1.1.170 0	1.1.171 400
1.1.172 1	1.1.173 1	1.1.174 1	1.1.175 2	1.1.176 400	1.1.177 800
1.1.178 2	1.1.179 1	1.1.180 2	1.1.181 3	1.1.182 800	1.1.183 1200
1.1.184 3	1.1.185 1	1.1.186 3	1.1.187 4	1.1.188 1200	1.1.189 1600
1.1.190 4	1.1.191 1	1.1.192 4	1.1.193 5	1.1.194 1600	1.1.195 2000
1.1.196 5	1.1.197 1	1.1.198 5	1.1.199 6	1.1.200 2000	1.1.201 2400
1.1.202 6	1.1.203 1	1.1.204 6	1.1.205 7	1.1.206 2400	1.1.207 2800
1.1.208 7	1.1.209 1	1.1.210 7	1.1.211 8	1.1.212 2800	1.1.213 3200
1.1.214 8	1.1.215 1	1.1.216 8	1.1.217 9	1.1.218 3200	1.1.219 3600
1.1.220 9	1.1.221 1	1.1.222 9	1.1.223 10	1.1.224 3600	1.1.225 4000
1.1.226 10	1.1.227 1	1.1.228 10	1.1.229 11	1.1.230 4000	1.1.231 4400
1.1.232 11	1.1.233 2	1.1.234 11	1.1.235 13	1.1.236 4400	1.1.237 5200
1.1.238 12	1.1.239 2	1.1.240 13	1.1.241 15	1.1.242 5200	1.1.243 6000
1.1.244 13	1.1.245 2	1.1.246 15	1.1.247 17	1.1.248 6000	1.1.249 6800
1.1.250 14	1.1.251 2	1.1.252 17	1.1.253 19	1.1.254 6800	1.1.255 7600
1.1.256 15	1.1.257 2	1.1.258 19	1.1.259 21	1.1.260 7600	1.1.261 8400
1.1.262 16	1.1.263 3	1.1.264 21	1.1.265 24	1.1.266 8400	1.1.267 9600
1.1.268 17	1.1.269 3	1.1.270 24	1.1.271 27	1.1.272 9600	1.1.273 10800
1.1.274 18	1.1.275 4	1.1.276 27	1.1.277 31	1.1.278 10800	1.1.279 12400
1.1.280 19	1.1.281 6	1.1.282 31	1.1.283 37	1.1.284 12400	1.1.285 14800
1.1.286 20	1.1.287 6	1.1.288 37	1.1.289 43	1.1.290 14800	1.1.291 17200
1.1.292 21	1.1.293 7	1.1.294 43	1.1.295 50	1.1.296 17200	1.1.297 20000
1.1.298 22	1.1.299 10	1.1.300 50	1.1.301 60	1.1.302 20000	1.1.303 24000

7.6.4.2.3 Joint Channel Coding

7.6.4.2.3.1 Overview

The Joint Channel Coding module consists of 3 sub-modules:

- Phase Alignment
- Mid/Side Transform
- Real-valued Side Signal Prediction

The usage of each of the sub modules can be signalled efficiently and the associated parameters are efficiently entropy coded. For example, for an amplitude panned stereo signal, Side signal prediction parameters may be used and transmitted to the decoder but no phase alignment parameters. The parameter frequency resolution corresponds to the perceptual frequency bands as shown in table Perceptual Banding Structure and the time resolution corresponds to the frame length. A carefully designed detector decides on the best coding mode in each frame considering the side rate and

perceptual benefits using joint channel coding based on the level-dependent perceptual model as described in section 7.6.4.2.6 Perceptual Model.

The encoder joint channel coding can be described in matrix notation as:

$$\begin{bmatrix} X_{b,n}^{M} \\ X_{b,n}^{S} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -p_k^Q & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{j\varphi_k^Q} \end{bmatrix} \begin{bmatrix} X_{b,n}^{L} \\ X_{b,n}^{R} \end{bmatrix},$$

where $X_{b,n}^{L}$ and $X_{b,n}^{R}$ are the left and right complex input samples in LCLD band b, k is the perceptual band to which the LCLD band b belongs, n is the time sample index within the coded frame, and φ_k^Q is the quantized phase alignment parameter and p_k^Q the quantized prediction parameter.

7.6.4.2.3.2 Bitstream Syntax

The transmission of joint coding parameters depends on the bitstream element *MSMode* and 2 additional flags *AnyNonZeroPhase* and *AnyNonZeroPred*. Based on the values of these bitstream elements different stereo coding types can be identified as shown in the following table:

Table 7.6-10: Joint Stereo coding types

1.1.304	1.1.305 Description	1.1.306 MS Mode	1.1.307 AnyNonZero Phase	1.1.308 AnyNonZeroPred
1.1.309	1.1.310 No joint coding	1.1.311 0	1.1.312 NA	1.1.313 NA
1.1.314	1.1.315 Basic Mid/Side	1.1.316 1 or 2	1.1.317 NA	1.1.318 NA
1.1.319	1.1.320 Mid/Side with phase alignment only	1.1.321 3 1.1.322	1.1.323 1	1.1.324 0
1.1.325	1.1.326 Mid/Side with side prediction only	1.1.327 3 1.1.328	1.1.329 0	1.1.330 1
1.1.331	1.1.332 Mid/Side with phase alignment and side prediction	1.1.333 3 1.1.334	1.1.335 1	1.1.336 1

7.6.4.2.3.3 Parameter Computation and Quantization

The phase alignment and prediction parameters are computed based on signal energies and (complex-valued) left/right covariance. Phase alignment is only considered if the left/right cross-correlation coefficient exceeds a threshold of $\sqrt{0.5}$, otherwise the phase parameter it is set to 0.

The left and right signal energies are computed as

$$E_k^L = \sum_{n=0}^{N_{Frame}-1} \sum_{b \in k} \left| X_{b,n}^L \right|^2 ,$$

$$E_k^R = \sum_{n=0}^{N_{Frame}-1} \sum_{b \in k} \left| X_{b,n}^R \right|^2 ,$$

Where $X_{b,n}^L$ and $X_{b,n}^R$is the complex left and right input sample respectively and time index n, b is the LCLD band belonging to the perceptual band k and N_{Frame} is the number of samples per frame and LCLD band. The left/right covariance is computed as

$$C_k^{LR} = \sum_{n=0}^{N_{Frame}-1} \sum_{b \in k} X_{b,n}^L \, X_{b,n}^{*R} ,$$

where the * indicates complex conjugate. The inter-channel phase difference is computed from the covariance as

$$\varphi_k = \arctan 2 \left(\frac{real(C_k^{LR})}{imag(C_k^{LR})} \right) \text{ if } |C_k^{LR}|^2 > 0.5 \cdot E_k^L E_k^R .$$

Otherwise $\varphi_k = 0$.

The phase quantization index is given as $\varphi_k^{idx} = round\left(\varphi_k \cdot \frac{12}{\pi} \right)$. If φ_k^{idx}equals 12 then $\varphi_k^{idx}$is set to 0.

And the quantized phase is

$$\varphi_k^Q = \varphi_k^{idx} \cdot \frac{\pi}{12} .$$

In the case of left/right phase alignment using the quantized phase, the modified covariance is given as

$$C_{k,\varphi}^{LR} = C_k^{LR} \cdot e^{j\varphi_k^{Q*}} .$$

The mid and side signal energies are computed from original or phase-modified covariance as

$$E_{k,\varphi}^M = 0.25 \cdot \left(E_k^L + E_k^R + 2 \cdot real\left(C_{k,\varphi}^{LR} \right) \right),$$

$$E_{k,\varphi}^S = 0.25 \cdot \left(E_k^L + E_k^R - 2 \cdot real\left(C_{k,\varphi}^{LR} \right) \right).$$

The real-valued side prediction parameter which minimizes the side signal energy is computed as

$$p_k = 0.25 \cdot \frac{\left(E_k^L - E_k^R \right)}{E_{k,\varphi}^M} .$$

The quantized prediction coefficient is computed as

$$p_k^Q = \frac{\max\{\min\{round\,(p_k \cdot 12), 12\}, -12\}}{12} .$$

Finally, the side signal (residual) energy, dependent on application of phase alignment and/or side prediction is computed as

$$E_{k,\varphi,p}^S = E_{k,\varphi}^S + \left(p_k^Q \right)^2 E_{k,\varphi}^M - 0.5 \cdot p_k^Q \cdot \left(E_k^L - E_k^R \right) .$$

The mid and side signal energies can also be expressed in terms of the stereo coding type t as shown in table Joint Stereo Coding Types and below where k is the perceptual band index:

1.1.337 Coding Type t	1.1.338 1	1.1.339 2	1.1.340 3	1.1.341 4
1.1.342 Parameters	1.1.343 $\varphi_k^Q = 0,\ p_k^Q = 0$	1.1.344 $\varphi_k^Q,\ p_k^Q = 0$	1.1.345 $\varphi_k^Q = 0,\ p_k^Q$	1.1.346 $\varphi_k^Q,\ p_k^Q$

7.6.4.2.3.4 Joint Coding Type Decision

For each of the 4 joint stereo coding types (1,2,3,4) listed in above table, a coding gain compared to independent left/right coding (type 0) is estimated based on the level-dependent perceptual model. For each joint coding type the inter-channel level difference of the jointly coded perceptual band is compared to the inter-channel level difference of the independently coded band. If the magnitude of the inter-channel level difference of the jointly coded band exceeds this of the independently (left/right) coded band, then the band is flagged as candidate for the joint coding type in an array called MSFlags. The coding gain in units of bits for the band is calculated as follows:

$$BitsGain_{k,t} = S_k \cdot \max\{D_{k,t} - D_{k,0}, 0\}\, 2 \cdot B_{Width,k} \cdot N_{frame} \cdot F_{Frame}\ ,$$

where k refers to the perceptual band as given in Table 7.6-9 (Perceptual Banding Structure), S_k is the perceptual model slope parameter as given in Table 7.6-11 (Perceptual Model SMR Slope) for a given band, $B_{Width,k}$ is the number of complex-valued LCLD bands belonging to the perceptual band, N_{frame} is the number of complex-valued samples per frame and F_{frame} is an estimate of the number of Bits needed for an SNR increase by 1dB. The estimate depends on the frame length as follows:

$$F_{frame} = \frac{0.1}{\log 10(2)} \cdot 0.7 + \frac{N_{frame}-4}{16-4} \cdot (1.0 - 0.7)\ .$$

Table 7.6-11: Perceptual Model SMR Slope Parameter S dependent on the perceptual Band

1.1.347	1.1.348	1.1.349	1.1.350	1.1.351	1.1.352	1.1.353	1.1.35	1.1.35	1.1.35	1.1.357	1.1.358	1.1.359
1.1.360	1.1.361	1.1.362	1.1.363	1.1.364	1.1.365	1.1.366	1.1.36	1.1.36	1.1.36	1.1.370	1.1.371	1.1.372
1.1.373	1.1.374	1.1.375	1.1.376	1.1.377	1.1.378	1.1.379	1.1.38	1.1.38	1.1.38	1.1.383	1.1.384	1.1.385
1.1.386	1.1.387	1.1.388	1.1.389	1.1.390	1.1.391	1.1.392	1.1.39	1.1.39	1.1.39	1.1.396	1.1.397	

The inter-channel level differences D are computed as follows:

$$D_{k,t} = \left| 10 \cdot \log_{10}\left(\frac{E_{k,t}^M}{E_{k,t}^S}\right) \right|,\ t = 1,2,3,4\ ,$$

$$D_{k,0} = \left| 10 \cdot \log_{10}\left(\frac{E_k^L}{E_k^R}\right) \right| .$$

The joint stereo coding candidate bands are given for each stereo coding type t for all k as

$$MSFlags_{k,t} = 1,\ if\ D_{k,t} > D_{k,0}\ ,$$

$$MSFlags_{k,t} = 0,\ if\ D_{k,t} \leq D_{k,0}\ ,$$

where $E_{k,t}^M$ is the energy of Mid signal in band k for joint coding type t, $E_{k,t}^{S'}$ is the energy of the Side (residual) signal in band k for coding type t, and E_k^L and E_k^R are the energies of the left and right signals respectively.

For each of the joint coding types, a total estimated Bit Saving compared to independent coding is computed as:

$$BitsTotalGain_t = \max\left\{-BitsCost_t + \sum_{k=0}^{K-1} BitsGain_{k,t}\ , 0\right\} ,$$

where $BitsCost_t$ is the number of signalling and parameter bits needed for a joint coding type t and K is the total number of coded perceptual bands. The joint coding type with the highest estimated bit saving is identified as t_{Best}.

The final joint coding type decision is as follows in pseudo code:

If (t_{Best}== 2 OR t_{Best} == 3 OR t_{Best}== 4) AND $BitsTotalGain_{t_{Best}} > 1.1 BitsTotalGain_1$

$$t_{use} = t_{Best}$$

Else if ($BitsTotalGain_1 > 0$)

$$t_{use} = 1$$

Else

$$t_{use} = 0$$

7.6.4.2.3.5 Entropy Coding of Parameters

Phase alignment parameters are entropy coded by computing second order frequency differentials from the phase quantization indices with phase wrapping and Huffman coding using the same code book which is used to code the RMS envelopes. Specifically, the second order differences are computed by first computing the index differences over the perceptual bands and then computing the differences over the first differences. All index differences are wrapped into the quantization range of [-12,12] which exactly corresponds to $[-\pi, +\pi]$, This means that the value for the first frequency band is unmodified, the value for the second frequency band corresponds to a difference and the values in the third frequency band and all other bands correspond to a difference of the first difference. This scheme is efficient for stereo signals with constant inter-channel phase shift and with inter-channel time delay which can be observed for example with HRTF processed signals with a dominant sound source.

Side prediction parameters are entropy coded by quantization and taking differences of the quantization indices over perceptual bands and Huffman coding, again using the same code book as is used for the RMS envelopes. Given the quantization ranges of the joint stereo coding parameters and that of the RMS envelopes index differences never exceed the RMS envelope quantization table range.

7.6.4.2.4 Temporal Grouping

A LCLD frame either contains 4,8, or 16 temporal slots of the CLDFB coefficients depending on the specific frame length in use. To reduce side-information cost of transmitting the RMS Envelope, the individual temporal slots of the CLDFB coefficients can be grouped together into groups, where a group is defined to be 1 or more temporal slots of CLDFB coefficients. The LCLD codec transmits a new set of RMS envelope data for each group in a frame. Every frame starts with a new group (i.e. groups cannot cross frame boundaries).

Grouping control is transmitted to the decoder as a set of single-bit flags for each slot in the frame except the first slot (the first slot is guaranteed to be the start of a new group). A ‘1’ is transmitted to indicate the start of new group, while a ‘0’ is transmitted to indicate the continuation of a group. For example, the group information for a 16-slot frame will contain 15 single-bit flags as the first slot is known to be the start of new group.

Furthermore, for 2-channel content, the encoder can either send grouping information for each individual channel or transmit a single set of grouping information that is applicable to both channels. When 2 channels are present, a single-bit flag is transmitted prior to the grouping information to indicate if the grouping is common for both channels. A '1' means grouping is common and there will only be 1 set of grouping information shared across both channels, while a '0' means the grouping is not common for each channel and 2 sets of grouping information will exist.

The grouping decision is computed using an iterative greedy-merge algorithm. The greedy-merge algorithm attempts to trade-off the cost of transmitting the RMS envelope versus the accuracy of the merged RMS envelope for each individual slot. The greedy-merge algorithm works as follows:

1. Initialize groups to have at least 1 temporal slot each group. The groups can be enumerated g = 0, 1, ... G-1, where G is the total number of groups.
2. For each group compute a cost function associated with merging the group to the next adjacent group. Starting from group 0, compute the merge cost to merge group 0 and group 1. Then compute the cost of merging group 1 and group 2, proceed until the merge cost of group G-2 and G-1 has been evaluated.
3. After all possible group merges have been evaluated, find the minimum merge cost computed in step 2. If the merge cost is below a threshold merge the two groups associated with minimum merge cost, reduce the total number of groups (G = G -1), and return to step 2. If the minimum merge cost is greater than a threshold, the greedy-merge algorithm terminates. If the number of groups is 1 the algorithm terminates.

Figure 7.6-2 shows the flow diagram for the greedy-merge algorithm.

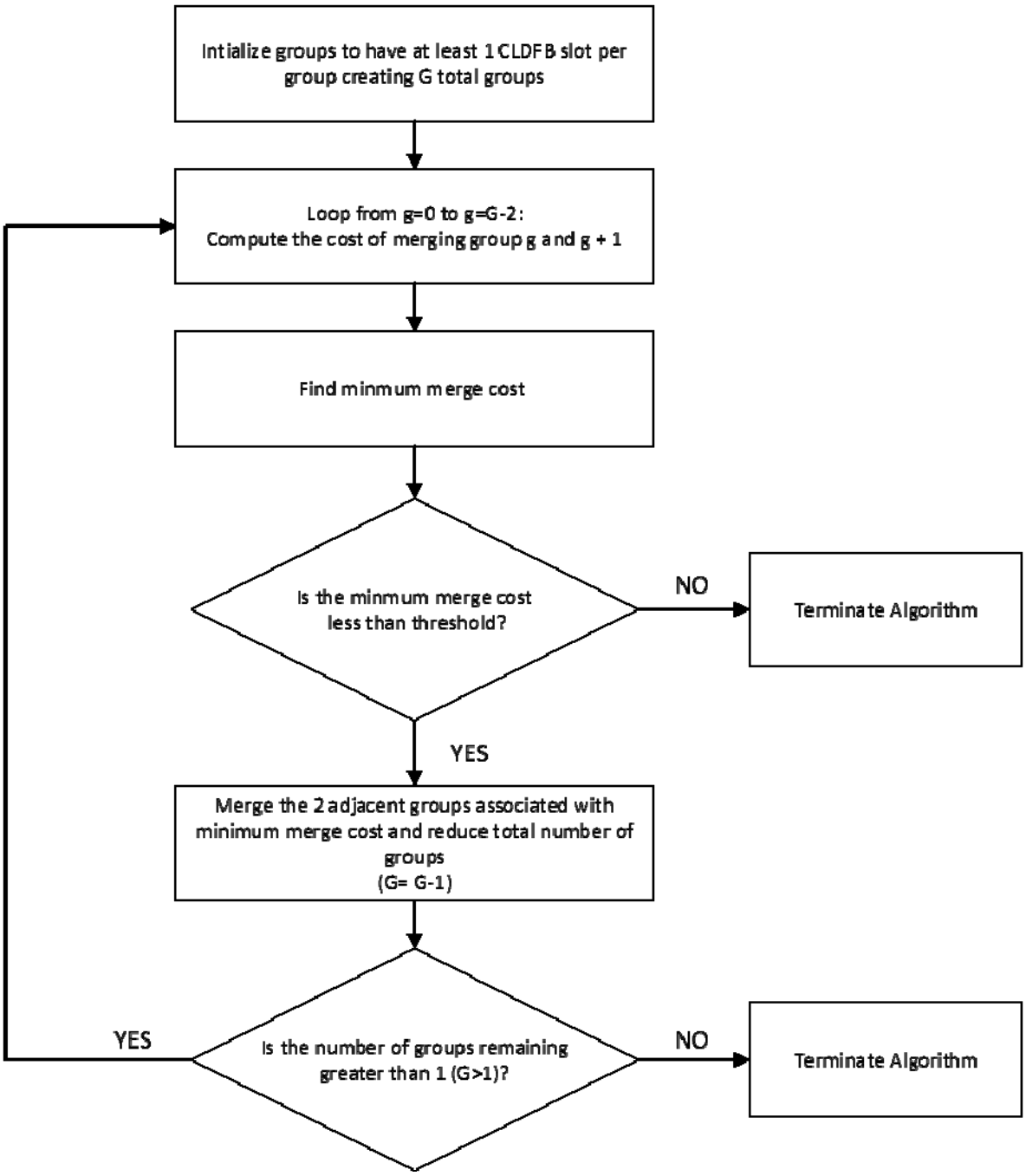


Figure 7.6-2. Flow diagram of the greedy-merge algorithm

The merge cost used in the greedy-merge algorithm is computed using the following steps:

Step 1: Compute the weighted cost of the first group in the possible merge as shown in equation 7.6-21:

$$C_1 = \sum_{k=N_1}^{N_2-1} \sum_{m=0}^{M-1} 2 b_m w_m |R_m^1 - E_{nm}^{dB}| \ , \qquad (7.6\text{-}21)$$

where:

C_1 is the weighted cost associated with the first group in a possible merge,

n is the temporal slot index,

N_1 is the temporal slot index of the first group in the possible merge,

N_2 is the temporal slot index of the second group in the possible merge,

m is the perceptual sub-band index,

M is the total number of perceptual sub-bands,

b_m is the number of CLDFB coefficients in the m^{th} perceptual band,

w_m is the weight for the m^{th} perceptual band (the weight values are provided in table 7.6-9,

E_{nm}^{dB} is the energy in dB for the m^{th} perceptual band for the n^{th} slot, and

R_m^1 is the estimated RMS envelope for the first group in the possible merge in dB

The energy in dB in each perceptual band (S_{nk})for the n^{th} temporal slot can computed using equations 7.6-22 and 7.6-23.

$$E_{nm} = \frac{1}{b_m}\sum_{k=B_m^{Low}}^{B_m^{High}-1}\left|S_{nk}^{CLDFB}\right|^2 , \quad (7.6\text{-}22)$$

$$E_{nm}^{dB} = 10log_{10}(E_{nm}) , \quad (7.6.23)$$

where:

S_{nk}^{CLDFB} is the CLDFB domain signal,

n is the temporal slot index,

k is the CLDFB band index,

B_m^{Low} is the CLDFB band index at the start of the m^{th} perceptual band,

B_m^{High} is the CLDFB band index at the end of the m^{th} perceptual band, and

X_{nb} is the complex CLDFB coefficient for the b^{th} CLDFB band and n^{th} temporal block in the frame.

The estimated RMS envelope values in dB for the first group in the possible merge is given by equation 7.6-24:

$$R_m^1 = 3.01 * ROUND\left(log_2\left(\frac{1}{N_2-N_1}\sum_{n=N_1}^{N_2-1}E_{nm}\right)\right). \quad (7.6\text{-}24)$$

Step 2: Compute the weighted cost of the second group in the possible merge as shown in equation 7.6-25:

$$C_2 = \sum_{n=N_2}^{N_3-1}\sum_{m=0}^{M-1}2b_m w_m|R_m^2 - E_{nm}^{dB}| , \quad (7.6\text{-}25)$$

where:

C_2 is the weighted cost associated with the second group in a possible merge, and

N_3 is the temporal slot index of the end of the second group in the possible merge.

The estimated RMS envelope values in dB for the second group in the possible merge is given by equation 7.6-26.

$$R_m^2 = 3.01 * ROUND\left(log_2\left(\frac{1}{N_3-N_2}\sum_{n=N_2}^{N_3-1}E_{nm}\right)\right). \quad (7.6\text{-}26)$$

Step 3: Compute the weighted cost of the group resulting from merging the group 1 and group 2 as shown in equation 7.6-27:

$$C_M = \sum_{n=N_1}^{N_3-1}\sum_{m=0}^{M-1}2b_m w_m|R_m^M - E_{nm}^{dB}| , \quad (7.6\text{-}27)$$

where:

C_M is the weighted cost associated with the merge of groups 1 and 2, and

R_m^M is the estimated RMS envelope in dB for the merge of groups 1 and 2.

The estimated RMS envelope values in dB for the merge of group 1 and group 2 is given by equation 7.6-28:

$$R_m^M = 3.01 * ROUND\left(log_2\left(\frac{1}{N_3 - N_1}\sum_{n=N_1}^{N_3-1} E_{nm}\right)\right). \quad (7.6\text{-}28)$$

Step 4: Compute the total merge cost for the 2 groups in the possible merge as shown in equation 7.6-29:

$$T_g = \frac{C_M - (C_1 + C_2)}{6.02} - (B_1 + B_2 - B_m), \quad (7.6.\text{-}29)$$

where:

T_g is the merge cost for merging the group g and $g + 1$,

C_M is the weighted cost associated with merging group 1 and group 2,

B_1 is the estimated bit-cost of transmitting the RMS envelope for group 1,

B_2 is the estimated bit-cost of transmitting the RMS envelope for group 2, and

B_m is the estimated bit-cost of transmitting the RMS envelope for the merged group.

The estimated bit-cost (B_1, B_2, and B_m) is based on the delta coding of the RMS envelope, which is detailed in section 7.6.4.2.5.

For the case of 2-channel content when common grouping is used across both channels the cost functions are modified in each step of the greedy-merge algorithm to include both channels in the calculations.

The greedy-merge algorithm terminates under 2 conditions; if the number of groups (G) is 1, meaning all blocks in a frame have merged into a single group, or the minimum merge cost function is greater than a threshold. The optimal threshold used for the cost function in equation 7.6-29 is zero. The result of the greedy-merge grouping algorithm is the number of groups G and a set of group lengths (number of temporal slots in a group) L_g for each of the G groups.

Table 7.6-12: Weight values for computing the weighted group costs.

1.1.398 Perceptual Band (m)	1.1.399 Weight (w_m)	1.1.400 Perceptual Band (m)	1.1.401 Weight (w_m)
1.1.402 0	1.1.403 0.5625	1.1.404 12	1.1.405 0.75
1.1.406 1	1.1.407 0.5625	1.1.408 13	1.1.409 0.75
1.1.410 2	1.1.411 0.625	1.1.412 14	1.1.413 0.75
1.1.414 3	1.1.415 0.6875	1.1.416 15	1.1.417 0.75
1.1.418 4	1.1.419 0.6875	1.1.420 16	1.1.421 0.75
1.1.422 5	1.1.423 0.75	1.1.424 17	1.1.425 0.75
1.1.426 6	1.1.427 0.75	1.1.428 18	1.1.429 0.75
1.1.430 7	1.1.431 0.75	1.1.432 19	1.1.433 0.75
1.1.434 8	1.1.435 0.75	1.1.436 20	1.1.437 0.75
1.1.438 9	1.1.439 0.75	1.1.440 21	1.1.441 0.75
1.1.442 10	1.1.443 0.75	1.1.444 22	1.1.445 0.75
1.1.446 11	1.1.447 0.75	1.1.448	1.1.449

7.6.4.2.5 RMS Envelope

7.6.4.2.5.1 RMS Envelope Calculation

The RMS envelope values are computed for each group in a frame. The output of the grouping algorithm (see section 7.6.4.2.4) is the number of groups G and the length or number L_g. We can also define the temporal block index of the start of a group (N_g) as the sum of the group lengths of the prior groups as shown in equation 7.6-30. Group 0 starts at 0th temporal block index (i.e., $N_0 = 0$).

$$N_g = \sum_{n=0}^{g-1} L_n \ . \tag{7.6-30}$$

The RMS envelope for each group and each perceptual band can then be calculated as shown in equation 7.6-31.

$$R_{gm} = ROUND\left(log_2\left(\frac{1}{b_m L_g}\sum_{n=N_g}^{N_g+L_g-1}\sum_{m=B_m^{Low}}^{B_m^{High}-1}\left|S_{nk}^{CLDFB}\right|^2\right)\right), \tag{7.6-31}$$

where:

R_{gm} is the RMS envelope for the g^{th} group and the m^{th} perceptual band index,

S_{nk}^{CLDFB} is the complex CLDFB coefficient for the k^{th} CLDFB band and n^{th} temporal slot in the frame,

b_m is the number of CLDFB coefficients in the m^{th} perceptual band,

B_m^{Low} is the CLDFB band index at the start of the m^{th} perceptual band, and

B_m^{High} is the CLDFB band index at the end of the m^{th} perceptual band.

The RMS envelope values are limited to the range -64 to 63 as shown in equation 7.6-32:

$$R_{gm} = \begin{cases} -64 & R_{gm} \le -64 \\ R_{gm} & -64 < R_{gm} < 63 \\ 63 & R_{gm} \ge 63 \end{cases} \ . \tag{7.6-32}$$

7.6.4.2.5.2 Normalizing the CLDFB Coefficients with the RMS Envelope

The raw CLDFB coefficients are normalized by the RMS envelope in each group and each perceptual band prior to the DPCM and quantization stage discussed in section 7.6.4.2.9. The normalization of the CLDFB coefficients is shown in equation 7.6-33:

$$\tilde{S}_{nk}^{CLDFB} = S_{nk}^{CLDFB} 2^{-0.5 R_{gm}} \ , \tag{7.6-33}$$

where:

$\tilde{S}_{nk}^{CLDFB}$ is the normalized CLDFB coefficient at the n^{th} temporal block index and k^{th} CLDFB band index,

S_{nk}^{CLDFB} is the original CLDFB coefficient at the n^{th} temporal block index and k^{th} CLDFB band index,

n is the temporal slot index that belongs in g^{th} group, $n \in \{N_g, \cdots, N_g + L_g - 1\}$, and

k is the CLDFB band index that belongs in the m^{th} perceptual band, $k \in \{B_m^{Low}, \cdots, B_m^{High} - 1\}$.

7.6.4.2.5.3 RMS Envelope Transmission

The RMS envelope values in each group and each perceptual band excluding the 0th band are transmitted as the Huffman coded difference to the preceding perceptual bands RMS envelope value. The differential RMS envelope values are calculated and limited to the range -32 to 31 as shown in equation 7.6-34.

$$\Delta R_{gm} = \begin{cases} -32 & \left(R_{gm} - R_{g(m-1)}\right) \le -32 \\ \left(R_{gm} - R_{g(m-1)}\right) & -32 < \left(R_{gm} - R_{g(m-1)}\right) < 31 \\ 31 & \left(R_{gm} - R_{g(m-1)}\right) \ge 31 \end{cases} \quad \forall m \ge 1 \ . \tag{7.6-34}$$

The RMS envelope values for the 0th perceptual band for all groups in a frame are transmitted as 7-bit unsigned integer by adding 64 to the value ($R_{g0} + 64$).

The differential RMS envelope for all bands above the 0th band are coded using a fixed Huffman codebook.

7.6.4.2.6 Perceptual Model

The perceptual model is designed to operate in both the encoder and decoder based only on transmitted information. Specifically, only the RMS envelope (see section 7.6.4.2.5) is used to compute the target SNR requirements, for each channel, for each perceptual band, and each group segment of a frame. As the target SNR calculated by the perceptual model in both the encoder and decoder must match exactly, all arithmetic calculations described here are integer calculations with a maximum precision of 24bits.

The perceptual model computes the target SNR for just noticeable distortion JND (the signal to mask ratio) for each group in a frame and for each perceptual band. As the RMS envelope is transmitted to the decoder in a log space the perceptual model is also computed in a log space. The following steps are used to compute the signal to mask ratio (SMR_{gm}) for each group and perceptual band:

Step 1: Scale the RMS envelope and add the log domain width of the perceptual band as shown in equation 7.6-35:

$$E_{gm} = 64 \cdot R_{gm} + 64 \cdot \lfloor log_2(b_m) \rfloor , \quad (7.6\text{-}35)$$

where:

E_{gm} is the scaled energy for the g^{th} group and m^{th} perceptual band,

R_{gm} is the RMS envelope for the g^{th} group and m^{th} perceptual band,

$64 \cdot \lfloor log_2(b_m) \rfloor$ is the scaled log domain bandwidth of the m^{th} perceptual band, and

$\lfloor \cdot \rfloor$ is a floor operation.

The scaled log domain bandwidth ($64 \cdot \lfloor log_2(b_m) \rfloor$) values are constant so can be pre-calculated as shown in Table 7.6-13. In Table 7.6-13, $64 \cdot \lfloor log_2(b_m) \rfloor$ is expressed as BWA_m.

Step 2: Compute the sensation level for each perceptual band as the energy level of the band relative to the absolute threshold of hearing as shown in equation 7.6-36:

$$SV_{gm} = E_{gm} - TQ_m , \quad (7.6\text{-}36)$$

where:

SV_{gm} is the sensation level for the g^{th} group in a frame and m^{th} perceptual band, and

TQ_m is the hearing threshold in quiet. The values for TQ_m are provided in Table 7.6-14.

Step 3: Compute the mask offset due to the sensation level as shown in equation 7.6-37:

$$SV_m^{Offset} = \left\lfloor \frac{D_m SV_m}{256} \right\rfloor , \quad (7.6\text{-}37)$$

where:

SV_m^{Offset} is the mask offset due to sensation level, and

D_m is the sensation level offset gradient for each perceptual band. The values for D_m are provided in Table 7.6-15.

Step 4: Offset the energy in each perceptual band by the sensation level offset as shown in equation 7.6-38:

$$E_{gm}^{Offset} = E_{gm} - SV_m^{Offset} , \quad (7.6\text{-}38)$$

where:

E_{gm}^{Offset} is the energy for the g^{th} group and m^{th} perceptual band offset by the sensation level offset.

Step 5: Convert the sensation offset energy in each perceptual band to the loudness domain as shown in equation 7.6-39:

$$E_{gm}^{Loudnes} = \left\lfloor \frac{614 \cdot E_{gm}^{Offset}}{2048} \right\rfloor . \quad (7.6\text{-}39)$$

Step 6: Spread the loudness energy in each perceptual band with a spreading function as shown in equation 7.6-40:

$$M_{gm}^{Loudness} = LogAdd\left(E_{gm}^{Loudnes}, E_{gl}^{Loudnes} + SF_{ml}\right) \ \forall l = 0,1, \ldots M-1 , \quad (7.6\text{-}40)$$

where:

$M_{gm}^{Loudness}$ is the loudness domain masking level,

SF_{ml} is the spreading function applied to the m^{th} perceptual band due to the energy in the l^{th} perceptual band. The values for SF_{ml} are provided in table 7.6-16.

$LogAdd(a,b)$ is a table look-up approximation of a multiply and add in the linear domain as shown in equation 7.6-41:

$$LogAdd(a,b) = \begin{cases} a + L[a-b] & a > b \\ b + L[b-a] & b \geq a \end{cases} , \quad (7.6\text{-}41)$$

where:

$L[a]$ is a table look up, the values of $L[a]$ are provided in Table 7.6-17.

Step 7: Convert mask back from the loudness domain as shown in equation 7.6-42:

$$M_{gm} = \left\lfloor \frac{6827 \cdot M_{gm}^{Loudness}}{2048} \right\rfloor . \quad (7.6\text{-}42)$$

Step 8: Compute the signal to mask ratio (target SNR at JND) for the g^{th} group and m^{th} perceptual band as shown in equation 7.6-43:

$$SMR_{gm} = E_{gm} - M_{gm} . \quad (7.6\text{-}43)$$

The above steps are for mono or independent channel coding, however, if 2 channels are present in the signal and joint channel coding is in use, then there is an additional step after step 7. The additional step sets the masking level in both channels to be the greater of the two masking levels as shown in equation 7.6-44. This additional step is only used if there are 2 channels present, the m^{th} band is jointly coded, and the band m is greater than 0.

$$M_{gm}^{1} = \begin{cases} M_{gm}^{1} & M_{gm}^{1} \geq M_{gm}^{2} \\ M_{gm}^{2} & M_{gm}^{2} > M_{gm}^{1} \end{cases} , \quad (7.6\text{-}44)$$

$$M_{gm}^{2} = M_{gm}^{1} .$$

Table 7.6-13: Bandwidth adjust values (BWA_m) for each perceptual band

Band (*m*)	BWA_m	Band (*m*)	BWA_m
0	0	12	64
1	0	13	64
2	0	14	64
3	0	15	64
4	0	16	101
5	0	17	101
6	0	18	128
7	0	19	165
8	0	20	165
9	0	21	180
10	0	22	213
11	64		

Table 7.6-14: Threshold in quiet (TQ_m)for each perceptual band

Band (m)	TQ_m	Band (m)	TQ_m
0	133	12	282
1	133	13	307
2	133	14	330
3	133	15	352
4	133	16	404
5	133	17	461
6	133	18	525
7	133	19	631
8	138	20	1249
9	159	21	1511
10	183	22	1519
11	241		

Table 7.6-15. Sensation level offset gradient (D_m) for each perceptual band

Band (m)	D_m	Band (m)	D_m
0	112	12	64
1	112	13	64
2	96	14	64
3	80	15	64
4	80	16	64
5	64	17	64
6	64	18	64
7	64	19	64
8	64	20	64
9	64	21	64
10	64	22	64
11	64		

Table 7.6-16. Spreading function (SF_{ml})

1.1.450 m	1.1.451 SF_{ml}
1.1.452 0	1.1.453 0, -1561, -2552
1.1.454 1	1.1.455 -289, -4, -1234, -2295, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.456 2	1.1.457 -569, -229, -8, -905, -1705, -2324, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.458 3	1.1.459 -789, -445, -173, -16, -656, -1271, -1765, -2172, -2520, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.460 4	1.1.461 -961, -616, -340, -136, -28, -488, -976, -1382, -1729, -2032, -2305, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.462 5	1.1.463 -1088, -743, -465, -257, -148, -31, -371, -769, -1114, -1417, -1689, -2054, -2483, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.464 6	1.1.465 -1198, -852, -574, -364, -209, -148, -42, -300, -635, -936, -1207, -1572, -2000, -2376, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.466 7	1.1.467 -1293, -948, -669, -458, -301, -183, -145, -56, -258, -547, -816, -1179, -1606, -1982, -2311, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.468 8	1.1.469 -1375, -1029, -750, -539, -381, -260, -180, -142, -68, -231, -487, -846, -1272, -1647, -1976, -2261, -2552, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.470 9	1.1.471 -1444, -1099, -820, -608, -449, -328, -233, -194, -138, -77, -213, -555, -978, -1352, -1681, -1966, -2268, -2552, -2552, -2552, -2552, -2552, -2552, -2552
1.1.472 10	1.1.473 -1501, -1155, -876, -665, -505, -383, -287, -210, -193, -130, -79, -298, -711, -1083, -1411, -1696, -1997, -2288, -2550, -2552, -2552, -2552, -2552
1.1.474 11	1.1.475 -1567, -1221, -942, -730, -570, -448, -351, -272, -206, -189, -151, -72, -349, -713, -1039, -1324, -1625, -1915, -2177, -2448, -2552, -2552, -2552
1.1.476 12	1.1.477 -1650, -1304, -1025, -813, -653, -530, -432, -352, -285, -227, -177, -163, -69, -297, -613, -895, -1195, -1485, -1746, -2017, -2238, -2401, -2545
1.1.478 13	1.1.479 -1727, -1381, -1102, -890, -730, -607, -509, -428, -360, -301, -249, -180, -153, -72, -257, -527, -824, -1112, -1373, -1643, -1865, -2028, -2171
1.1.480 14	1.1.481 -1798, -1452, -1173, -960, -800, -677, -579, -498, -430, -370, -317, -246, -192, -145, -76, -224, -505, -790, -1050, -1320, -1540, -1703, -1847
1.1.482 15	1.1.483 -1860, -1514, -1234, -1022, -862, -738, -640, -559, -490, -430, -377, -306, -224, -197, -136, -81, -242, -515, -771, -1040, -1260, -1422, -1566
1.1.484 16	1.1.485 -1923, -1577, -1297, -1085, -925, -801, -703, -621, -553, -492, -439, -367, -284, -213, -198, -144, -83, -235, -479, -744, -963, -1125, -1268
1.1.486 17	1.1.487 -1986, -1640, -1360, -1148, -988, -864, -766, -684, -615, -555, -501, -429, -345, -273, -211, -204, -146, -89, -216, -465, -680, -841, -984
1.1.488 18	1.1.489 -2043, -1697, -1417, -1205, -1044, -921, -822, -741, -672, -611, -557, -485, -401, -328, -264, -211, -205, -140, -93, -227, -430, -588, -729
1.1.490 19	1.1.491 -2104, -1758, -1479, -1266, -1106, -982, -884, -802, -733, -673, -619, -546, -461, -388, -324, -269, -212, -211, -151, -100, -195, -336, -472
1.1.492 20	1.1.493 -2163, -1817, -1537, -1324, -1164, -1040, -942, -860, -791, -731, -676, -604, -519, -445, -380, -325, -268, -226, -219, -147, -114, -167, -280
1.1.494 21	1.1.495 -2203, -1857, -1577, -1365, -1205, -1081, -982, -901, -831, -771, -717, -644, -559, -485, -420, -364, -306, -252, -239, -206, -132, -122, -163

1.1.496 22	1.1.497 -2224, -1878, -1598, -1386, -1225, -1102, -1003, -921, -852, -792, -737, -665, -580, -505, -441, -385, -326, -271, -222, -224, -176, -121, -114

Table 7.6-17. Log Add look up table values ($L[a]$)

a	L[a]	a	L[a]	a	L[a]	a	L[a]	a	L[a]	a	L[a]	a	L[a]	a	L[a]
0	64	64	37	128	21	192	11	256	6	320	3	384	1	448	1
1	64	65	37	129	20	193	11	257	6	321	3	385	1	449	1
2	63	66	37	130	20	194	11	258	5	322	3	386	1	450	1
3	63	67	36	131	20	195	11	259	5	323	3	387	1	451	1
4	62	68	36	132	20	196	10	260	5	324	3	388	1	452	1
5	62	69	36	133	20	197	10	261	5	325	3	389	1	453	1
6	61	70	35	134	19	198	10	262	5	326	3	390	1	454	1
7	61	71	35	135	19	199	10	263	5	327	3	391	1	455	1
8	60	72	35	136	19	200	10	264	5	328	3	392	1	456	1
9	60	73	35	137	19	201	10	265	5	329	3	393	1	457	1
10	59	74	34	138	19	202	10	266	5	330	3	394	1	458	1
11	59	75	34	139	19	203	10	267	5	331	3	395	1	459	1
12	58	76	34	140	18	204	10	268	5	332	2	396	1	460	1
13	58	77	33	141	18	205	10	269	5	333	2	397	1	461	1
14	57	78	33	142	18	206	9	270	5	334	2	398	1	462	1
15	57	79	33	143	18	207	9	271	5	335	2	399	1	463	1
16	56	80	32	144	18	208	9	272	5	336	2	400	1	464	1
17	56	81	32	145	17	209	9	273	5	337	2	401	1	465	1
18	55	82	32	146	17	210	9	274	5	338	2	402	1	466	1
19	55	83	32	147	17	211	9	275	5	339	2	403	1	467	1
20	55	84	31	148	17	212	9	276	5	340	2	404	1	468	1
21	54	85	31	149	17	213	9	277	4	341	2	405	1	469	1
22	54	86	31	150	17	214	9	278	4	342	2	406	1	470	1
23	53	87	30	151	16	215	9	279	4	343	2	407	1	471	1
24	53	88	30	152	16	216	8	280	4	344	2	408	1	472	1
25	52	89	30	153	16	217	8	281	4	345	2	409	1	473	1
26	52	90	30	154	16	218	8	282	4	346	2	410	1	474	1
27	51	91	29	155	16	219	8	283	4	347	2	411	1	475	1
28	51	92	29	156	16	220	8	284	4	348	2	412	1	476	1
29	51	93	29	157	15	221	8	285	4	349	2	413	1	477	1
30	50	94	28	158	15	222	8	286	4	350	2	414	1	478	1
31	50	95	28	159	15	223	8	287	4	351	2	415	1	479	1
32	49	96	28	160	15	224	8	288	4	352	2	416	1	480	1
33	49	97	28	161	15	225	8	289	4	353	2	417	1	481	1
34	49	98	27	162	15	226	8	290	4	354	2	418	1	482	0
35	48	99	27	163	15	227	8	291	4	355	2	419	1	483	0
36	48	100	27	164	14	228	8	292	4	356	2	420	1	484	0
37	47	101	27	165	14	229	7	293	4	357	2	421	1	485	0
38	47	102	26	166	14	230	7	294	4	358	2	422	1	486	0
39	47	103	26	167	14	231	7	295	4	359	2	423	1	487	0
40	46	104	26	168	14	232	7	296	4	360	2	424	1	488	0
41	46	105	26	169	14	233	7	297	4	361	2	425	1	489	0

42	45	106	25	170	14	234	7	298	4	362	2	426	1	490	0
43	45	107	25	171	13	235	7	299	4	363	2	427	1	491	0
44	45	108	25	172	13	236	7	300	4	364	2	428	1	492	0
45	44	109	25	173	13	237	7	301	3	365	2	429	1	493	0
46	44	110	24	174	13	238	7	302	3	366	2	430	1	494	0
47	43	111	24	175	13	239	7	303	3	367	2	431	1	495	0
48	43	112	24	176	13	240	7	304	3	368	2	432	1	496	0
49	43	113	24	177	13	241	7	305	3	369	2	433	1	497	0
50	42	114	24	178	13	242	6	306	3	370	2	434	1	498	0
51	42	115	23	179	12	243	6	307	3	371	2	435	1	499	0
52	42	116	23	180	12	244	6	308	3	372	2	436	1	500	0
53	41	117	23	181	12	245	6	309	3	373	2	437	1	501	0
54	41	118	23	182	12	246	6	310	3	374	2	438	1	502	0
55	41	119	22	183	12	247	6	311	3	375	2	439	1	503	0
56	40	120	22	184	12	248	6	312	3	376	2	440	1	504	0
57	40	121	22	185	12	249	6	313	3	377	2	441	1	505	0
58	39	122	22	186	12	250	6	314	3	378	2	442	1	506	0
59	39	123	22	187	11	251	6	315	3	379	2	443	1	507	0
60	39	124	21	188	11	252	6	316	3	380	1	444	1	508	0
61	38	125	21	189	11	253	6	317	3	381	1	445	1	509	0
62	38	126	21	190	11	254	6	318	3	382	1	446	1	510	0
63	38	127	21	191	11	255	6	319	3	383	1	447	1	511	0

7.6.4.2.7 Linear Prediction

7.6.4.2.7.1 Overview

Since the frequency resolution of the LCLD frequency bands is relatively coarse (400 Hz at 48kHz sample rate) there is no significant coding gain for signals with strong harmonic content. However, the signal redundancy within LCLD bands for this type of signal can be greatly reduced by linear prediction over time and the SNR increased. .

7.6.4.2.7.2 Bitstream Syntax

For each frame flags for every LCLD band in a certain LCLD band range are transmitted in the bitstream which enable or disable linear predictive coding (LPC). If LPC is enabled for a certain band the magnitude and the phase of the first order prediction parameter are transmitted at the total cost of 8 bit per band and frame. Additional bitstream elements indicate if any LPC is active for a channel and maximum band using LPC. For short frames like 10ms or 5ms a dedicated signalling scheme is used to reduce the prediction side rate.

The prediction processing at the encoder is done in a closed loop fashion, and a dedicated Huffman code book is used to encode the prediction residual. Due to the closed loop encoding, the prediction is part of the quantization and bit allocation loop which determines the allocation offset and associated quantizers for each perceptual band.

7.6.4.2.7.3 Prediction for 20ms frames

For every LCLD band b with active prediction, the quantized prediction residual is computed as

$$X_{n,b}^{Q} = Q\left(X_{n,b} - S_{n-1,b}^{Q} \cdot P_{b}^{Q}\right),\ n = 0,1,\ldots,N_{frame} - 1\ ,$$

where $X_{n,b}$ is the complex LCLD input sample at index n and LCLD band b.

Note that the Superscript Q is used here to indicate quantized and inverse quantized numbers,

$S^Q_{n-1,b}$ is the prediction state which is computed as:

$$S^Q_{n,b} = X^Q_{n,b} + S^Q_{n-1,b} P^Q_b \ ,$$

where $P^Q_{n,b}$ is the quantized Prediction coefficient.

For 20ms frame length, the first sample in each LCLD band is not predicted, thus the prediction state is initialized as $S^Q_{-1,b} = 0$.

$Q(.)$ indicates quantization and inverse quantization of a complex number, where the scale factor and the quantization range are determined by the perceptual model and the allocation offset.

7.6.4.2.7.4 Prediction for frames shorter than 20ms

The prediction operation depends on a certain subset of LCLD bands (subset in the following text) and the number of possible subsets. Typically, the number K of possible subsets is

$$K = \frac{16}{N_{frame}} \ .$$

For example, $N_{frame} = 4$ in the case of 5ms frame length.

Then bands b_s belonging to the subset s are defined as

$$b_s = \{b \in mod(b, K) = s\},\ b = 0,1, \ldots, N_{LCLD} - 1,\ s = 0,1, .., K - 1 \ ,$$

where N_{LCLD} is the total number LCLD bands (60).

For each frame the encoder determines a current subset s_{cur} which is, together with the number K of possible subsets, transmitted in the bitstream. The current subset shall cycle through the possible subsets in ascending order as $0,1, \ldots, K-1, 0, 1, \ldots$ and so on for each new frame.

The prediction state for the first sample in the current frame for all bands belonging to the current subset is initialized to zero just as in the 20ms frame length case:

$$S^Q_{-1,b_{s_{cur}}} = 0 \ .$$

Otherwise, the state is according to the last sample of the previous frame. In other words, the prediction is only reset at the beginning of a frame if the LCLD band belongs to the current subset of LCLD bands.

7.6.4.2.7.5 Prediction Signalling

Prediction processing is determined by the following bitstream elements:

Table 7.6.-18: Prediction Syntax Elements

1.1.498 Syntax Element	1.1.499 Comment	1.1.500 Number of Bits
1.1.501 NumSubSets	1.1.502 Possible number of Subsets K	1.1.503 3
1.1.504 SubSetId	1.1.505 Current subset s_{cur}	1.1.506 if K > 4 1.1.507 3 1.1.508 Elseif K > 1 1.1.509 2 1.1.510 Else 1.1.511 0
1.1.512 PredChanEnable	1.1.513 Prediction active/passive flags, one bit for each possible subset	1.1.514 K 1.1.515 (per channel)
1.1.516 NumPredBands	1.1.517 The maximum LCLD band with active prediction for the current subset	1.1.518 if K >= 4 1.1.519 4 1.1.520 elseif K > 1 1.1.521 5 1.1.522 Else 1.1.523 6 1.1.524 (per channel)
1.1.525 PredBandEnable	1.1.526 Array of active/passive flags for all LCLD bands up to NumPredBands for each audio channel.	1.1.527 Int(NumPredBands / K)+1 1.1.528 (per channel)
1.1.529 PredictionMagnitude	1.1.530 For every active LCLD band belonging to the current subset for each audio channel	1.1.531 3 bits per active LCLD band
1.1.532 Prediction Phase	1.1.533 For every active LCLD band belonging to the current subset for each audio channel	1.1.534 5 bits per active LCLD band

7.6.4.2.7.6 Quantization of Prediction Parameters

The prediction magnitude quantization indices are computed as follows:

$$P_M^{idq} = int\left(scale_M \cdot \arcsin(P_M) + 0.5\right) ,$$

where P_M is the optimal prediction magnitude and the scale is given as

$$scale_M = \frac{17}{\pi} .$$

The “quantized” (quantized and inverse quantized) prediction magnitude is given as

$$P_M^Q = \sin\left(P_M^{idq} \cdot \frac{1}{scale}\right) .$$

The prediction phase quantization indices are computed as

$$P_{Ph}^{idq} = round(scale_{Ph} P_{Ph}) \ ,$$

where *round()* indicates rounding to the nearest integer and the quantized prediction phase is given by

$$P_{Ph}^{Q} = P_{Ph}^{idq} \cdot \frac{1}{scale_{Ph}} \ .$$

The scale is given as $scale_{Ph} = \frac{16}{\pi}$.

7.6.4.2.7.7 Estimation of Prediction Parameters

Regardless of the frame length prediction parameters are computed based on 20ms (16 samples at 48kHz) of audio data per LCLD band. Prediction parameters may be updated only for the current subset of LCLD bands (all LCLD bands if the number of possible subsets is 1). For LCLD bands belonging to other subsets which are not the current subset may either be deactivated or can be continuously used without any modification. This information is available in the bitstream element PredChanEnable as shown in table Prediction Syntax Elements.

The optimal, complex first order prediction coefficient per band b is computed as

$$P_b = \frac{R_{xx}^1}{R_{xx}^0},$$

where $R_{xx}^0$and R_{xx}^1 is the auto-correlation function at lag 0 and lag 1 respectively. The auto-correlation function at lag 0 (Energy) is computes as follows:

$$R_{xx,b}^0 = \sum_{n=0}^{N_{frame}^{20ms}-1} \left| X_{n,b} \right|^2 \ .$$

The auto-correlation function at lag 1 is computed as

$$R_{xx,b}^1 = \sum_{n=1}^{N_{frame}^{20ms}-1} X_{n-1,b} \cdot X_{n,b}^* \ .$$

The prediction magnitude is given as

$$P_{M,b} = |P_b| \ .$$

And the prediction phase is given as

$$P_{Ph,b} = \arctan 2 \left(\frac{imag(P_b)}{real(P_b)} \right) \ .$$

Based on the optimal prediction coefficient a prediction gain (ratio of signal energy to residual signal energy) is computed for every LCLD band as

$$G_b^P = \frac{1}{\left(1 - \min\left\{P_{b,M}, P_{b,M}^Q\right\}\right)} \ .$$

Based on the prediction gain a bitrate saving is estimated as follows:

$$BitsGain_b^P = 0.65 \cdot \log 2 \left(G_b^P\right) N_{frame}^{20ms} - 8 \ ,$$

where 8 corresponds to the fixed number of bits needed to transmit the prediction parameters.

Now, LCLD bands b with $BitsGain_b^P > 0$ are candidates for prediction.

Finally, the optimal maximum prediction band $B - 1$ is determined by maximizing the accumulated bitrate saving, subject to the cost of signalling the active prediction bands.

$$B = \arg\max \left(\sum_{b \in b_{s_{cur}}}^{B-1} BitsGain_b^P - N(B) - 7 \right) \ ,$$

where $N(B)$ is the number of prediction band candidates for $b < B$ and 7 corresponds to a fixed signalling cost.

To improve the recovery after frame loss, the audio data is written in chunks where each chunk corresponds to one subset of LCLD bands. The data associated with the current subset is always written first followed by the chunks containing data for subsets with decremented IDs. If, after decrementing, the subset ID becomes –1 then the subset ID of the written chunk is set to $K - 1$ (K being the number possible subsets).

7.6.4.2.8 Bit Allocation

The bit allocation routine distributes the available bits in a frame proportion to the target SNR (SMR) in each channel, group, and perceptual band. The allocation can be viewed as two components; a core allocation that takes the SMR generated by the perceptual model and a single control parameter ($AllocOffset$) to generate an allocation to channels, groups, and perceptual bands. The second component of the allocation in the encoder only is the search for the control parameter ($AllocOffset$). As the control parameter is transmitted to the decoder, the decoder can exactly mirror the core allocation routine, thus, mitigating the need to transmit explicit allocations for channels, groups, and bands.

The core allocation is shown in equation 7.6-45, similar to the perceptual model described in section 7.6.4.2.6, the core allocation uses integer arithmetic such that it can be exactly replicated in the decoder.

$$A_{c,g,m} = \left\lfloor \frac{SMR_{c,g,m} + 8 \cdot AllocOffset}{32} \right\rfloor , \quad (7.6\text{-}45)$$

where:

$A_{c,g,m}$ is the allocation level for channel c, group g, and perceptual band m,

$SMR_{c,g,m}$ is the target SNR for channel c, group g, and perceptual band m computed by the perceptual model, and

$AllocOffset$ is the control parameter that will be transmitted to the decoder.

The bit allocation control parameter ($AllocOffset$) is limited to the range between -128 and 127 and transmitted to the decoder as 8-bit unsigned value by adding 128 to the derived value.

The search for the control parameter requires the encoder to quantize the normalized CLDFB coefficients and count the number of bits needed to Huffman code the quantized CLDFB coefficients. The quantization and Huffman coding of the normalized CLDFB coefficients is described in section 7.6.4.2.9. Furthermore, if the prediction is enabled (see section 7.6.4.2.7), then the prediction stage must also be computed during the quantization and bit counting. The search for the control parameter could be brute force, that is, try every possible value of the $AllocOffset$ parameter and choose the value that would lead to the number of bits needed to code the quantized CLDFB coefficients that is closest to the available bits without exceeding the available bits. However, a binary search algorithm will arrive at the same solution while mitigating some complexity.

The iterative binary search to derive the allocation offset parameter can be summarised in the following steps:

1. Initialize $AllocOffset$ to 0 and $delta$ to 128
2. Compute the core allocation shown in equation 7.6-45
3. Quantize the normalized CLDFB coefficients and count the number of bits needed to Huffman code the coefficients.
4. Compare the number of bits used to code the CLDFB coefficients and the available bits, return to step 2 or terminate based on the following conditions:
    a. If the number of bits needed to the CLDFB coefficients is *greater* than the available bits and $delta$ is greater than 0, set $AllocOffset = AllocOffset - delta$, set $delta = delta \gg 1$, return to step 2.
    b. If the number of bits needed to the CLDFB coefficients is *greater* than the available bits and $delta$ is equal to 0, set $AllocOffset = AllocOffset - 1$, set $delta = 1$, return to step 2.
    c. If the number of bits needed to the CLDFB coefficients is *less* than the available bits and $delta$ is greater than 0, set $AllocOffset = AllocOffset + delta$, set $delta = delta \gg 1$, return to step 2.
    d. If the number of bits needed to the CLDFB coefficients is *less* than the available bits and $delta$ is equal to 0, terminate search.
    e. If the number of bits needed to the CLDFB coefficients is *less* than the available bits and $AllocOffset$ is equal to 127, terminate search.
    f. If the number of bits needed to the CLDFB coefficients is *equal* to the available bits, terminate the search.

7.6.4.2.9 Quantization of the Normalized CLDFB Coefficients

7.6.4.2.9.1 Overview

LCLD supports both direct quantization of the normalized CLDFB coefficients and quantization of differentially coded coefficients (DPCM), where the differential is relative to the previous slot value within the same CLDFB band. In both cases the quantization technique is the same. This section is split into three parts, the differential coding of the

normalized CLDFB coefficients, the quantization of either the differential residual or the direct normalized CLDFB coefficients, and the Huffman coding and transmission of the quantized CLDFB coefficients.

7.6.4.2.9.2 Differential Coding of the Normalized CLDFB Coefficients

The differential coding of the normalized CLDFB coefficients uses a complex first-order predictor stage that is running along time (slots) within the same CLDFB band. The calculation of the first-order predictor is described in section 7.6.4.2.7. The differential coding can be disabled based on the prediction gain achieved as described in section 7.6.4.2.7. For the longer LCLD frames, the first slot in a frame is not predicted, that is a predictor reset occurs at the start of the frame.

The prediction process is described in equation 7.6-46:

$$e_{nk} = \tilde{S}_{nk}^{CLDFB} + P_k \hat{S}_{n-1,k}^{CLDFB} \quad , \tag{7.6-46}$$

where:

e_{nk} is the prediction residual for the n^{th} slot and k^{th} CLDFB band,

$\tilde{S}_{nk}^{CLDFB}$ is the normalized CLDFB coefficient for the n^{th} slot and k^{th} CLDFB band,

$\hat{S}_{n-1,k}^{CLDFB}$ is the quantized normalized CLDFB coefficient for the previous slot and k^{th} CLDFB band, and

P_k is the prediction coefficient for the k^{th} CLDFB band.

If the n^{th} slot is directly after the predictor reset, then $\hat{S}_{n-1,k}^{CLDFB}$ is the quantized value from the previous slot. However, if the n^{th} slot follows a previously predicted slot, then the $\hat{S}_{n-1,k}^{CLDFB}$ is reconstructed from the quantized residual as shown in equation 7.6-47:

$$\hat{S}_{n-1,k}^{CLDFB} = \hat{e}_{n-1,k} - P_k \hat{S}_{n-2,k}^{CLDFB} \quad , \tag{7.6-47}$$

where $\hat{e}_{n-1,k}$ is the quantized residual signal from the n^{th} -1 slot and k^{th} CLDFB band.

7.6.4.2.9.3 Quantization of Normalized CLDFB coefficients and Prediction Residuals

The quantization of both normalized CLDFB coefficients and predictor residuals are the same and controlled by an allocation level generated by the bit allocation algorithm (see section 7.6.4.2.8). The allocation level (A_{gm}) is calculated for groups and perceptual bands, so the following equations assume the n^{th} slot is a member of the g^{th} group and the k^{th} CLDFB band is a member of the m^{th} perceptual band. The quantization of either the normalized CLDFB coefficients or the prediction residual is shown in equation 7.6-48:

$$I_{nk} = ROUND\left(g\left[A_{gm}\right] \cdot x_{nk}\right) \quad , \tag{7.6-48}$$

where:

Q_{nk} is a complex signed integer that will be Huffman coded and transmitted to the decoder,

A_{gm} is the allocation level for the g^{th} group and m^{th} perceptual band,

$g\left[A_{gm}\right]$ is the allocation level scaling factor provided in table 7.6-19, and

x_{nk} is either the normalized CLDFB coefficient ($\tilde{S}_{nk}^{CLDFB}$) or the prediction residual (e_{nk}) for the n^{th} slot and k^{th} CLDFB band. The n^{th} slot is assumed to be an element of the g^{th} group and the k^{th} CLDFB band is assumed to be an element of the m^{th} perceptual band.

As the quantized values for both the normalized CLDFB and the residual are needed when the prediction is enabled, equation 7.6-49 shows the inverse quantization from the quantization index (I_{nk}):

$$\hat{x}_{nk} = g^{-1}\left[A_{gm}\right] \cdot Q_{nk} \quad , \tag{7.6-49}$$

where:

$\hat{x}_{nk}$ is either the quantized CLDFB coefficient or the quantized residual and

$g^{-1}[A_{gm}]$ is the inverse quantization scale factor provided in Table 7.6-19.

Table 7.6-19. Quantizer scaling factors () and inverse quantizer scaling factors () for each allocation level (A_{gm})

A_{gm}	$g[A_{gm}]$	$g^{-1}[A_{gm}]$	A_{gm}	$g[A_{gm}]$	$g^{-1}[A_{gm}]$
0	0	0	16	4.7568	0.2098
1	0.3536	2.3675	17	5.6569	0.1765
2	0.4204	2.0464	18	6.7272	0.1485
3	0.5	1.7759	19	8	0.1249
4	0.5946	1.5364	20	9.5137	0.1051
5	0.7071	1.3231	21	11.3137	0.0884
6	0.8409	1.1329	22	13.4543	0.0743
7	1	0.9658	23	16	0.0625
8	1.1892	0.8213	24	19.0273	0.0526
9	1.4142	0.6951	25	22.6274	0.0442
10	1.6818	0.5878	26	26.9087	0.0372
11	2	0.4958	27	32	0.0312
12	2.3784	0.4181	28	38.0546	0.0263
13	2.8284	0.3522	29	45.2548	0.0221
14	3.3636	0.2962	30	53.8174	0.0186
15	4	0.2494	31	64	0.0156

7.6.4.2.9.4 Huffman Coding of Quantized Normalized CLDFB coefficients and Quantized Prediction Residuals

As discussed in the previous section (7.6.4.2.9.3), the output of the quantizer is a complex signed integer (Q_{nk}) that will Huffman coded and transmitted to the decoder. Prior to Huffman coding, the complex signed integer is first separated into real and imaginary as shown in equation 7.6-50 and then further separated into a magnitude ($|Q_{nk}^r|$and $|Q_{nk}^i|$) and a sign bit as shown in equation 7.6-51 for both real and imaginary parts:

$$\begin{aligned} Q_{nk}^r &= \Re\{Q_{nk}\} \\ Q_{nk}^i &= \Im\{Q_{nk}\} \end{aligned} \tag{7.6-50}$$

$$\begin{aligned} sign_{nk}^r &= \begin{cases} 1, & Q_{nk}^r < 0 \\ 0, & Q_{nk}^r \geq 0 \end{cases} \\ sign_{nk}^i &= \begin{cases} 1, & Q_{nk}^i < 0 \\ 0, & Q_{nk}^i \geq 0 \end{cases} \end{aligned} . \tag{7.6-51}$$

The magnitude component is then limited to a max value that is dependent on the allocation level (A_{gm}), as shown in equation 7.6-52:

$$\begin{aligned} |Q_{nk}^r| &= \begin{cases} |Q_{nk}^r|, & |Q_{nk}^r| < max[A_{gm}] \\ max[A_{gm}], & |Q_{nk}^r| \geq max[A_{gm}] \end{cases} \\ |Q_{nk}^i| &= \begin{cases} |Q_{nk}^i|, & |Q_{nk}^i| < max[A_{gm}] \\ max[A_{gm}], & |Q_{nk}^i| \geq max[A_{gm}] \end{cases} \end{aligned} , \tag{7.6-52}$$

where $max[A_{gm}]$ is the maximum supported value depending on the allocation level A_{gm}, the maximum values are provided in Table 7.6-20.

The Huffman codebook used to code the real and imaginary components is dependent on the allocation level (A_{gm},) and dependent on whether the underlying quantized values are prediction residuals or normalized CLDFB coefficients.

Furthermore, for low values of the allocation level, the real and imaginary quantized magnitudes are packed into a single codeword. The Huffman packaging depends on the following logic:

- If A_{gm} is equal to 0, no information is transmitted, and the decoder generates 0's for all CLDFB coefficients in the g^{th} group and m^{th} perceptual band.
- If A_{gm} is greater than 0 and less than 12, then the real and imaginary parts of the quantized magnitude are packed into a single code word. After the Huffman code the sign bit for Q_{nk}^{r} if $|Q_{nk}^{r}|$ is greater than 0 then the sign bit for Q_{nk}^{i} if $|Q_{nk}^{i}|$ is greater than 0. The code and number of bits written can be expressed as a table look up as follows:

$$[code, length] = HuffLookup[res_{nk}][A_{gm}][|Q_{nk}^{r}| \cdot (\max[[A_{gm}] + 1) + |Q_{nk}^{i}|],$$

where:

$code$ is the codeword written to stream,

$length$ is the number bits to write the code, and

res_{nk} is a binary value that is 1 if the underlying quantized values are prediction residuals or 0 if the underlying quantized values are normalized CLDFB coefficients.

- If A_{gm} is greater than or equal to 12, then the real and imaginary parts of the quantized magnitude are packed into separate codewords. After the Huffman code the sign bit for Q_{nk}^{r} if $|Q_{nk}^{r}|$ is greater than 0 then the sign bit for Q_{nk}^{i} if $|Q_{nk}^{i}|$ is greater than 0. The code and number of bits written can be expressed as a table look up as follows:

$$[code^{r}, length^{r}] = HuffLookup[res_{nk}][A_{gm}][|Q_{nk}^{r}|],$$

$$[code^{i}, length^{i}] = HuffLookup[res_{nk}][A_{gm}][|Q_{nk}^{i}|].$$

Due to the size of the Huffman lookup tables, the tables can be found in the reference code.

Table 7.6-20. Maximum values $max[A_{gm}]$ per allocation level A_{gm}

A_{gm}	$max[A_{gm}]$	A_{gm}	$max[A_{gm}]$
0	0	16	26
1	3	17	26
2	3	18	27
3	4	19	28
4	5	20	31
5	5	21	36
6	6	22	38
7	7	23	45
8	8	24	54
9	9	25	64
10	12	26	76
11	13	27	90
12	16	28	108
13	17	29	128
14	19	30	152
15	23	31	180

7.6.4.3 LCLD decoder

7.6.4.3.1 Overview

Figure 7.6-3 shows a high-level schematic of the LCLD decoder structure. As shown in figure 7.6-3, the grouping information that controls the number and update of the RMS envelope is first decoded as described in section 7.6.4.3.2. Following the grouping information, the RMS envelope is Huffman decoded and reconstructed as described in section7.6.4.3.3. The RMS envelope values are then used to compute a backward adaptive perceptual model and bit allocation as decreed in sections 7.6.4.3.4 and 7.6.4.3.5 respectively. The bit allocation information derived in the decoder is then used to Huffman decode and inverse quantize the CLDFB coefficients as described in section 7.6.4.3.6. Following inverse quantization, a prediction filter is applied as described in section 7.6.4.3.7. The CLDFB coefficients are then inverse normalized to reinstate the original signal's energy levels as described in section 7.6.4.3.8. The final stage of the decoder is the application of joint channel decoding, which is described in section 7.6.4.3.9.

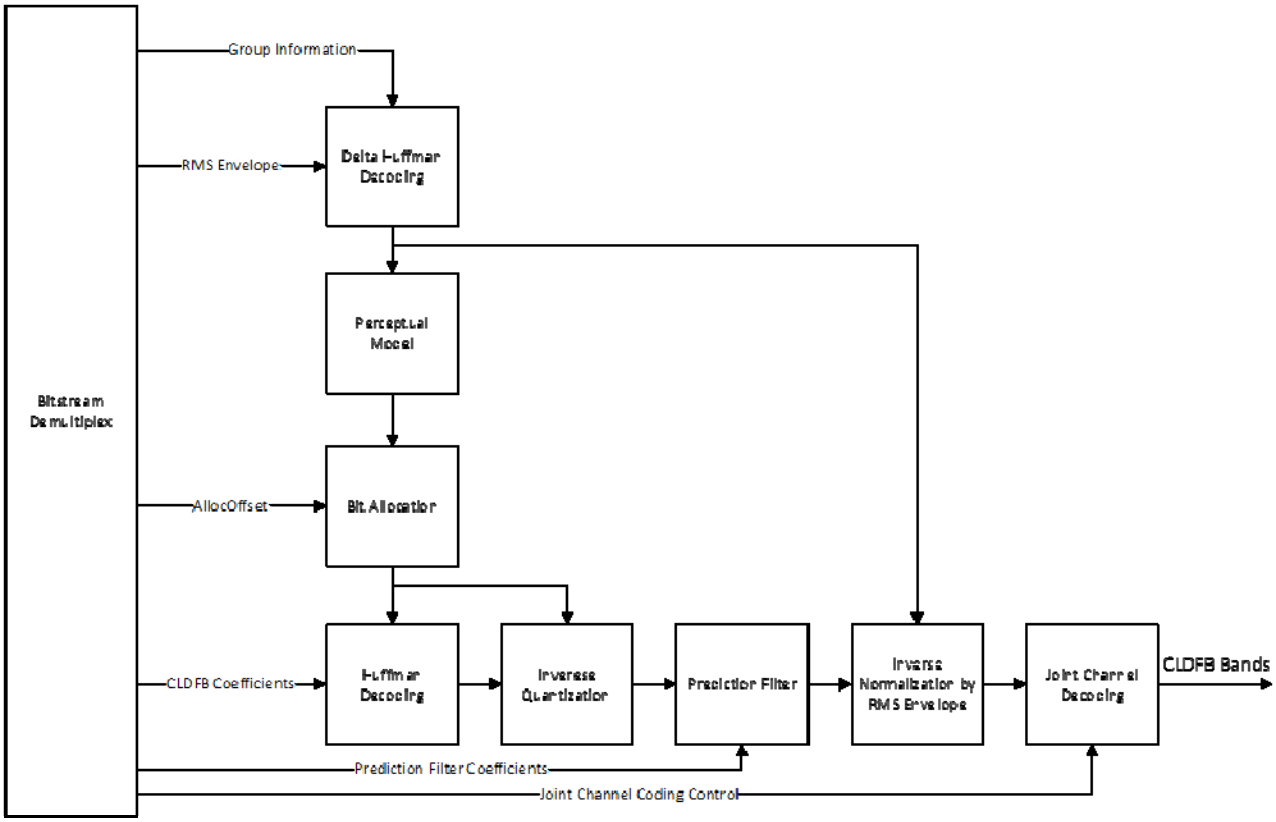


Figure 7.6-3: The structure of the LCLD decoder

7.6.4.3.2 Decoding Group Information

The group information which controls the number of RMS envelopes and therefore bit-allocation is coded as a set of single-bit flags. A value 1 indicates the start of a new group, while a value of 0 indicates the continuation of a group. The first slot in a frame is always the start of a new group and no flag is transmitted for the first slot in a frame. If a flame contains 16 slots there will be 15 flags containing group information. For 2 channel content, a single-bit flag is transmitted prior to the grouping information to specify common grouping. If the common grouping flag is 1, then a single set of grouping information follows and both channels share grouping information. If the common grouping flag is 0, then the channels have independent groups and 2 sets of grouping information follow.

7.6.4.3.3 Decoding RMS Envelope Information

RMS envelope information is present in the bitstream for each channel and each group for that channel. The ordering of RMS envelope information is by channel and then group. The RMS envelope in each channel and each group are differentially and Huffman coded except for the envelope values for the first perceptual band (0^{th} index). The envelope value for first perceptual band for all groups in a channel is transmitted as an absolute number in an 7-bit unsigned

integer. To obtain the signed RMS envelope for the first perceptual band 64 is subtracted from the value read from the bitstream. Equation 7.6-53 shows the process to obtain the RMS envelope values for remaining perceptual bands in a group:

$$R_{gm} = R_{g,m-1} + \Delta R_{gm} \ \forall m > 0 \ , \quad (7.6\text{-}53)$$

where:

R_{gm} is the RMS envelope values for the g^{th} group and m^{th} perceptual band, and

ΔR_{gm} is the RMS envelope delta that is Huffman decode from the bitstream.

The Huffman decode tables can be obtained from the reference code.

7.6.4.3.4 Perceptual Model

The perceptual model used in the decoder is the same as the perceptual model calculation in the encoder, see section 7.6.4.2.6.

7.6.4.3.5 Bit Allocation

The bit allocation in the LCLD is the same as the core allocation in the encoder (see section 7.6.4.2.8). The firsts step of the bit allocation is to read the *AllocOffse*t from the bitstream, which is transmitted as an 8-bit unsigned integer. The *AllocOffset* parameter is then converted back to a signed integer by subtracting 128. The bit allocation for each channel, group, and perceptual band is then calculated as shown in equation 7.6-54:

$$A_{c,g,m} = \left\lfloor \frac{SMR_{c,g,m} + 8 \cdot AllocOffset}{32} \right\rfloor \ , \quad (7.6\text{-}54)$$

where:

$A_{c,g,m}$ is the allocation level for channel c, group g, and perceptual band m,

$SMR_{c,g,m}$ is the target SNR for channel c, group g, and perceptual band m computed by the perceptual model, and

$AllocOffset$ is the control parameter that is read from the bitstream.

7.6.4.3.6 Normalized CLDFB Coefficient and Prediction Residual Huffman Decoding and Inverse Quantization

The normalized CLDFB coefficients or the prediction residual in the bitstream are Huffman coded. The following description assumes the n^{th} slot is an element of the g^{th} group and the k^{th} CLDFB band is element of the m^{th} perceptual band. The specific Huffman codebook used to code the coefficients or residual is dependent on the allocation level (A_{gm}) and whether the values are normalized CLDFB coefficients or prediction residuals. The quantized values that are Huffman coded in the bitstream represent complex integers. The complex integers are expressed as separate magnitudes and sign bits. The packaging of the complex integers is dependent on the allocation level (A_{gm}) as follows:

- If A_{gm} is equal to 0, no information is in the bitstream and all values in the g^{th} group and m^{th} perceptual band are set to 0.
- If A_{gm} is greater than 0 and less than 12, the real and imaginary parts of the complex integer (Q_{nk}) are packed into a single Huffman codeword. After Huffman decoding, the magnitude of the real and imaginary parts of the complex integer can be obtained with the following equations 7.6-55 and 7.6-56. If $|Q_{nk}^{r}|$ is greater than 0, then a sign bit ($sign_{nk}^{r}$) for real part of the complex integer follows in the bitstream. If $|Q_{nk}^{i}|$ is greater than 0, then a sign bit ($sign_{nk}^{i}$) for imaginary part of the complex integer follows in the bitstream.

$$|Q_{nk}^{r}| = \left\lfloor \frac{value}{(max[A_{gm}]+1)} \right\rfloor \ , \quad (7.6\text{-}55)$$

$$|Q_{nk}^{i}| = mod(value, max[A_{gm}] + 1) \ , \quad (7.6\text{-}56)$$

where:

$|Q_{nk}^{r}|$ is the magnitude of the real part of the complex integer,

$|Q_{nk}^{i}|$ is the magnitude of the imaginary pert of the complex integer,
$value$ is the Huffman decoded value from the bitstream, and
$max[A_{gm}]$ is the max values which are provided in Table 7.6-20 in section 7.6.4.7.3.

- If A_{gm} is greater than or equal to 12, then the magnitude of the real part of the complex integer (Q_{nk}) and the magnitude of the imaginary part of the complex integer (Q_{nk}) are coded in separate Huffman codes. If $|Q_{nk}^{r}|$ is greater than 0, then a sign bit for real part of the complex integer follows in the bitstream. If $|Q_{nk}^{r}|$ is greater than 0, then a sign bit ($sign_{nk}^{r}$) for real part of the complex integer follows in the bitstream. If $|Q_{nk}^{i}|$ is greater than 0, then a sign bit ($sign_{nk}^{i}$) for imaginary part of the complex integer follows in the bitstream.
- Once the magnitudes of the real and imaginary parts are Huffman decode and the sign bits are read, the complex integer (Q_{nk}) can be reconstructed as shown in equation 7.6-57:

$$Q_{nk} = sign_{nk}^{r} \cdot |Q_{nk}^{r}| + i \cdot sign_{nk}^{i} \cdot |Q_{nk}^{i}| \ . \quad (7.6\text{-}57)$$

While the Huffman decoding is dependent on the type of values in the bitstream, the inverse quantization is only dependent on the allocation level as shown in equation 7.6-58.

$$\hat{X}_{nk} = g^{-1}[A_{gm}] \cdot Q_{nk} \ , \quad (7.6\text{-}58)$$

where:

$\hat{X}_{nk}$ is either the quantized CLDFB coefficient or the quantized residual,

Q_{nk} is the complex signed integer decode from the bit stream, and

$g^{-1}[A_{gm}]$ is the inverse quantization scale factor provided in table 7.6-19 in section 7.6.4.7.2.

7.6.4.3.7 Inverse Prediction

7.6.4.3.7.1 Overview

The reconstruction of transmitted prediction residuals is identical to the processing at the encoder side where the prediction state corresponds to the reconstructed sample as shown below:

$$S_{n,b}^{Q} = X_{n,b}^{Q} + S_{n-1,b}^{Q} P_{b}^{Q} \ ,$$

where $S_{n,b}^{Q}$ is the reconstructed complex sample at time sample index n and LCLD band b and $X_{n,b}^{Q}$ are the transmitted residual samples.

7.6.4.3.7.2 Status Tracking after Frame Loss

In case of short frames, part of the prediction parameters may be held constant for one or more frames as indicated in the syntax element PredChanEnable. After frame loss, for this this type of prediction metadata the data in the decoder memory might be incorrect which is indicated as unresolved in the following text. Since the selection of Huffman code books to decode the audio data depends on the prediction metadata, audio data cannot be safely decoded. To handle this situation, the decoder keeps track on the Unresolved status of prediction metadata in a persistent array named DecodingUnresolved with of size NumChannels x NumPossibleSubsets.

During frame loss all entries of DecodingUnresolved are set to true (no prediction data present), When reading a good frame after frame loss the variable is updated as follows:

For the current subset prediction metadata is always present in the bitstream and the status for the current subset will be set to Resolved (false in the variable). If prediction is inactive for a subset, then the status is also resolved. For other subsets the status will remain unresolved in the first good frame after frame loss.

Based on the DecodingResolved status, audio data is decoded in chunks corresponding to subsets until an Unresolved status occurs. The information of correctly decoded subsets is stored in the additional, persistent arrays DecodingFailed and DecodingFailedPrev for the current and the previous frame respectively which are used to guide packet loss concealment for the first few good frames after packet loss as shown in Figure 7.6-4.

During frame loss all entries of DecodingFailed and DecodingFailedPrev are set to true (no data decoded). During frame loss and shortly after frame loss, data for subsets for which decoding failed are concealed (all data during frame loss). Correctly decoded subsets are combined with concealed subsets until all subsets can be decoded correctly. If the

status changes from concealed to decoded (as indicated by variables DecodingFailed and DecodingFailedPrev), then the first few samples are crossfaded from concealed to decoded data.

More details on LCLD packet loss concealment are provided in clause 7.6.4.4.

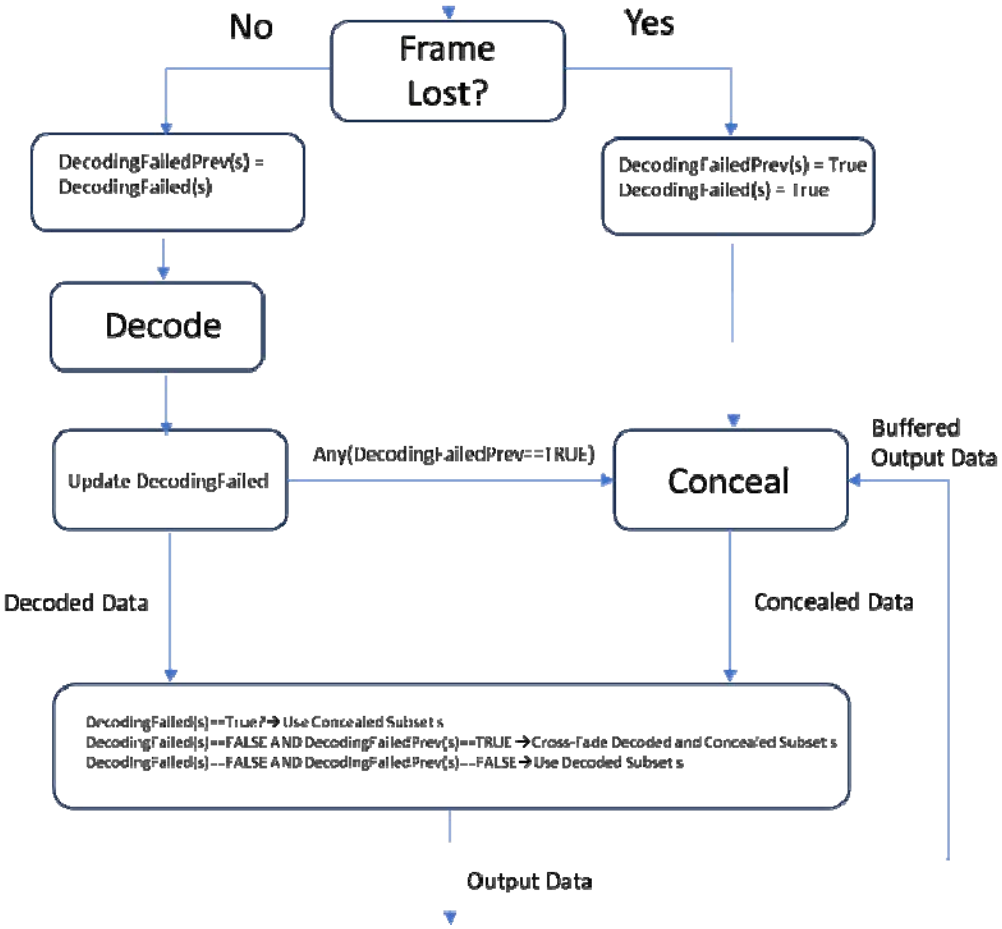


Figure 7.6-4: High-level flowchart for concealment operations during packet loss and following packet loss

7.6.4.3.8 Inverse RMS Envelope Normalization

The inverse normalization of the CLDFB coefficients by the RMS envelope returns the normalized spectrum to the original energy levels. In this description of the inverse normalization process, the n^{th} slot is assumed to be an element of the g^{th} group and the k^{th} CLDFB band is assumed to be an element the m^{th} perceptual band. The inverse normalization of the CLDFB coefficients are shown in equation 7.6-59:

$$S_{nk}^{CLDFB} = \tilde{S}_{nk}^{CLDFB} 2^{0.5 \cdot R_{gm}} \quad , \tag{7.6-59}$$

where:

S_{nk}^{CLDFB} is the inverse normalized CLDFB coefficients for the n^{th} slot and k^{th} CLDFB band,

$\tilde{S}_{nk}^{CLDFB}$ is the normalized CLDFB coefficients for the n^{th} slot and k^{th} CLDFB band, and

R_{gm} is the RMS envelope for the g^{th} group that contains the n^{th} slot and m^{th} perceptual band that contains the k^{th} CLDFB band.

7.6.4.3.9 Inverse Joint Stereo Processing

Depending on the read bitstream element MSMode and other flags, inverse joint stereo processing is applied for all LCLD bands b belonging to perceptual band k as described in the below matrix notation:

$$\begin{bmatrix} X_{b,n}^{L} \\ X_{b,n}^{R} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\varphi_k^Q} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ p_k^Q & 1 \end{bmatrix} \begin{bmatrix} X_{b,n}^{M} \\ X_{b,n}^{S} \end{bmatrix} ,$$

where $X_{b,n}^{L}$ and $X_{b,n}^{R}$ are the decoded left and right complex input samples, $X_{b,n}^{M}$ and $X_{b,n}^{S}$ are the transmitted jointly coded samples and n is the time sample index within the coded frame.

The joint stereo coding parameters p_k^Q and φ_k^Q are the transmitted prediction and phase alignment parameters which are found as follows:

- Prediction Parameters
  - Undo frequency differential coding of prediction parameters
  - De-Quantization of prediction parameters
- Phase Parameters
  - Undo second order frequency differential coding
  - Wrap indices into the quantization range of [-12,12]
  - Derive phase rotation factors $e^{-j\varphi_k^Q}$ by table-lookup

7.6.4.4 LCLD packet loss concealment

7.6.4.4.1 General

LCLD packet loss concealment is a low-complexity method operating in CLDFB domain. Whenever a frame is marked as lost or unavailable by the BFI flag raised, frame loss concealment operations produce substitution CLDFB coefficients for each CLDFB band. The band-wise operation follows either a sinusoidal extension strategy or a strategy based on linear prediction combined with repetition of CLDFB coefficients of the same CLDFB band of the preceding frame. The selection of strategy is made based on a tonality determination of the CLDFB band signal. After band-wise substitution frame generation in CLDFB domain, the substitution frame is converted to time domain using CLDFB synthesis like a regularly decoded good frame.

To ensure smooth signal evolution at frame boundaries, a crossfade mechanism is active at transitions from a substitution frame to a succeeding good frame.

Also provided is a mechanism for burst loss handling including muting.

Figure 7.6-4 above is a high-level flowchart of the LCLD frame loss concealment operations. It illustrates how and under what conditions decoded (good) frames and concealed, i.e., substituted frames, are used to generate an output frame to be converted back to time domain using CLDFB synthesis. It is further shown how state information about successfully decoding of a previous frame and a current frame is used to decide if crossfading is used when transitioning from a substituted bad frame to a decoded good frame.

7.6.4.4.2 Synthesis model

Two alternative synthesis models are available for generation of the CLDFB TF tiles of a given CLDFB band k of the substitution frame.

Sinusoidal model

The one is a sinusoidal model, capable of extending a damped sinusoid in the given CLDFB band of the preceding frame into the substitution frame. This model is especially useful for relatively tonal signals. According to this model, a complex sinusoid in the given band k of the preceding frame $x(n), n = 0 \ldots L-1$ can be expressed with the following equation:

$$x_k(n) = a_{0,k}\ \bar{\kappa}_k^n\ e^{jn\overline{\Delta\phi}_k} , \qquad (7.6\text{-}60)$$

wherein $a_{0,k}$ is a complex valued start parameter equal to the value of the first CLDFB sample (i.e., $n = 0$) in that frame, $\bar{\kappa}_k$ is a by-sample magnitude evolution parameter, and $\overline{\Delta\phi}_k$ is a per-sample phase evolution parameter. This leads to the following recursive formulation:

$$x_k(n) = x_k(n-1)\,\bar{\kappa}_k\, e^{j\overline{\Delta\phi}_k}, n > 1 \quad (7.6\text{-}61)$$

with $x(0) = a_0$.

To account for the real-world case that that the assumed sinusoid is not a clean sinusoid with strict by-sample phase evolution by $\overline{\Delta\phi}_k$, additive phase perturbation term is $\delta\phi_k(n)$ is used to model random phase deviations.

Based on these model assumptions, the CLDFB samples of the given band k of the substitution frame to be generated $y_k(n), n = 0 \ldots L$ is obtained as

$$y_k(n) = b_k(n)y_k(n-1)\,, \quad (7.6\text{-}62)$$

where $y_{-1,k}$ is a complex start value and

$$b_k(n) = \bar{\kappa}_k e^{j\left(\overline{\Delta\phi}_k + \delta\phi_k(n)\right)} \quad (7.6\text{-}63)$$

In the sections below, it is described how $y_{-1,k}$, $\overline{\Delta\phi}_k$, $\delta\phi_k(n)$ and $\bar{\kappa}_k$ are calculated from the frame preceding the frame to be substituted.

Linear predictive model

The other model available for substitution of the CLDFB samples of the given CLDFB band k is a first-order linear predictor combined with repetition of the same-band CLDFB samples of the preceding frame. It is especially useful for noise-like signals potentially with weak tonal components.

The first-order predictor is expressed by the following equation:

$$\hat{x}_k(n) = \gamma_{1,k}x_k(n-1)\ , \quad (7.6\text{-}64)$$

wherein $\gamma_{1,k}$ is a complex coefficient.

Together with the CLDFB samples of the preceding frame $x_k(n), n = 0 \ldots L-1$, this leads to the following model for the generation of a given CLDFB band k of the substitution frame:

$$y_k(n) = y_{-1,k}\,\gamma_{1,k}^{n+1} + x_k(n)(1 - |\gamma_{1,k}|^{n+1}). \quad (7.6\text{-}65)$$

The factor $(1 - |\gamma_{1,k}|^{n+1})$ is a complementary weighting factor for the decreasing contribution of the linear predictor, whereby it is assumed that the magnitude of the predictor is less than 1. This is a cross-fade approach facilitating a smooth transition from the preceding frame to the substituted frame. Signal discontinuities often observed in frame-repetitive methods are effectively avoided with this technique.

In the sections below, it is described how $y_{-1,k}$ and $\gamma_{1,k}$ are calculated from the frame preceding the frame to be substituted.

7.6.4.4.3 Analysis and parameter estimation

To calculate the phase parameter $\overline{\Delta\phi}_k$, the phase perturbation parameter sequence $\delta\phi_k(n)$, the magnitude evolution parameter $\bar{\kappa}$ and input parameters for the tonality estimator, the CLDFB samples of the given band k of the preceding frame are first converted to magnitude/phase representation:

$$x_k(n) \rightarrow \left(\rho_k(n), \theta_k(n)\right), n = 0 \ldots L-1\,. \quad (7.6\text{-}66)$$

The phases are obtained using the atan2 function, which returns wrapped phase values. These phase values are subsequently unwrapped, yielding a sequence of unwrapped phases $\breve{\theta}_k(n)$. After that, per-sample phase differences are calculated according to

$$\Delta\theta_k(n) = \breve{\theta}_k(n+1) - \breve{\theta}_k(n),\ n = 0 \ldots L-2\,. \quad (7.6\text{-}67)$$

After that the mean value $\overline{\Delta\phi}_k$ and standard deviation $\sigma_{\delta\phi,k}$ of the by-sample phase differences is calculated for the sequence $\Delta\theta_k(n)$. The latter will be used as an input parameter of the tonality estimator. A sequence of phase perturbation parameters is calculated by subtracting the mean value $\overline{\Delta\phi}_k$ from the sequence of per-sample phase differences:

$$\delta\phi(n) = \begin{cases} \Delta\theta(n) - \overline{\Delta\phi}_k\,, & n = 0 \ldots L-2 \\ \Delta\theta(n-L+1) - \overline{\Delta\phi}_k\,, & L-1 \le n \le L+1 \end{cases} . \quad (7.6\text{-}68)$$

Note: Phase perturbation parameters for $L-1 < n \le L+1$ are calculated for application of the sinusoidal model in a cross-fade region spanning 2 time slots length following the current substitution frame.

The magnitude evolution parameter $\bar{\kappa}_k$ is calculated in a similar manner. This parameter is calculated as mean over the sequence of $L-1$ per-sample magnitude ratios $\frac{\rho_k(n+1)}{\rho_k(n)}, n = 0 \ldots L-2$, of the preceding frame. To avoid modelling exponentially increasing magnitudes, $\bar{\kappa}_k$ is limited to 1 as maximum value.

In addition, the standard deviation $\sigma_{\kappa,k}$ of that sequence is calculated to be used as a further input parameter of the tonality estimator.

The start value for the sinusoidal model $y_{-1,k}$ for a given CLDFB band k is subsequently calculated as the average of the last CLDFB sample of the preceding frame and the second last CLDFB sample with applied sinusoidal model, i.e.,

$$y_{-1,k} = \frac{1}{2}\left(\bar{\kappa}_k e^{j\overline{\Delta\phi}_k} x_k(L-2) + x_k(L-1)\right) . \quad (7.6\text{-}69)$$

The averaging yields a smoothed start value where potential statistical variations are reduced.

The start value for the substitution with the linear predictive model is the last CLDFB sample of the preceding frame, i.e.,

$$y_{-1,k} = x_k(L-1) . \quad (7.6\text{-}70)$$

The complex predictor coefficient of the first-order linear predictive model is calculated following the Levinson-Durbin method based on the autocorrelation sequence $r_{xx,k}(n), n = 0{,}1$ of the (rectangularly windowed) CLDFB band samples $x_k(n)$ of the preceding frame. The first-order predictor coefficient is obtained as:

$$\gamma_{1,k} = \frac{r_{xx,k}(1)}{r_{xx,k}(0)} . \quad (7.6\text{-}71)$$

7.6.4.4.4 Tonality determination

Tonality determination is made separately for each CLDFB band k based on the corresponding standard deviation values $\sigma_{\delta\phi,k}$ and $\sigma_{\kappa,k}$. A CLDFB band of the preceding frame is declared tonal in case $\sigma_{\delta\phi,k} < \pi/6$ or $\sigma_{\kappa,k} < 0.1$. Otherwise, it is declared noise-like.

7.6.4.4.5 Sinusoidal extension

In case a CLDFB band k of the preceding frame is tonal, the sinusoidal model is used to generate the corresponding band of the substitution frame.

Using the start value $y_{-1,k}$ according to equation 7.6-69 and model parameters $\overline{\Delta\phi}_k$, $\delta\phi_k(n)$ and $\bar{\kappa}_k$ for the given CLDFB band k, the substitution frame band is generated according to equations 7.6-62 and 7.6-63.

7.6.4.4.6 Predictive extension

In case a CLDFB band of the preceding frame is noise-like, the linear predictive model is used to generate the corresponding band k of the substitution frame.

Using the start value $y_{-1,k}$ according to equation 7.6-70, the complex first-order coefficient $\gamma_{1,k}$ for the given CLDFB band k and the CLDFB band samples of the previous frame $x_k(n), n = 0 \ldots L-1$, the substitution frame band is generated according to equation 7.6-65.

7.6.4.4.7 Cross-fade

Cross-fade is used to avoid signal discontinuities at the boundary from a substituted frame to the next frame. The model-based technique described above ensures that such discontinuities to a subsequent substitution frame do not occur. However, when a good frame (marked with BFI flag=0) follows a bad frame, a special cross-fading technique is

applied. For that case, during substitution processing of the bad frame, cross-fade to a potential good frame following is prepared as follows:

In case of band k determined tonal, two extra CLDFB band samples are calculated beyond the end of the current frame according to the method described in clause 7.6.4.4.5 and equations 7.6-62 and 7.6-63 for sample indexes $n = L$ and $L + 1$. These samples are multiplied with fading weights of 2/3 for $y_k(L)$ and, respectively, 1/3 for $y_k(L + 1)$ and stored instead of the original samples. The respective complementary weights of 1/3 and 2/3 are stored in a buffer for frame onset weights $\beta_k(n), n = 0, 1$ for each CLDFB band k.

In case of band k determined noise-like, two extra CLDFB band samples are calculated beyond the end of the current frame according to the method described in clause 7.6.4.4.6 and equation 7.6-65 for sample indexes $n = L$ and $L + 1$. These samples are fading out exponentially with the magnitude of the predictor coefficient $\gamma_{1,k}$. Accordingly, no specific additional fading weight is applied. The respective complementary weights of $1 - |\gamma_{1,k}|$ and, $1 - |\gamma_{1,k}|^2$ are stored in the buffer for frame onset weights $\beta(n), n = 0, 1$ for each CLDFB band.

Upon the reception of a good frame following a bad frame, the first two samples (in each CLDFB band) are replaced by cross-faded samples while the other (later) samples are retained. This is illustrated by the following equation that is carried out for each CLDFB band k:

$$x_k(n) = \begin{cases} x_k(n)\beta_k(n) + y_k(L + n), n = 0, 1 \\ x_k(n) \qquad\qquad\quad , n \geq 2 \end{cases} . \tag{7.6-72}$$

7.6.4.4.8 Burst-loss handling

In case of burst frame loss with more than 10 frames in a row, the substitution frames generated according to the above-described technique are increasingly attenuated with 3 dB per additional bad frame. After 30 bad frames in a row, the substitution frames are completely zeroed.

7.6.5 LC3plus coded intermediate split renderer binaural audio format

7.6.5.1 Introduction (Informative)

The Low Complexity Communication Codec plus (LC3plus) [28] has been designed as the low complexity counterpart of EVS [REF] in order to make SWB also available on computationally constrained terminals such as VoIP, DECT [30] or wireless headsets. The codec allows perfect interoperability between mobile and other networks by means of transcoding [29][30] and fits complexity wise very well to the requirements of low power embedded devices. Due to the codec's flexible design the applications are not limited to voice services but can be extended to high quality music streaming as well [29]. The LC3plus codec can also be configured to be interoperable with LC3 coding as used in the Basic Audio Profile for Bluetooth Low Energy audio [31].

The application of LC3plus for coding the intermediate split renderer binaural audio format is described in the following clauses.

7.6.5.2 Overview

The LC3plus specification includes a full algorithmic description of both the encoder and decoder. It includes reference fixed-point and floating-point ANSI C source code (clauses 4.3 and 6 of [28]) and conformance test procedures (clause 4.4 of [28]). The entire codec as specified in [28] is included in the present document.

The codec can operate at highly flexible modes, including operation at multiple bandwidths, multiple frame durations and at any byte aligned bit rate (clause 5.1 of [28]). It includes the support of a high-performance Packet Loss Concealment (PLC) (see clause 5.6 of [28]) for all kinds of audio signals. LC3plus comes with an RTP Payload Format for transmission over IP/UDP, defined in Annex B of [28].

The following clauses provide additional details and information specific to the application as split renderer binaural audio format.

7.6.5.3 Encoder

The encoding process for LC3plus follows the procedures of clause 5.3 of [28]. To make full use of the available bits remaining after metadata encoding (clause 7.6.3), the LC3plus encoder may change the size of a compressed audio frame in a seamless manner, following the procedures of clause 5.7 of [28].

7.6.5.4 Decoder

The decoding process for LC3plus follows the procedures of clause 5.4 of [28].

7.6.5.5 Frame Structure

The frame structure of LC3plus follows the procedures of clause 5.5 of [28]. Multiple LC3plus 5ms or 10ms frame data blocks are used for the split renderer binaural audio format, following the procedures of Annex B using the payload structure as defined in B.2.4 of [28].

7.6.5.6 Packet Loss Concealment

The packet loss concealment for LC3plus follows the procedures of clause 5.6 of [28].

7.6.5.7 LC3 interoperable mode

The LC3plus codec can also be configured to be interoperable with LC3 coding as used in the Basic Audio Profile for Bluetooth Low Energy audio using the configurations 48_2, 48_4 and 48_6 as described in Table 3.5 of [31]. In these configurations the bandwidth is FB, high resolution mode is disabled, error protection is disabled, and the frame duration is 10 ms.

To generate 48_6 compatible LC3plus frames for the 256000 bps split rendering configuration the bit allocation as as defined in Table 7.6-24 shall be followed. The Frame Data Length Request (FDLR) (see clause B.2.5 of [28]) inside the LC3plus Payload Header shall use FDLR1 (1 octet) only and indicate NO_REQ.

NOTE: For this configuration, the overhead of the LC3plus payload format amounts to 40 bits per split rendering frame. The LC3plus frame data block is followed by 40 bits of zero padding, to fill up the 2560 bits of the split rendering frame. This leaves 2480 bits for the LC3plus frame data block, which results in 1240 bits per LC3plus frame data. This amounts to an LC3plus per channel bit rate of 124000 bps, which matches the 48_6 configuration defined in BAP Table 3.5.

7.6.6 Split post-rendering

7.6.6.1 Overview

The split rendering post-renderer performs lightweight audio processing with either pose correction (1-3 DOF) around multiple rotation axes or direct decoding (0 DOF). The post-renderer receives the coded bitstream $b2$ from the pre-renderer as shown in Figure 7.6-1 and 7.6-2. The bitstream includes LCLD, LC3plus or PCM audio coded bits and coded metadata bits for pose correction (1-3 DOF). The audio coded bits are decoded to obtain reference binaural signal associated with reference head pose P'. The LCLD decoder outputs a reference binaural signal in CLDFB domain, whereas LC3plus or PCM outputs are in the time domain.

If pose correction metadata was transmitted, time domain outputs are converted to CLDFB domain by applying CLDFB analysis. Subsequently the metadata is decoded as in clause 7.6.6.2, and then post rendering is performed, depending on DOF either as described in clause 7.6.6.2 or 7.6.6.3.

If a backchannel exists from post-renderer to pre-renderer, the latest head pose P is encoded and transmitted to the pre-renderer.

7.6.6.2 Post rendering with pose correction

7.6.6.2.1 Metadata decoding

The metadata bits are decoded (as described in clause 7.6.3.4) to obtain the CLDFB banded metadata that enables the reconstruction of the set of binaural signals (BIN_n) associated with the set of probing poses (P_n) from the reference binaural signal. The metadata associated with the set of probing poses (P_n) represents deviation from the reference head pose (P') by rotation around the multiple rotational axes.

The reference head pose P' is also decoded as part of metadata decoding, as described in clause 7.6.3.4. The probing poses (P_n) are then computed by adding the offsets given in Table 7.6-1 to P'. The post-render obtains the latest head pose (P) associated with the user. Then axis-specific metadata for each rotation axis is computed by selecting the probing pose that is closest to pose P along the given rotation axis and performing interpolation (or extrapolation) on the metadata associated with the selected probing pose. The interpolation is done based on the difference between the latest head pose P and reference head pose P' along the given rotation axis.

The interpolated axis-specific metadata for each rotation axis is combined into one 2x2 pose correction matrix per frequency band. The combined matrix is applied to the reference binaural signal to obtain the output binaural signal corresponding to the latest head pose P. The time domain output is generated by performing CLDFB synthesis on the CLDFB domain output binaural signal.

The metadata bits in bitstream $b2$ that correspond to roll axis or yaw axis (if the ILD flag was enabled at the pre-renderer) are decoded to obtain a complex-valued matrix M_{P_s} in low frequency bands (0 to 1600 Hz) that enables reconstruction of both the magnitude and phase of the set of binaural signals (BIN_n) from the reference binaural signal, and a real valued diagonal matrix D_{P_s} in high frequency bands that enables reconstruction of only the magnitude of the set of binaural signals (BIN_n) from the reference binaural signal. The output binaural signal in the low frequency bands is generated by applying the interpolated or extrapolated matrix M_{P_s} to the corresponding lowfrequency bands of the reference binaural signal. Similarly, the output binaural signal in the high frequency bands is generated by applying the interpolated or extrapolated real-valued diagonal matrix D_{P_s} to the corresponding high frequency bands of the reference binaural signal. Here, the interpolation or extrapolation is done based on the difference between the latest head pose P and reference head pose P' along the given rotation axis, as described in clause 7.6.6.2.2. The output binaural signal is generated by combining the output binaural signal in low frequency bands and high frequency bands.

7.6.6.2.2 Metadata interpolation or extrapolation

The metadata interpolation is done using linear interpolation method and the same approach is used for every rotation axis. First, the probing pose that is closest to the latest head pose P along a given rotation axis is selected. Then metadata parameters are interpolated as follows:

$$\mathrm{M}_p = \sum_{j=0}^{j=1} a_j M_{p_j} , \quad (7.6\text{-}21)$$

where M_{p_0} is an identity matrix, M_{p_1} is the metadata matrix corresponding to the closest probing pose P_1 and a_j are interpolation coefficients as given below:

$$a_0 = 1 - \frac{d}{offset}, a_1 = \frac{d}{offset} ,$$

where $offset$ is the deviation between the closest probing pose P_1 and reference pose P' along the said rotation axis such that $offset = P_1 - P'$, d is the deviation between the latest head pose P and reference pose P' along the said rotation axis such that $d = P - P'$.

When absolute value of d is greater than absolute value of $offset$, then extrapolation is performed. In this case, M_{p_1} is an identity matrix, M_{p_0} is the metadata matrix corresponding to the closest probing pose P_0 and a_j are extrapolation coefficients as given below:

$$a_0 = 1 - \frac{\sin(d)}{\sin(offset)}, a_1 = \frac{\sin(d)}{\sin(offset)} ,$$

where $offset$ is the deviation between the closest probing pose P_0 and reference pose P' along the said rotation axis such that $offset = P' - P_0$, d is the deviation between the latest head pose P and closest probing pose P_0 along the said rotation axis such that $d = P_0 - P$.

7.6.6.2.3 Matrix mixing

The interpolated or extrapolated matrices $M_{\mathrm{yaw,p}}$, $M_{\mathrm{roll,p}}$ and $H_{\mathrm{pitch,p}}$ are then combined into one 2x2 pose correction matrix per frequency band as shown below:

$$M_{[2x2]} = M_{\mathrm{roll,p}} H_{\mathrm{pitch,p}} M_{\mathrm{yaw,p}} . \quad (7.6\text{-}22)$$

The output binaural signal in each CLDFB band is computed as

$$BIN_{[2x1]}^{out} = M_{[2x2]} BIN_{[2x1]}^{ref} , \quad (7.6\text{-}23)$$

where, $BIN_{[2x1]}^{ref}$ is the reference binaural signal corresponding to reference head pose P'.

7.6.6.3 Post rendering in 0 DOF mode

In 0 DOF mode or no pose correction mode, the split rendering bitstream received by the post-renderer includes LCLD or LC3plus coded bits which are decoded to obtain the reference binaural signal associated with the reference head pose P'. The LC3plus decoder outputs in time domain, whereas the LCLD decoder outputs reference binaural signal in CLDFB domain which is then converted to time domain by applying CLDFB synthesis. The time domain reference binaural signal is the output of the post-renderer.

7.6.7 Bit allocation for Split rendering

Detailed bit allocation principles at different bitrates of the Split rendering operation are provided in Table 7.6-21, Table 7.6-22, Table 7.6-23, Table 7.6-24, Table 7.6-25 and Table 7.6-26.

Table 7.6-21: Bit allocation for split rendering with 1, 2 and 3 DOF pose correction with LCLD or LC3plus codecs

1.1.535 Description	1.1.536 384 kbps	1.1.537 512 kbps	1.1.538 768 kbps
1.1.539 Total number of bits per 20ms frame	1.1.540 7680	1.1.541 10240	1.1.542 15360
1.1.543 DOF (degree of freedom)	1.1.544 2	1.1.545 2	1.1.546 2
1.1.547 HQ mode	1.1.548 1	1.1.549 1	1.1.550 1
1.1.551 Rotation axes	1.1.552 2 (if 1, 2 DoF), 0 otherwise	1.1.553 2 (if 1, 2 DoF), 0 otherwise	1.1.554 2 (if 1, 2 DoF), 0 otherwise
1.1.555 HF ILD flag	1.1.556 1	1.1.557 1	1.1.558 1
1.1.559 Reference Pose P' (yaw, pitch and roll angle)	1.1.560 27	1.1.561 27	1.1.562 27
1.1.563 Coding strategy	1.1.564 1	1.1.565 1	1.1.566 1
1.1.567 Quantization strategy	1.1.568 2	1.1.569 2	1.1.570 2
1.1.571 Metadata bits	1.1.572 Variable	1.1.573 variable	1.1.574 variable
1.1.575 LCLD or LC3plus bits	1.1.576 Variable	1.1.577 variable	1.1.578 variable

In case of LC3plus, LC3plus bits are prepended with byte-alignment 0-bits to the next closest octet boundary. If the number of bits available for audio coding exceeds the maximum frame size of LC3plus, the leftover bytes in the LC3plus section are filled up with padding 0-bytes.

Table 7.6-22: Bit allocation for split rendering with 0 DOF (no pose correction) with 5ms split rendering frame size

1.1.579 **Description**	**1.1.580 256 kbps**	**1.1.581 384 kbps**	**1.1.582 512 kbps**
1.1.583 Total number of bits per 5ms frame	1.1.584 1280	1.1.585 1920	1.1.586 2560
1.1.587 LCLD or LC3plus bits	1.1.588 1280	1.1.589 1920	1.1.590 2560

Table 7.6-23: Bit allocation for split rendering with 0 DOF (no pose correction) with 10ms split rendering frame size using LCLD

1.1.591 Description	**1.1.592 256 kbps**	**1.1.593 384 kbps**	**1.1.594 512 kbps**
1.1.595 Total number of bits per 10ms frame	1.1.596 2560	1.1.597 3840	1.1.598 5120
1.1.599 LCLD	1.1.600 2560	1.1.601 3840	1.1.602 5120

Table 7.6-24: Bit allocation for split rendering with 0 DOF (no pose correction) with 10ms split rendering frame size using LC3plus

1.1.603 Description	1.1.604 256 kbps	1.1.605 384 kbps	1.1.606 512 kbps
1.1.607 Total number of bits per 10ms frame	1.1.608 2560	1.1.609 3840	1.1.610 5120
1.1.611 LC3plus bits	1.1.612 2520	1.1.613 3840	1.1.614 5120
1.1.615 Zero bits	1.1.616 40	1.1.617 0	1.1.618 0

Table 7.6-25: Bit allocation for split rendering with 0 DOF (no pose correction) with 20ms split rendering frame size

1.1.619 Description	1.1.620 256 kbps	1.1.621 384 kbps	1.1.622 512 kbps
1.1.623 Total number of bits per 20ms frame	1.1.624 5120	1.1.625 7680	1.1.626 10240
1.1.627 LCLD or LC3plus bits	1.1.628 5120	1.1.629 7680	1.1.630 10240

Table 7.6-26: Bit allocation for split rendering with 1, 2 and 3 DOF pose correction with PCM binaural output

1.1.631 **Description**	**1.1.632 Bitrate >= 34 kbps**
1.1.633 Total number of bits per 20ms frame	1.1.634 >= 680
1.1.635 DOF (degree of freedom)	1.1.636 2
1.1.637 HQ mode	1.1.638 1
1.1.639 Rotation axes	1.1.640 2 (if 1, 2 DoF), 0 otherwise
1.1.641 HF ILD flag	1.1.642 1
1.1.643 Reference Pose P' (yaw, pitch and roll angle)	1.1.644 27
1.1.645 Coding strategy	1.1.646 1
1.1.647 Quantization strategy	1.1.648 2
1.1.649 Metadata bits	1.1.650 Remaining bits

7.6.8 Interface for Split rendering

Split renderer and its interface provide support to ISAR codec design constraints. The details of the split rendering library API are provided in [7] for the fixed-point code and [12] for the floating-point code. The details of the split rendering library API are provided in [9].

8 Description of the transmitted parameter indices

8.1 Bit allocation overview

Each active IVAS frame signals the IVAS format using the first 2-4 bits of the IVAS bitstream. This enables for the IVAS decoder operation to correctly detect the IVAS format and associated decoding functionality. A summary of active frame IVAS format signalling is provided in Table 5.1-1. The bit allocation principles in each IVAS operation are described in the following clauses.

In case of SID frames during DTX operation, the IVAS format is signalled using the first 2 bits of the IVAS bitstream for operations where DTX is supported. The IVAS format signalling in SID frames is presented in Table 5.1-2.

8.2 Bit allocation for stereo audio

8.2.1 Bit allocation for stereo in active frames

Active frame signalling for IVAS stereo operation (STEREO) is summarized in Table 8.2-1.

Table 8.2-1: STEREO frame signalling, active frames

IVAS format	configuration	number of bits	Value
STEREO	-	2	0

8.2.2 Bit allocation for stereo in SID frames

SID frame signalling for IVAS stereo operation (STEREO) is summarized in Table 8.2-2.

Table 8.2-2: STEREO frame signalling, SID frames

IVAS format	configuration	number of bits	Value
STEREO	Unified stereo	2	0x0
	MDCT stereo	2	0x1

8.3 Bit allocation for scene-based audio (SBA)

8.3.1 Bit allocation for SBA in active frames

Active frame signalling for IVAS SBA operation (SBA) is summarized in Table 8.3-1.

Table 8.3-1: SBA frame signalling, active frames

IVAS format	configuration	number of bits	Value
SBA	-	3	6

Detailed bit allocation principles at different bitrates of the SBA operation are provided in Table 8.3-2, Table 8.3-3, Table 8.3-4, and Table 8.3-5.

Table 8.3-2: Bit allocation at 13.2, 16.4, 24.4 and 32 kbps

Description	Ordering of bits	13.2 kbps	16.4 kbps	24.4 kbps	32 kbps
total bits	Forward ordering of bits	264	324	488	640
IVAS common header (format)		3	3	3	3
SBA header – planar mode		1	1	1	1
SBA header – ambisonics order		2	2	2	2
Core-coder - SCE		variable	variable	variable	variable
AGC gain bits	Reverse ordering of bits	0 or 2	0 or 2	0 or 2	0 or 2
AGC flag		1	1	1	1
SPAR metadata (PR, CP and D coefficients)		variable	variable	variable	variable
SPAR coding strategy		3	3	3	3
SPAR quantization strategy		2	2	2	2
DirAC metadata		variable, max 28	variable, max 32	variable, max 62	variable, max129
Metadata active/inactive mode flag		1	1	1	1

Table 8.3-3: Bit allocation at 48, 64 and 80 kbps

Description	Ordering of bits	48 kbps	64 kbps	80 kbps
total bits	Forward ordering of bits	960	1280	1600
IVAS common header (format)		3	3	3
SBA header – planar mode		1	1	1
SBA header – ambisonics order		2	2	2
Core-coder - CPE		variable	variable	variable
SPAR metadata (PR, CP and D coefficients)	Reverse ordering of bits	variable	variable	variable
SPAR coding strategy		3	3	3
SPAR quantization strategy		2	2	2
active W residual index		0 or 1	0 or 1	0 or 1
Dynamic active W flag		1	1	1
DirAC metadata		variable, max 144	variable, max 120	variable, max 120
Metadata active/inactive mode flag		1	1	1

Table 8.3-4: Bit allocation at 96, 128, 160 and 192 kbps

Description	Ordering of bits	96 kbps	128 kbps	160 kbps	192 kbps
total bits	Forward ordering of bits	1920	2560	3200	3840
IVAS common header (format)		3	3	3	3
SBA header – planar mode		1	1	1	1
SBA header – ambisonics order		2	2	2	2
Core-coder - MCT		variable	variable	variable	variable
SPAR metadata (PR, CP and D coefficients)	Reverse ordering of bits	variable	variable	variable	variable
SPAR coding strategy		3	3	3	3
SPAR quantization strategy		2	2	2	2
Dynamic active W flag		1	1	1	1
DirAC metadata		variable, max 120	variabel, max 150	variable	variable
reserved		1	1	1	1

Table 8.3-5: Bit allocation at 256, 384 and 512 kbps

Description	Ordering of bits	256 kbps		384 kbps	512 kbps
	Forward ordering of bits	**FOA**	**HOA2, HOA3**		
total bits		5120	5120	7680	10240
IVAS common header (format)		3	3	3	3
SBA header – planar mode		1	1	1	1
SBA header – ambisonics order		2	2	2	2
Core-coder - MCT		variable	variable	variable	variable
PCA metadata	Reverse ordering of bits	0 or TBD	0	0	0
PCA flag		1	0	0	0
SPAR metadata (PR, CP and D coefficients)		variable	variable	variable	variable
SPAR coding strategy		3	3	3	3
SPAR quantization strategy		1	1	1	1
reserved		1	1	1	1
DirAC metadata		variable	variable	variable	variable
reserved		1	1	1	1

8.3.2 Bit allocation for SBA in SID frames

SID frame signalling for IVAS SBA operation (SBA) is summarized in Table 8.3-6.

Table 8.3-6: SBA frame signalling, SID frames

IVAS format	configuration	number of bits	Value
SBA	1 TC	2	0x5
	2 TCs	2	0x6
	3+ TCs	N/A	N/A

Detailed bit allocation principles for SID frames of the SBA operation are provided in Table 8.3-7.

Table 8.3-7: Bit allocation for SID frames

Description	Ordering of bits	Number of bits
total bits	Forward ordering of bits	104
SID format bits		3
Core-coder – SCE/CPE		48
reserved	Reverse ordering of bits	1
SPAR metadata (PR, CP and D coefficients)		36
DirAC metadata		15
Metadata active/inactive mode flag		1

8.4 Bit allocation for metadata-assisted spatial audio (MASA)

8.4.1 Bit allocation for MASA in active frames

Active frame signalling for IVAS MASA operation (MASA) is summarized in Table 8.4-1.

Table 8.4-1: MASA frame signalling, active frames

IVAS format	configuration	number of bits	Value
MASA	-	3	7

Detailed bit allocation principles at different bitrates of the MASA operation are provided in Table 8.4-1A, Table 8.4-1B, and Table 8.4-1C.

Table 8.4-1A: Bit allocation at 13.2, 16.4, 24.4, and 32 kbps

Description		Ordering of bits	13.2 kbps	16.4 kbps	24.4 kbps	32 kbps
total bits		Forward ordering of bits	264	324	488	640
IVAS common header (format)			3	3	3	3
Core-coder – SCE/CPE			variable	variable	variable	variable
No. of transport channels		Reverse ordering of bits	1	1	1	1
Reserved			2	2	2	2
No. of spatial directions			1	1	1	1
Subframe mode (SF = 0 or 1)			1	1	1	1
Low bitrate mode	**1 subframe (SF=1)**		0	0	0	0
	4 subframes (SF=0)		1	1	1	1
MASA metadata	**LR mode**		variable, max 45-(1-SF)	variable, max 45-(1-SF)	variable, max 55-(1-SF)	variable, max 65-(1-SF)
	Normal mode		variable, max 45-(1-SF)	variable, max 55-(1-SF)	variable, max 65-(1-SF)	variable, max 80-(1-SF)

Table 8.4-1B: Bit allocation at 48, 64, 80, and 96 kbps

Description	Ordering of bits	48 kbps	64 kbps	80 kbps	96 kbps
total bits	Forward ordering of bits	960	1280	1600	1920
IVAS common header (format)		3	3	3	3
Core-coder – SCE/CPE		variable	variable	Variable	variable
No. of transport channels	Reverse ordering of bits	1	1	1	1
Reserved		2	2	2	2
No. of spatial directions		1	1	1	1
Subframe mode		1	1	1	1
MASA metadata		variable, max 135	variable, max 175	variable, max 215	variable, max 251

Table 8.4-1C: Bit allocation at 128, 160, 192, 256, 384, and 512 kbps

Description	Ordering of bits	128 kbps	160 kbps	192 kbps	256 kbps	384 kbps	512 kbps
total bits	Forward ordering of bits	2560	3200	3840	5210	7680	10240
IVAS common header (format)		3	3	3	3	3	3
Core-coder – SCE/CPE		variable	variable	variable	variable	variable	variable
No. of transport channels	Reverse ordering of bits	1	1	1	1	1	1
Reserved		2	2	2	2	2	2
No. of spatial directions		1	1	1	1	1	1
Subframe mode		1	1	1	1	1	1
MASA metadata		variable, max 325	variable, max 427	variable, max 523	variable, max 827	variable	variable

8.4.2 Bit allocation for MASA in SID frames

SID frame signalling for IVAS MASA operation (MASA) is summarized in Table 8.4-2 and the corresponding bitstream description in Table 8.4-3.

Table 8.4-2: MASA frame signalling, SID frames

IVAS format	configuration	number of bits	Value
MASA	-	2	0x3
	-	2	0x7

Table 8.4-3: Bit allocation for MASA in SID frames

Description	Ordering of bits	Number of bits
total bits	Forward ordering of bits	104
SID format bits		3
Core-coder – SCE/CPE		48
MASA metadata	Reverse ordering of bits	53

8.4.2 Bit allocation for MASA in SID frames

SID frame signalling for IVAS MASA operation (MASA) is summarized in Table 8.4-2.

Table 8.4-2: MASA frame signalling, SID frames

IVAS format	configuration	number of bits	Value
MASA	-	2	0x3
	-	2	0x7

8.5 Bit allocation for object-based audio (ISM)

8.5.1 Bit allocation for ISM in active frames

Active frame signalling for IVAS ISM operation (ISM) is summarized in Table 8.5-1.

Table 8.5-1: ISM frame signalling, active frames

IVAS format	configuration	number of bits	Value
ISM	< 24.4 kbps	2	2
	≥ 24.4 kbps	3	4

8.5.2 Bit allocation for ISM in SID frames

SID frame signalling for IVAS ISM operation (ISM) is summarized in Table 8.5-2.

Table 8.5-2: ISM frame signalling, SID frames

IVAS format	configuration	number of bits	Value
ISM	-	2	0x2

8.6 Bit allocation for multi-channel audio (MC)

Active frame signalling for IVAS MC operation (MC) is summarized in Table 8.6-1. DTX operation is not supported with multi-channel (MC) inputs.

Table 8.6-1: MC frame signalling, active frames

IVAS format	configuration	number of bits	Value
MC	-	2	1

8.7 Bit allocation for combined Object-based audio and SBA (OSBA)

Active frame signalling for IVAS OSBA operation (OSBA) is summarized in Table 8.7-1. DTX operation is not supported with OSBA inputs.

Table 8.7-1: OSBA frame signalling, active frames

IVAS format	configuration	number of bits	Value
OSBA	< 24.4 kbps	3	6
	≥ 24.4 kbps	4	11

8.8 Bit allocation for combined Object-based audio and MASA (OMASA)

Active frame signalling for IVAS OMASA operation (OMASA) is summarized in Table 8.8-1. DTX operation is not supported with OMASA inputs.

Table 8.8-1: OMASA frame signalling, active frames

IVAS format	configuration	number of bits	Value
OMASA	Rend mode	3	7
	other modes	4	10

Detailed bit allocation principles for OMASA format in pre-rendering coding mode are presented in Tables 8.8-2 and 8.8-3 for different bitrates and different number of objects.

Table 8.8-2: Bit allocation for pre-rendering coding mode – part 1

Description		Ordering of bits	13.2 kbps 1 and 2 obj	16.4 kbps 1 and 2 obj	24.4 kbps 2 obj
total bits		Forward ordering of bits	264	324	488
IVAS common header (format)			3	3	3
Core-coder – CPE			variable	variable	variable
Additional info on number of objects		Reverse ordering of bits	1	1	1
Initial info on number of objects			2	2	2
No. of spatial directions			1	1	1
Subframe mode (SF = 0 or 1)			1	1	1
Low bitrate mode	**1 subframe (SF=1)**		0	0	0
	4 subframes (SF=0)		1	1	1
MASA metadata			variable, max 45 – (1-SF)	variable, max 45 – (1-SF)	variable, max 55 – (1-SF)

Table 8.8-3: Bit allocation for pre-rendering coding mode – part 2

Description		Ordering of bits	13.2 kbps 3 and 4 obj	16.4 kbps 3 and 4 obj	24.4 kbps 3 and 4 obj
total bits		Forward ordering of bits	264	324	488
IVAS common header (format)			3	3	3
Core-coder – CPE			variable	variable	variable
No. of spatial directions		Reverse ordering of bits	1	1	1
Info on number of objects			2	2	2
Subframe mode (SF = 0 or 1)			1	1	1
Low bitrate mode	**1 subframe (SF=1)**		0	0	0
	4 subframes (SF=0)		1	1	1
MASA metadata			variable, max 45-(1-SF)	variable, max 45-(1-SF)	variable, max 55-(1-SF)

Detailed bit allocation principles for OMASA format in one object with MASA representation coding mode are presented in Table 8.8-4 for the bitrates and corresponding number of objects for which the coding mode is used.

Table 8.8-4: Bit allocation for one object with MASA representation coding mode

Description		Ordering of bits	32 kbps 3 and 4 obj	48 kbps 3 and 4 obj
total bits		Forward ordering of bits	640	960
IVAS common header (format)			4	4
Separated object (SCE) with object metadata			variable	Variable
Core-coder – CPE			variable	Variable
Number of objects		Reverse ordering of bits	2	2
Separated object importance flags			2 or 4	2 or 4
Reserved MASA bits			2	2
No. of spatial directions			1	1
Subframe mode			1	1
Low bitrate mode	**1 subframe (LRSF)**		0	0
	4 subframes (LRSF)		1	0
MASA metadata			variable, max 52-LRSF	variable, max 62-SF

Detailed bit allocation principles for OMASA format in parametric one object coding mode are presented in Table 8.8-5 for the bitrates and corresponding number of objects for which the coding mode is used.

Table 8.8-5: Bit allocation for parametric one object coding mode

Description		Ordering of bits	32 kbbps 2 obj	64 kbps 3 and 4 obj	80 kbps 3 and 4 obj	96 kbps 4 obj
total bits		Forward ordering of bits	640	1280	1600	1920
IVAS common header (format)			4	4	4	4
Separated object (SCE) with object metadata			variable	variable	variable	variable
Core-coder – CPE			variable	variable	variable	variable
Number of objects		Reverse ordering of bits	2	2	2	2
Index of separated object			2	2	2	2
Separated object importance flags			2	2	2	2
Reserved MASA bits			2	2	2	2
No. of spatial directions			1	1	1	1
Subframe mode (SF = 0 or 1)			1	1	1	1
Low bitrate mode	**1 sub frame (SF=1)**		0	0	0	0
	4 sub frames (SF=0)		1	0	0	0
MASA to total energy ratios **ISM energy ratios** **ISM metadata** **MASA metadata**			variable, max 65-(1-SF)	variable, max 130	variable, max 170	variable, max 210

Detailed bit allocation principles for OMASA format in discrete coding mode are presented in Table 8.8-6, 8.8-7, and 8.8-8 for the bitrates and corresponding number of objects for which the coding mode is used.

Table 8.8-6: Bit allocation for discrete coding mode – part 1

Description		Ordering of bits	24 kbps *N_{obj} =1	32 kbps N_{obj}=1	48 kbps N_{obj} = 1, 2	64 kbps *N_{obj}=1,2
total bits		Forward ordering of bits	640	1280	1600	1920
IVAS common header (format)			4	4	4	4
ISM format data			variable	variable	variable	variable
Core-coder – CPE			variable	variable	variable	variable
Number of objects		Reverse ordering of bits	2*N_{obj}	2*N_{obj}	2*N_{obj}	2*N_{obj}
Objects importance flags			2	2	2	2
Reserved MASA bits			2	2	2	2
No. of spatial directions			1	1	1	1
Subframe mode (SF = 0 or 1)			1	1	1	1
Low bitrate mode	**1 subframe (SF = 1)**		0	0	0	0
	4 subframes (SF = 0)		1	1	1	0
MASA metadata			variable, max 42-(1-SF)	variable, max 52-(1-SF)	variable max 62-(1-SF)	variable, max 62-(1-SF)

Table 8.8-7: Bit allocation for discrete coding mode – part 2

Description		Ordering of bits	80 kbps N_{obj} = 1, 2	96 kbps N_{obj} = 1, 2, 3	128 kbps N_{obj} = 1, 2	128 kbps N_{obj} = 3, 4
total bits		Forward ordering of bits	640	1280	1600	1920
IVAS common header (format)			4	4	4	4
Separated object (SCE) with object metadata			variable	variable	variable	variable
Core-coder – CPE			variable	variable	variable	variable
Number of objects		Reverse ordering of bits	2	2	2	2
Objects importance flags			(2 or 4) *N_{obj}	(2 or 4)* N_{obj}	(2 or 4)* N_{obj}	(2 or 4)* N_{obj}
OMASA bitrate flag			0	0	0	1
Reserved MASA bits			2	2	2	2
No. of spatial directions			1	1	1	1
Subframe mode (SF)			1	1	1	1
Low bitrate mode	**1 subframe**		0	0	0	0
	4 subframes		0 or 1	0 or 1	0	0 or 1
MASA metadata			variable	variable	variable	variable

Table 8.8-8: Bit allocation for discrete coding mode – part 3

Description	Ordering of bits	160 kbps N_{obj} = 1..4	192 kbps N_{obj} = 1..4	256 kbps N_{obj} = 1..4	384 kbps N_{obj} = 1..4	512 kbps N_{obj} = 1.. 4
total bits	Forward ordering of bits	3200	3840	5210	7680	10240
IVAS common header (format)		4	4	4	4	4
Separated object (SCE) with object metadata		variable	variable	variable	variable	variable
Core-coder – CPE		variable	variable	variable	variable	variable
Number of objects	Reverse ordering of bits	2	2	2	2	2
Objects importance flags		(2 or 4) $*N_{obj}$	(2 or 4) $*N_{obj}$	(2 or 4)* N_{obj}	(2 or 4)* N_{obj}	(2 or 4)* N_{obj}
Reserved MASA bits		2	2	2	2	2
No. of spatial directions		1	1	1	1	1
Subframe mode (SF)		1	1	1	1	1
MASA metadata		variable	variable	variable	variable	variable

8.9 Bit allocation for EVS-compatible mono audio

Bit allocation for the EVS-compatible mono audio operation is the same as described in clause 7 of [3].

Annex A (normative): RTP Payload Format and SDP Parameters

A.1 Introduction

This annex describes a generic RTP payload format and SDP parameters for the Immersive Voice and Audio Services (IVAS) codec for mobile communication [6]. The IVAS RTP packets consist of the RTP header, and the IVAS payload. The IVAS payload consists of IVAS-specific payload header, frame data, and optionally processing information (PI) data.

IVAS is the immersive voice and audio extension of the Enhanced Voice Services (EVS) codec [2], fully incorporating the EVS codec.

A.2 Conventions, Definitions and Acronyms

A.2.1 Byte Order

The byte order used in this document is the network byte order, i.e., the most significant byte is transmitted first. The bit order is most significant bit first. This practice is presented in all figures as having the most significant bit located left-most on each line and indicated with the lowest number.

A.2.2 List of Acronyms

See clause 3.3 for the abbreviations.

A.3 Payload Format

A.3.1 Format Overview

The RTP Payload Format described in this document addresses the specific requirements of the IVAS codec. The format supports the transmission of IVAS Immersive mode frames or EVS coded frames with the following features:

- IVAS Immersive mode operation
  - WB, SWB and FB audio bandwidths, respectively 16, 32, and 48 kHz sampling rates
  - all immersive formats of the IVAS codec
    - 1-4 independent (mono) streams with meta data (ISM)
    - stereo (including binaural audio)
    - multi-channel in 5.1, 7.1, 5.1+2, 5.1+4, 7.1+4
    - scene-based audio (Ambisonics) up to order 3 (SBA)
    - metadata-assisted spatial audio (MASA)
    - combinations of ISM+MASA (OMASA) and ISM+SBA (OSBA)
  - bitrates ranging from 13.2 kbps to 512 kbps
- EVS operation

- supporting all EVS operation modes (mono) of the IVAS codec, including the EVS Primary and AMR-WB IO modes, using a payload syntax compatible to the header-full format defined in Annex A of [3] (with some limitations)

NOTE: The format does not support the compact format, present in Annex A of [3].

- NB, WB, SWB and FB audio, respectively 8, 16, 32, and 48 kHz sampling rates
- bitrates ranging from 5.9 (VBR) to 128 kbps

- 20 ms frame duration
- multiple frames per RTP payload
- Discontinuous Transmission (DTX)
- transmission of Processing Information (PI), i.e. PI data, in forward direction to support the rendering
- switching between EVS (mono) and IVAS (stereo and immersive) operation in the same payload type

A.3.2 RTP Header Usage

The format of the RTP header is specified in [r3]. This IVAS RTP payload format uses the fields of the RTP header in a manner consistent with the usages in [r3].

The assignment of the RTP payload type for IVAS is out of scope of this document. In most cases SDP would be used to signal the payload type for dynamic assignment.

The RTP clock rate for IVAS is 16000, regardless of the audio bandwidth. A clock rate of 16000 is also used for the AMR-WB [r5] and EVS codecs [3]; having a unique clock rate across all payload types of one media avoids the issues described in [r6].

The RTP timestamp defines the sampling instant (media time) of the first sample of the first IVAS frame in an RTP packet. The duration of one IVAS frame is 20 ms. Thus, the media time is increased for each successive IVAS frame of an RTP packet by 320 ticks. The RTP timestamp of a packet is used for the first PI data in the IVAS RTP payload. The timing of PI frames during DTX is explained in clause A.3.5.4.

The RTP header marker bit (M) shall be set to 1 for the first packet of a talk spurt, i.e. if the first frame-block carried in the RTP packet contains the frame first in a talkspurt. For all other RTP packets the marker bit shall be set to zero (M=0). This is the same usage as described in [r4].

A.3.3 Packet Payload Structure

A.3.3.1 General

The IVAS encoder generates encoded frames representing 20 ms of speech or audio data. The IVAS payload contains:

- (optional) E-bytes (including the CMR) for adaptation and indication of optional PI data section;
- one or more ToC(s) describing the IVAS audio frame(s) included in the payload;
- IVAS frame data block(s), representing 20 ms of speech or audio data (depending on ToC signaling), and;
- optional PI data section;

A.3.3.2 Format Description

An RTP payload comprises the IVAS payload, which consist of the IVAS-specific payload header followed by the frame data and optional PI data as shown in Figure A.3.3.2-1. The frame data consists of one or more IVAS or EVS coded frames (including NO_DATA, see A.3.3.3.2). The optional PI data section can be considered as additional metadata to support the rendering.There may be zero-padding bits in addition at the end of the payload. Padding bits shall be discarded by the receiver.

NOTE: The purpose of padding is that in the case of EVS AMR-WB IO frames, payload data may need to be octet-aligned using zero-padding bits at the end of the payload. EVS Primary frames are by definition octet-aligned (see clause A.2.2.1.4.1 of [3]).

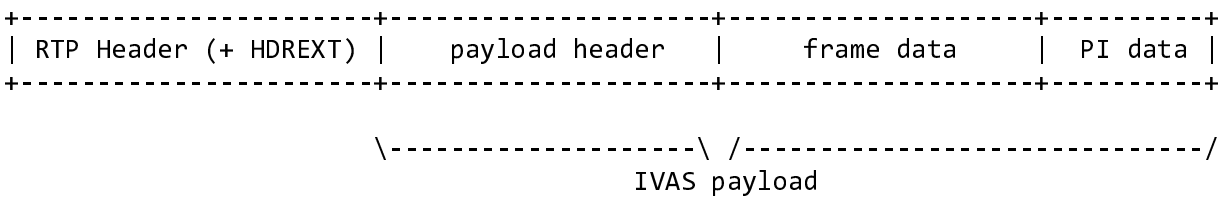


Figure A.3.3.2-1: RTP Header with IVAS payload structure

A.3.3.3 Payload Header

A.3.3.3.1 General

The IVAS payload header consists of Table of Contents (ToC) bytes and Extra (E) bytes, defined in clauses A.3.3.3.2 and A.3.3.3.3, respectively. The first bit of each as header byte is the Header Type identification bit (H) to identify whether a header byte is a ToC or E byte. If the H bit is set to 0, the corresponding byte is a ToC byte, and if set to 1, the corresponding byte is an E byte. The second bit of a ToC byte is the Following (F) bit (see clause A.3.3.3.2), which if set to 1 indicates that another header byte is following. The last header byte shall be a ToC byte and have the F bit set to 0.

The general structure of a header byte is shown in figure A.3.3.3.1-1.

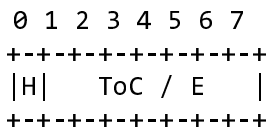


Figure A.3.3.3.1-1: Generic structure of a payload header byte.

H (1 bit): Header Type identification bit. For a ToC byte this is set to 0, for an E byte this is set to 1.

A.3.3.3.2 ToC byte

The ToC bytes define the content of the frame data in the IVAS payload following the IVAS payload header. For each IVAS or EVS frame and for each NO_DATA frame (i.e. a frame that has zero size frame data) in the payload there shall be one ToC byte to signal the IVAS mode and bit rate. ToC bytes and the respective frame data shall be in the same order.

The Table of Content (ToC) byte structure is an extension of the ToC byte structure defined in clause A.2.2.1.2 in [3]. In the EVS payload format in [3] a code point in the ToC byte (see Figure A.5 in [3]) for extensions has been reserved, the "Unused" bit. In the present document this "Unused" bit of the Frame type index bits is activated and called "IVAS indicator" to distinguish EVS and IVAS frame data. The specific ToC structure for an IVAS frame is shown in Figure A.3.3.3.2-1.

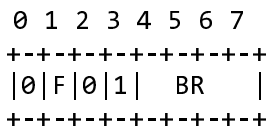


Figure A.3.3.3.2-1: Table of Content (ToC) byte structure for an IVAS frame.

F (1 bit): If set to 1, the bit indicates that the header byte is followed by another header byte. If set to 0, the bit indicates that this header byte is the last one in this payload and no further header bytes follows this entry.

BR (4 bits): Bit rate index as defined in Table A.3.3.3.2-1 .

Table A.3.3.3.2-1: Frame Type index when EVS mode bit = 0 and "Unused"/IVAS indicator bit = 1

EVS/IVAS mode bit (1 bit)	IVAS indicator (1 bit)	IVAS bit rate	Indicated IVAS mode and bit rate
0	1	0000	IVAS 13.2 kbps
0	1	0001	IVAS 16.4 kbps
0	1	0010	IVAS 24.4 kbps
0	1	0011	IVAS 32 kbps
0	1	0100	IVAS 48 kbps
0	1	0101	IVAS 64 kbps
0	1	0110	IVAS 80 kbps
0	1	0111	IVAS 96 kbps
0	1	1000	IVAS 128 kbps
0	1	1001	IVAS 160 kbps
0	1	1010	IVAS 192 kbps
0	1	1011	IVAS 256 kbps
0	1	1100	IVAS 384 kbps
0	1	1101	IVAS 512 kbps
0	1	1110	Reserved
0	1	1111	IVAS 5.2 kbps SID

The ToC also allows signaling the EVS bit rates defined in Tables A.4 and A.5 in [3] when the EVS/IVAS mode bit is set to 1 or when the EVS/IVAS mode bit and the Unused/IVAS indicator bit are set to 0.

NO_DATA and SPEECH_LOST frames for both EVS and IVAS modes are signalled with the bit combinations in Table A.4 in Annex A of [3].

NOTE: Received NO_DATA or SPEECH_LOST frames do not relate to either EVS or IVAS modes but simply indicate a non-existent or lost frame.

A.3.3.3.3 E (Extra) byte

A.3.3.3.3.1 General

The specific E byte structure in the IVAS payload header is shown in Figure A.3.3.3.3.1-1. E bytes contain extra information and shall precede the ToC bytes of the coded frames they relate to. There may be multiple E bytes preceding a ToC byte. After the initial E-byte with the CMR there may be multiple subsequent E bytes preceding ToC bytes. Subsequent E bytes may be extended by another E byte of the same type. E bytes may precede any ToC byte; E bytes in the current version of this specification are only permitted before the first ToC byte.

The E (Extra) byte structure is shown in Figure A.3.3.3.3.1-1.

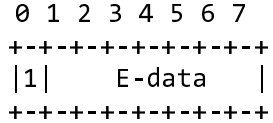


Figure A.3.3.3.3.1-1: E (Extra) byte structure

Parsing of one payload header byte follows the state machine of Figure A.3.3.3.3.1-2.

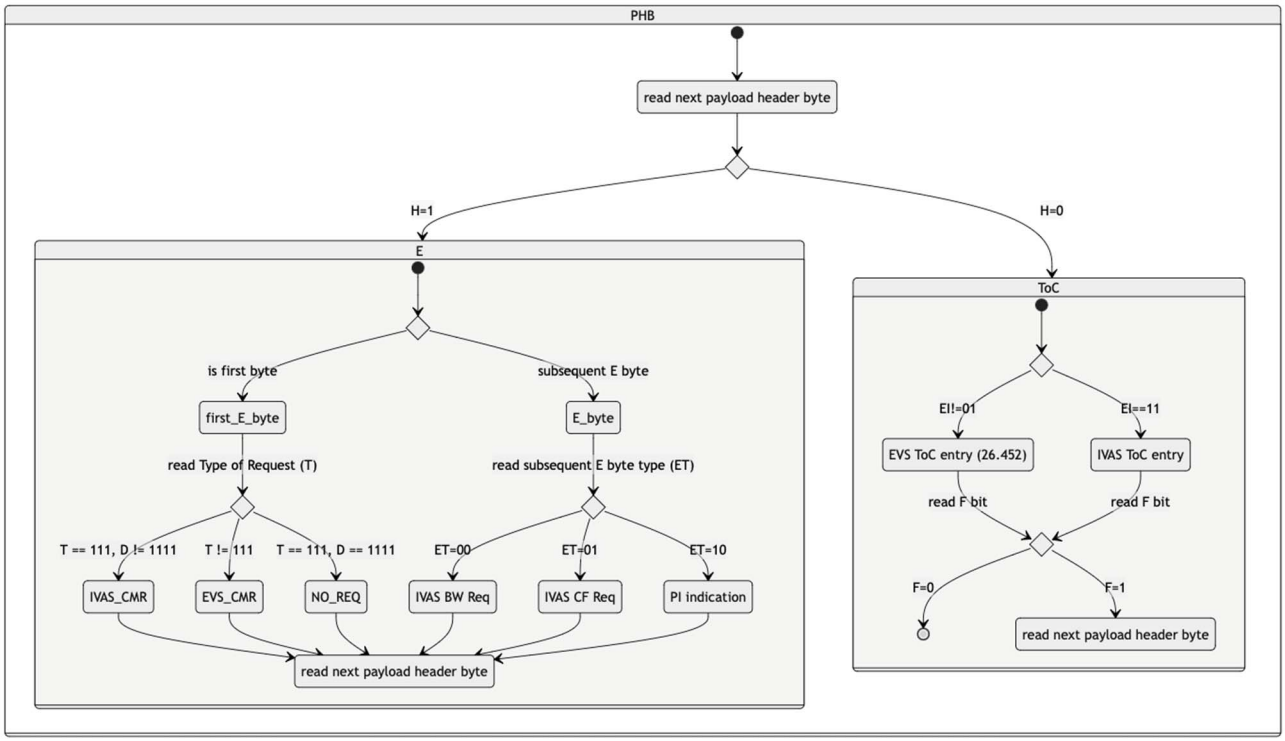


Figure A.3.3.3.3.1-2: State Machine for parsing a Payload Header Byte.

A.3.3.3.3.2 Initial E-byte (CMR)

If a codec mode request (CMR) is sent in the current RTP packet, the initial E byte follows the structure of the CMR byte as defined in Figure A.4 of [3]. The previously "Reserved" entries of Table A.3 in 3] when the T (Type of Request) field is 111 of Figure A.4 of [3] are replaced according to Table A.3.3.3.2-1 .

Table A.3.3.3.2-1: Structure of the CMR byte for T=111

Code		Definition
T	D	BR
111	0000	IVAS 13.2
	0001	IVAS 16.4
	0010	IVAS 24.4
	0011	IVAS 32
	0100	IVAS 48
	0101	IVAS 64
	0110	IVAS 80
	0111	IVAS 96
	1000	IVAS 128
	1001	IVAS 160
	1010	IVAS 192
	1011	IVAS 256
	1100	IVAS 384
	1101	IVAS 512
	1110	Reserved
	1111	NO_REQ

CMR code-point "NO_REQ" remains as defined in Table A.3 in [3]; it is specified as equivalent to no CMR-value being sent. The receiver of "NO_REQ" shall ignore it.

The resulting byte structure is shown in Figure A.3.3.3.3.2-1.

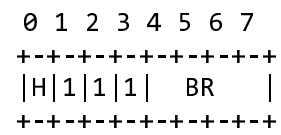


Figure A.3.3.3.3.2-1: Initial E byte structure for IVAS (same as EVS CMR byte structure)

BR (4 bit): IVAS bit rate as indicated in Table A.3.3.3.2-1.

NOTE: When operating in IVAS Immersive mode, a received EVS CMR (T=000..110) is be interpreted as a request to switch to EVS operation mode. When operating in EVS mode, a received IVAS (Immersive) CMR (T=111) is be interpreted as a request to switch to IVAS Immersive operation mode.

A.3.3.3.3.3 Subsequent E-bytes

A.3.3.3.3.3.1 General

Subsequent E byte(s) (after the initial E byte) may follow to request bandwidth, coded format, or to indicate the presence of PI data in the payload as described in the following clauses. Reserved bits in the following E byte structures shall be set to 0, unless defined. The common fields in a subsequent E-byte are:

H (1 bit): Header Type identification bit. For an E byte this bit is always set to 1.

ET (2 bits): Type of subsequent E byte (00, 01, 10, 11) as indicated in Table A.3.3.3.3.3-1. The value 11 is reserved and shall not be used.

Table A.3.3.3.3.3-1: ET field in a subsequent E byte

ET	Definition
00	Bandwidth Request
01	Format Request
10	PI indication
11	reserved

A.3.3.3.3.3.2 Bandwidth Request

Bandwidth requests are defined as shown in Figure A.3.3.3.3.3.2-1 .

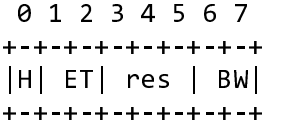


Figure A.3.3.3.3.3.2-1: Subsequent E byte structure for bandwidth request (ET=00)

BW (2 bits): Requested bandwidth as indicated in Table A.3.3.3.3.3.2-1.

Table A.3.3.3.3.3.2-1: BW field in a subsequent E byte

BW	Definition
00	WB
01	SWB
10	FB
11	NO_REQ

A.3.3.3.3.3.3 Coded Format Request

Coded format requests are defined as shown in Figure A.3.3.3.3.3.3-1

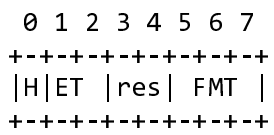


Figure A.3.3.3.3.3.3-1: Subsequent E byte structure for coded format request (ET=01)

FMT (3 bits): Requested coded format as indicated in Table A.3.3.3.3.3.3-1.

Table A.3.3.3.3.3.3-1: FMT field in a subsequent E byte

FMT	Definition
000	stereo
001	SBA
010	MASA
011	ISM
100	MC
101	OMASA
110	OSBA
111	NO_REQ

NOTE: Mono is not included in Table A.3.3.3.3.3.3-1as mono coding in IVAS is handled by the EVS modes.

A.3.3.3.3.3.4 PI Indication

PI indication to indicate PI data presence in the payload are defined as shown in Figure A.3.3.3.3.3.4-1 .

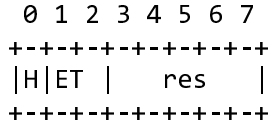


Figure A.3.3.3.3.3.4-1: Subsequent E byte structure to indicate PI data presence in the payload (ET=10)

A.3.4 Frame Data

The RTP payload comprises, apart from headers, one or several coded frames, the Frame Data.

The bits are in the same order as produced by the IVAS encoder, where the first bit is placed left-most immediately following the IVAS payload header.

The format supports the transmission of EVS (primary and AMRWB-IO) and IVAS coded frames.

A.3.5 Processing Information (PI) data

A.3.5.1 General

The support of PI data is disabled by default and needs to be explicitly enabled using the pi-types parameter as defined in clause A.4.1.

In addition to one or more IVAS frame(s), one or more PI data may be transmitted as part of the PI data section in the IVAS RTP payload. PI data in the forward direction (i.e., from a media sender to a media receiver) is a mechanism to assist the processing of the IVAS frame(s) at the renderer, where the PI data is acquired prior to the rendering of IVAS frame(s).

Figure A.3.5.1-1 presents the structure of a PI data section which is formed by the PI-specific header section followed by the PI frame data. The PI header section (see clause A.3.5.a) identifies the different PI data frames (see clause

A.3.5x2). The PI data section is included in the IVAS RTP payload as described in clause A.3.3.2 and in figure A.3.3.2-1.

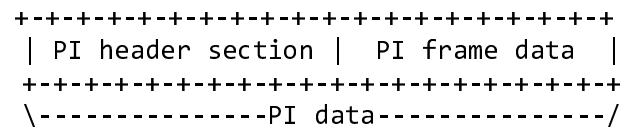


Figure A.3.5.1-1: The structure of a PI data section.

A.3.5.2 PI data header

As presented in clause A.3.5.1 and in figure A.3.5.1-1, a PI data block contains a header section and a frame data section. Each PI header identifies the type and size for the associated PI data frame. Furthermore, the PI header identifies to which audio frame(s) the PI data is associated with through the PI frame marker bits (PM). All the PI header indications are added to the PI header section in a row before the PI data frames.

PI data header indication structure is presented in figure A.3.5.2-1.

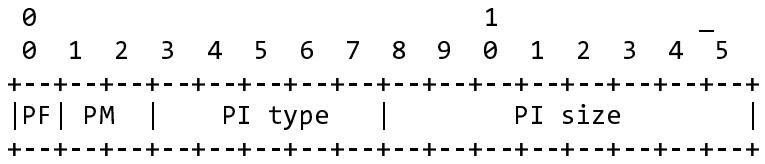


Figure A.3.5.2-1: PI data header.

The header elements are defined as:

PF (1 bit): If set to 1, the bit indicates that the PI header indication is followed by another PI header indication. If set to 0, the bit indicates that this PI header indication is the last one in this payload and no further PI header indication follows this entry. See Table A.3.5.2-1.

PM (2 bits): PI frame marker bits indicate the associated audio frame for the PI data frame, see Table A.3.5.2-2

PI type (5 bits): PI type bits indicate the type for the PI data as indicated in tables A.3.5.5-1 and A.3.5.5-2.

PI size (8 bits): PI size bits indicate the size for the PI data frame in bytes, see table A.3.5.2-3. A size of zero indicates that there is no PI data frame associated for the PI header indication.

Table A.3.5.2-1: PF bits for a PI header.

PF bits	Indication
0	Last header indication
1	Header indication follows

Table A.3.5.2-2: Marker bits for a PI header byte.

PM bits	Frame marker indication
00	Reserved
01	Not last PI header for this frame
10	Last PI header for this frame
11	General (applied to all frames)

The PI headers are listed in the beginning of a PI data section in order. First, the PI headers with marker bits PM=11 (i.e., the generally applied PI data is identified first). The following PI headers are associated with the audio frames in the payload in order. PM=01 indicates that the current PI header/data is not the last one for the current audio frame and more PI headers/data will follow. PM=10 indicates that this PI header/data is the last one that is associated with the current audio frame. The next PI data is then associated with the next audio frame, i.e. the time stamp of the PI frame is increase by 320 ticks. The parsing of a PI header indication is illustrated in a state machine in Figure A.3.5.2-2.

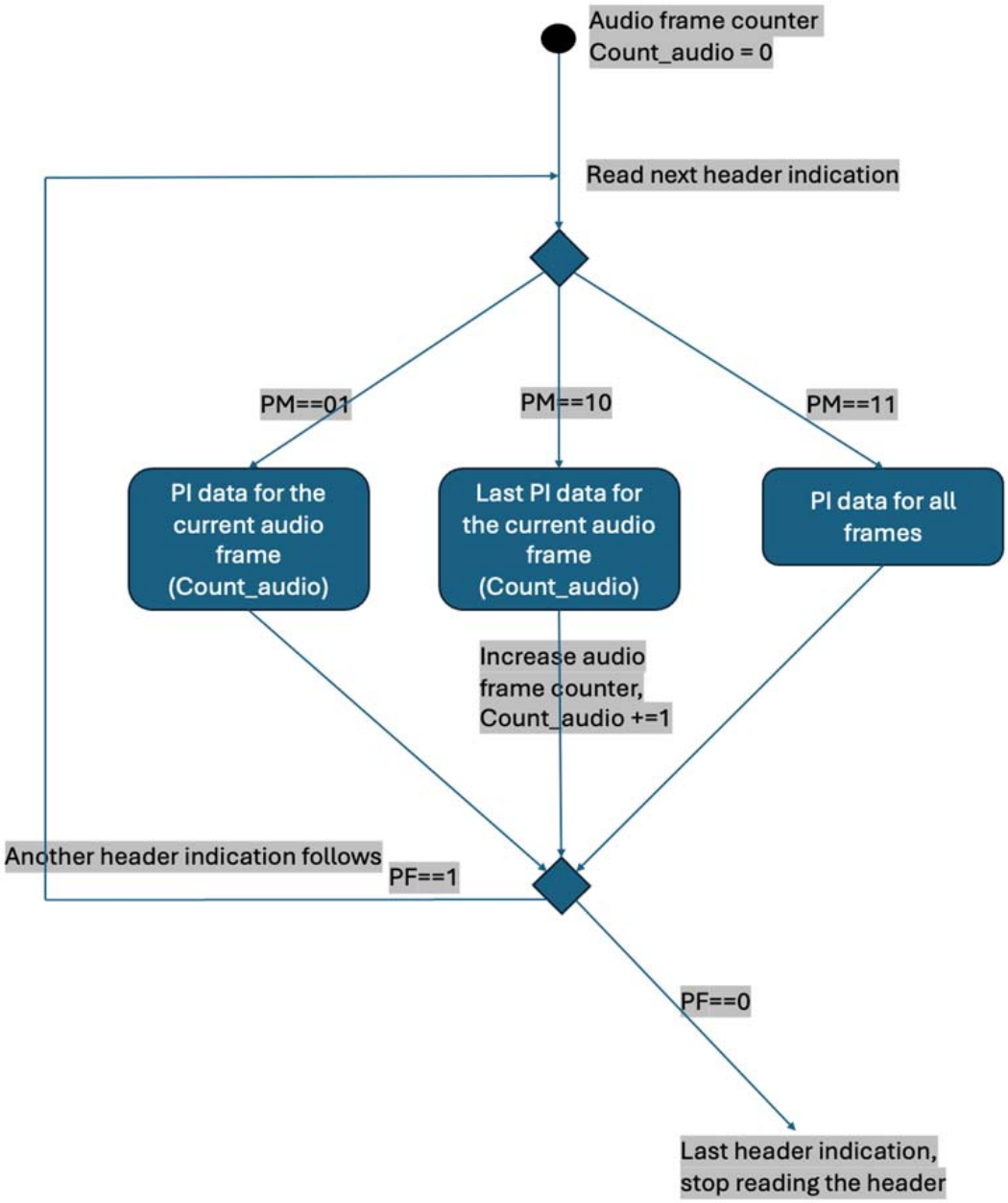


Figure A.3.5.2-2: State machine for parsing a PI header byte.

Table A.3.5.2-3 indicates the values for the “size” bits in the PI header. A size value of zero indicates that there is no associated PI data for this header entry. A value of (1111 1111) indicates that another size byte follows. The size of the PI data is then calculated as the sum of the respective sizes indicated by the “size” bytes, where a size of 255 bytes is used for the “size” byte with (1111 1111) bit sequence. For example, if there are two “size” bytes, a first byte of (1111 1111) and a second byte of (0000 1111, indicating a size of 15 bytes), the size of the PI data is 255 + 15 = 270 bytes. Recursive size indication with subsequent “size” bytes is also possible.

Table A.3.5.2-3: Size bits for PI header.

PI Size bits	Indicated size in bytes (or other indication)
0000 0000	0
0000 0001	1
…	…
1111 1110	254
1111 1111	Another size byte follows

A.3.5.3 Media time when IVAS PI data is included in RTP packets

When the IVAS codec is used, then RTP packets may include both PI data and audio data, and the PI data may need to be synchronized with the audio data.

When forward direction PI data is included in the RTP packets, the following applies:

- The PI data is associated with an audio frame and use the media time of the audio data.
  - If PI data needs to be transmitted and no audio frame is available, e.g., during DTX periods, then a NO_DATA frame is included in the packet containing the PI data. Alternatively, the PI data can be transmitted with the SID frames. See clause A.3.5.4 for more information.
  - If PI data is not needed to be transmitted when there is no audio frame available, e.g., during DTX periods, then the transmission of the PI data can be delayed until an audio frame is available. If there are multiple PI data frames with the same type available after a delay period, the most recent PI data may be selected for transmission (e.g., the most recent device orientation may be transmitted). The other (older) PI data frames with the same type may be ignored. Depending on the type of the delayed PI data frames, all of the PI data frames with the same type may be transmitted. See clause A.3.5.4 for more information.
- When receiving an RTP packet with both PI data and several audio frames:
  - the media time of the first audio frame is calculated from the RTP time stamp,
  - the media time of a subsequent audio frames is calculated from the media time of a preceding audio frame by adding 20 ms.
- PI data does not add to the media time.
- PI data can be sent in a separate RTP packet from the audio frame and then use the media time calculated from the RTP time stamp.

A.3.5.4 PI data handling during DTX

No audio frames are transmitted when DTX operation mode is determined by the media sender. Consequently, there is no RTP time stamp available to be associated with the transmitted PI data. If the PI data is obtained during DTX operation and needs to be transmitted as soon as possible, the PI data can be associated with the next transmitted SID frame or with a NO_DATA frame. In these cases, the RTP time stamp for the PI data transmission is obtained from the media time of the respective SID or NO_DATA frames.

If the transmission of the PI data can wait until the DTX period is over, the transmission of the PI data can be delayed until the next audio frame is available. In this case, the RTP timestamp of the transmitted packet is calculated from the media time of the oldest PI data included in the packet with one or more NO_DATA frames included before the first audio frame.

If the transmission of the PI data is delayed, there can be multiple PI data frames of the same type (e.g., of device orientation type) waiting to be transmitted at the end of DTX operation. In this case, the latest or all PI data can be selected to be transmitted (e.g., the latest device orientation).

A.3.5.5 Supported PI data types

Supported PI types are listed in tables A.3.5.5-1 and A.3.5.5-2 and described in the following subsections. Table A.3.5.5-1 lists PI types for forward direction signalling. Table A.3.5.5-2 lists additional PI types.

Table A.3.5.5-1 : Supported forward direction PI types in an IVAS session.

Type bits	Forward direction PI type	Description	SDP indication	Size (bytes)	Described in clause
00000	SCENE_ORIENTATION	Describes the orientation of a spatial audio scene in unit quaternions.	fsco	8	A.3.5.6.1.2
00001	DEVICE_ORIENTATION_COMPENSATED	Describes the orientation of a device in unit quaternions. The orientation is compensated in the transmitted audio.	fdoc	8	A.3.5.6.1.3
00010	DEVICE_ORIENTATION_UNCOMPENSATED	Describes the orientation of a device in unit quaternions. The orientation is not compensated in the transmitted audio.	fdou	8	A.3.5.6.1.3
00011	ACOUSTIC_ENVIRONMENT	Selects and optionally describes the acoustic environment.	face	1,5 or 8	A.3.5.6.2

Table A.3.5.5-2: Additional PI types in an IVAS session.

Type bits	PI type	Description	SDP indication	Size (bytes)
00100-11110	Reserved	-	-	-
11111	NO_PI_DATA	Indicates an empty PI data frame.	nopi	0

NO_PI_DATA PI data type can be used to indicate empty PI data sections. The PM marker bits for a NO_PI_DATA PI data type shall be set as PM=10, see table A.3.5.2-2. For example, if an IVAS RTP payload includes multiple audio frames, and some of the audio frames do not have associated PI data, NO_PI_DATA PI type can be used.

A.3.5.6 Forward direction PI data types

A.3.5.6.1 Orientation PI data (forward direction)

A.3.5.6.1.1 Orientation data structures

Figure A.3.5.6.1.1-1 below shows PI orientation data structures in quaternions with 16 bits reserved for each component. The quaternion component values range from -1 to 1 according to Q15, in which resolution for a single component is $(2^{15}-1)^{-1} \approx 0.00003$. The represented quaternion is a unit quaternion. Following the IVAS coordinate system in 7.4.3.1, a quaternion of (w=0, x=1, y=0, z=0) represents the frontal direction. The positive x-axis points towards the frontal direction, the positive y-axis points towards the left direction and the positive z-axis points towards the up direction.

NOTE: An orientation in Euler convention can be converted to quaternions before transmission, see clause 7.4.3.2 for the conversion operation and clause 7.4.3.1 for Euler angle definitions in the IVAS coordinate system.

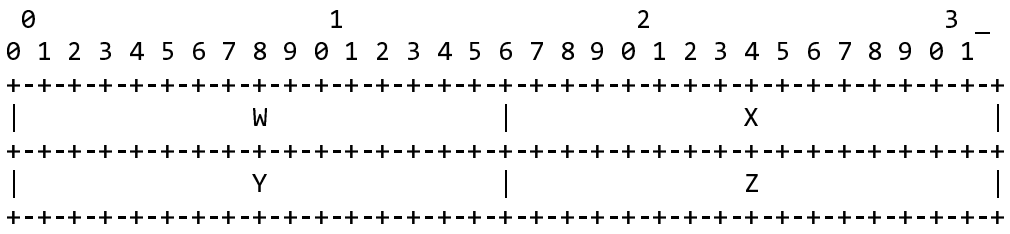


Figure A.3.5.6.1.1-1: PI orientation data as quaternions.

The received orientations can be transmitted to the external orientation handling and processed as stated in clause 7.4.4.

A.3.5.6.1.2 Scene orientation

The SCENE_ORIENTATION PI data describes the orientation of the capturer side scene. The frontal direction of the scene points towards the capture frontal direction. See also clause 7.4.2.1 (Scene orientation). In the PI data, the azimuth and elevation values of the scene orientation are transformed into (unit) quaternions. A frontal direction of zero azimuth and zero elevation corresponds to (w=0, x=1, y=0, z=0) in quaternions, see clause 7.4.3.1 for the IVAS coordinate system.

The SCENE_ORIENTATION PI data is applied to the audio frame with the same timestamp. The latest received SCENE_ORIENTATION PI data is used until a new SCENE_ORIENTATION PI data is received.

A.3.5.6.1.3 Device orientation

The DEVICE_ORIENTATION_COMPENSATED and DEVICE_ORIENTATION_UNCOMPENSATED PI data describes the orientation of the sender (capture) device. I.e., the orientation indicates the orientation deviation from the frontal capture direction, when the device is used for audio capture. The frontal direction of the device point towards the capture frontal direction, i.e. (w=0, x=1, y=0, z=0) in quaternions, see clause 7.4.3.1 for the IVAS coordinate system.

The orientation of a capturing device can be used to stabilize the captured spatial audio content by, e.g., removing undesirable orientation changes. The sender device may be a non-stationary capturing device (e.g., a mobile phone with multiple microphones or a spatial audio capture microphone array). For example, a caller is holding a mobile phone in their hand or wearing a head-mounted display with spatial audio capturing capabilities. Some of the movements of the caller that affect the spatial audio capture can be undesirable (e.g., hand or head movements). The undesirable movements (or orientations) can be compensated in the transmitted spatial audio content via the DEVICE_ORIENTATION_COMPENSATED PI type. When the device orientation is compensated in the transmitted spatial audio content, the received spatial audio can be rendered to the receiver (in their user space) without experiencing the undesirable orientations or movements. In case the spatial audio content is transmitted without orientation compensation, the compensation can be performed by the receiver device. In this case, DEVICE_ORIENTATION_UNCOMPENSATED PI type can be used to transmit the change in the capture device orientation.

DEVICE_ORIENTATION_COMPENSATED PI data indicates that the transmitted orientation is already compensated in the related transmitted audio.

DEVICE_ORIENTATION_UNCOMPENSATED PI data indicates that the transmitted orientation is not compensated in the related transmitted audio.

The device orientation PI data is applied to the audio frame with the same timestamp. The latest received device orientation PI data is used until a new device orientation PI data is received.

A.3.5.6.2 Acoustic environment PI data

Acoustic environment (AE) PI data frames can be used to transmit room acoustic data. The room acoustic data consist of late reverb parameters and optionally early reflections parameters. The detailed description of room acoustics parameters is provided in clause 7.4.8.

The late reverb parameters include:

- RT60 – indicating the time that it takes for the reflections to reduce 60 dB in energy level, per frequency band,
- DSR – diffuse to source signal energy ratio, per frequency band,
- Pre-delay – delay at which the computation of DSR values was performed, which can be also seen as the threshold point between early reflections and late reverberation phase.

Both RT60 and DSR parameters are specified per frequency band. Pre-defined or custom frequency bands can be used. Pre-delay is a scalar.

The simplified early reflections parameters include:

- 3D rectangular virtual room dimensions,
- Broadband energy absorption coefficient per wall.

To control acoustic environments runtime, acoustic environment PI data frames can be used. An acoustic environment PI data frame can contain:

- an acoustic environment identifier alone (7 bits),
- a compact representation of the acoustic environment containing only late reverb parameters (40 bits),
- a compact representation of the acoustic environment containing late reverb and simplified early reflections coefficients (64 bits).

The content of the PI frame received is determined by its size.

Acoustic environment can be selected by sending an acoustic environment PI frame containing a seven-bit AE identifier, as illustrated in figure A.3.5.6.2-1. Such an acoustic environment should be available upfront. It can be also provided using an AE PI data frame containing a compact acoustic environment representation. It allows for the following graceful degradation mechanism:

- full AE representation should be used if available, otherwise:
- compact AE representation should be used if available, otherwise:
- default AE definition should be used if none of the above are available.

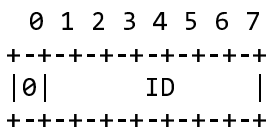


Figure A.3.5.6.2-1: Acoustic environment PI data frame (ACOUSTIC_ENVIRONMENT) containing an AE identifier.

Acoustic environment data can also get updated real-time. In case of real-time updates, the AE data can be transmitted as compact packets using RTP protocol.

Compact packets contain a low-resolution representation of the acoustic environment. This facilitates avoiding spikes in transmission and is better suited for repeated transmission allowing for instant synthesis of the new acoustic environment, although with initially reduced accuracy. A compact acoustic environment PI can be provided with or without early reflections coefficients. An AE data frame without early reflections is presented in Figure A.3.5.6.2-2.

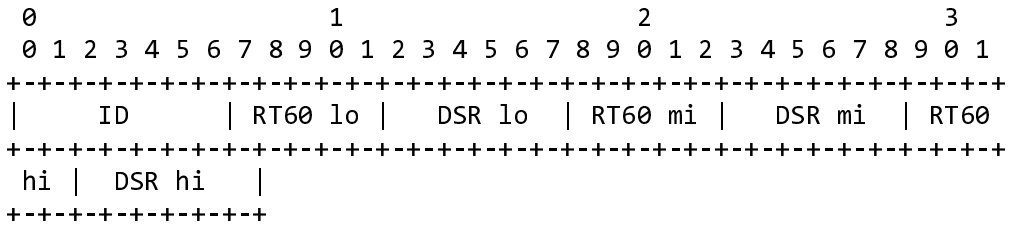


Figure A.3.5.6.2-2: Acoustic environment PI data frame (ACOUSTIC_ENVIRONMENT) containing a compact AE representation with late reverb parameters.

An AE data frame with early reflections is presented in Figure A.3.5.6.2-3.

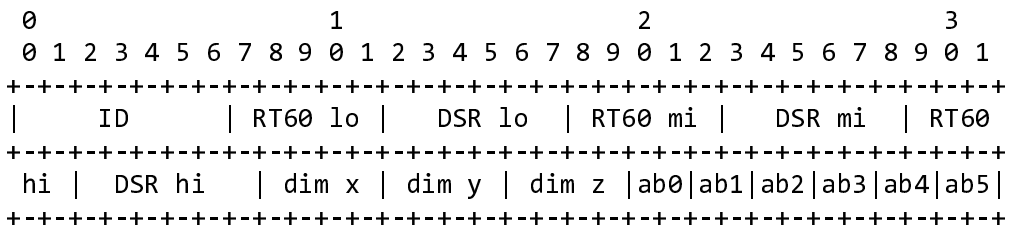


Figure A.3.5.6.2-3: Acoustic environment PI data frame (ACOUSTIC_ENVIRONMENT) containing a compact AE representation with late reverb and early reflection parameters.

The data frame including early reflections consists of an acoustic environment ID field (7 bits), three RT60 fields (5 bits each), three DSR fields (6 bits each), three room dimension (dim) fields (4 bits each) and six absorption coefficient (ab) fields summing up to 64 bits size. The RT60 and DSR values are provided for three frequency bands of center frequency {25 Hz, 250 Hz, 2.5 kHz}, which are denoted {lo, mi, hi} in figure A.3.5.6.2-2 respectively. The RT60 fields use 5-bit code *n* representing duration in seconds according to formula $RT_{60}(n) = \frac{1}{100} 2^{\frac{n}{3}}$ with $n \in [0..31]$. The 5-bit codes and related RT60 durations are also shown in table A.3.5.6.2-1.

Table A.3.5.6.2-1 : 5-bit codes and respective RT60 values

Code	Value	Code	Value	Code	Value	Code	Value
00000	0.01	01000	0.0635	10000	0.4032	11000	2.56
00001	0.0126	01001	0.08	10001	0.5080	11001	3.2254
00010	0.0159	01010	0.1008	10010	0.64	11010	4.0637
00011	0.02	01011	0.1270	10011	0.8063	11011	5.12
00100	0.0252	01100	0.16	10100	1.0159	11100	6.4508
00101	0.0317	01101	0.2016	10101	1.28	11101	8.1275
00110	0.04	01110	0.2540	10110	1.6127	11110	10.24
00111	0.0504	01111	0.32	10111	2.0319	11111	12.9016

The DSR values are computed as $DSR(n) = -20 - n$[dB] with $n \in [0..63]$ resulting in the range between -20 and -83 dB as shown in Table A.3.5.6.2-2.

Table A.3.5.6.2-2 : 6-bit codes and respective DSR values

Code	Value	Code	Value	Code	Value	Code	Value
000000	-20	010000	-36	100000	-52	110000	-68
000001	-21	010001	-37	100001	-53	110001	-69
000010	-22	010010	-38	100010	-54	110010	-70
000011	-23	010011	-39	100011	-55	110011	-71
000100	-24	010100	-40	100100	-56	110100	-72
000101	-25	010101	-41	100101	-57	110101	-73
000110	-26	010110	-42	100110	-58	110110	-74
000111	-27	010111	-43	100111	-59	110111	-75
001000	-28	011000	-44	101000	-60	111000	-76
001001	-29	011001	-45	101001	-61	111001	-77
001010	-30	011010	-46	101010	-62	111010	-78
001011	-31	011011	-47	101011	-63	111011	-79
001100	-32	011100	-48	101100	-64	111100	-80
001101	-33	011101	-49	101101	-65	111101	-81
001110	-34	011110	-50	101110	-66	111110	-82
001111	-35	011111	-51	101111	-67	111111	-83

No pre-delay value is transmitted. It gets computed as one tenth of RT60 of 250 Hz band.

The three room dimensions: length (x), width (y), and height (z) in [m], are computed as $dimension(n) = \frac{1}{2}2^{\frac{n}{2}}$ with $n \in [0..15]$ as described in Table A.3.5.6.2-3.

Table A.3.5.6.2-3 : 4-bit codes and respective room dimension values

Code	Value	Code	Value
0000	0.5	1000	8
0001	0.707	1001	11.314
0010	1	1010	16
0011	1.4141	1011	22.627
0100	2	1100	32
0101	2.8282	1101	45.255
0110	4	1110	64
0111	5.6568	1111	90.51

The six absorption coefficients, corresponding to the six room surfaces (front, back, left, right, ceiling, floor) are computed as $absorption\ (n) = 0.08 * 2^{1.05n}$ with $n \in [0..3]$ as shown on Table A.3.5.6.2-4.

Table A.3.5.6.2-4: 2-bit codes and respective absorption coefficient values

Code	Value
00	0.0800
01	0.1656
10	0.3430
11	0.7101

A.4 Payload Format Parameters

A.4.1 IVAS Media Type Registration

The media type for the IVAS codec is to be allocated from the standards tree. This clause defines parameters of the IVAS payload format. This media type registration covers real-time transfer via RTP and non-real-time transfers via stored files. All media type parameters defined in this document shall be supported.

Media type name: audio

Media subtype name: IVAS

Required parameters: none

Optional parameters:

The parameters defined below apply to RTP transfer only:

ptime: see [r1].

maxptime: see [r1].

dtx/dtx-recv: as defined in Annex A of [3].

max-red: see [r5].

channels: The number of audio channels shall not be present.

NOTE: The use of the channels parameter as defined in [r4] does not permit signaling all IVAS Immersive mode coded formats; formats need to be derived from the cf/cf-send/cf-recv parameters.

ivas-mode-switch: This parameter defines the mode at the start or update of the session for the send and the receive directions. Permissible values are 0 and 1. If ivas-mode-switch is 0 or not present, IVAS Immersive mode is used. If ivas-mode-switch is 1, depending on the setting of evs-mode-switch, EVS Primary or AMR-WB IO mode is used The mode initially used in the session may later be modified.

cmr: As defined in Annex A of [3] for the EVS Primary and AMRWB-IO modes. For IVAS Immersive modes the bit rate, bandwidth and format requests are disabled when cmr is -1. The bitrate, bandwidth and format requests are enabled when cmr is 0 or the cmr parameter is not present. When cmr is 1 the bit rate requests using the initial E byte shall be present in every packet (but may be NO_REQ); format and bandwidth requests for IVAS Immersive modes are optional when cmr is 1.

The following parameters are applicable only to IVAS Immersive operation:

NOTE: IVAS computational complexity and memory demands of depend on the setting of the following parameters for source codec bit rate, audio bandwidth, and coded format; in addition, factors beyond the signaling, such as complexity of a specific implementation and the (rendered) output format may be significant.

ibr: Specifies the range of source codec bitrate for IVAS Immersive mode in the session, in kilobits per second, for the direction specified by the session directionality attribute or the suffix. The br parameter can either have: a single bitrate (ibr1); or a hyphen-separated pair of two bitrates (ibr1-ibr2). If a single value is included, this bitrate, ibr1, is used. If a hyphen-separated pair of two bitrates is included, ibr1 and ibr2 are used as the minimum bitrate and the maximum bitrate respectively. ibr1 shall be smaller than ibr2. ibr1 and ibr2 have a value from the set in Table 4.2-2 of the present document. If none of these parameters is present, all bitrates consistent with the IVAS codec capabilities are allowed in the session.

ibr-send/ibr-recv: ibr parameter in send or receive direction.

ibw: Specifies the audio bandwidth for IVAS Immersive modes to be used in the session, for the direction specified by the session directionality attribute or the suffix. ibw has a value from the set: wb, swb, fb, wb-swb, and wb-fb. wb, swb, and fb represent wideband, super-wideband, and fullband respectively, and wb-swb, and wb-fb represent all bandwidths from wideband to super-wideband, and fullband respectively. If none of these parameters is present, all bandwidths consistent with the negotiated bitrate(s) are allowed in the session.

ibw-send/ibw-recv: ibw parameter in send or receive direction.

cf: Specifies the IVAS Immersive mode coded-format (cf) transmitted in the IVAS Immersive mode frames in the session. IVAS coded format corresponds to the format represented in the IVAS Immersive mode coded frames, which is generally the input format to the encoder. The cf parameter is a list of supported comma-separated IVAS Immersive mode coded formats in the order of preference, using the identifiers from Table A.4.1-1 of the present document (column "Identifier"). Selection of the format is application-specific and out of scope of this document. EVS frames in the session are in mono format; switching to mono shall be possible.

Table A.4.1-1: IVAS coded-format

Identifier	Full Name	Clause
Stereo	Stereo Operation	4.2.3
SBA	Scene-based Audio (SBA, Ambisonics) Operation	4.2.4
MASA	Metadata-assisted Spatial Audio (MASA) Operation	4.2.5
ISM	Objects (Independent Streams with Metadata, ISM) Operation	4.2.6
MC	Multi-Channel (MC) Operation	4.2.7
OMASA	Combined Objects and MASA (OMASA) Operation	4.2.9
OSBA	Combined Objects and SBA (OSBA) Operation	4.2.8

Mono is not listed as an IVAS Immersive mode coded-format as EVS is always supported and shall be used for mono.

cf-send/cf-recv: cf parameter in send or receive direction

pi-types: Specifies the supported PI data types for the session. The pi-types parameter is a list of supported comma-separated PI data types using the SDP indications listed in tables A.3.5.5-1 and A.3.5.5-2. If the pi-types parameter is not present, PI data is not enabled for the session.

pi-types-send/pi-types-recv: pi-types parameter in send or receive direction.

pi-br: Specifies the maximum peak bitrate for the PI data section (excluding the E-bytes for indication) for each packet in the session in kilobits per second. Bitrate calculation for PI data shall take the packet interval, i.e. value of ptime into account. The parameter indicates the maximum bitrate for the PI data. If pi-br parameter is not present, a default value of 0 shall be used.

pi-br-send/pi-br-recv: pi-br parameter in send or receive direction.

The following parameters are applicable only to EVS Primary and AMR-WB IO modes:

evs-mode-switch: as defined in Annex A of [3]. If ivas-mode-switch is 0 or not present, evs-mode-switch should not be present and shall be ignored.

hf-only: as specified in Annex A of [3] except that the default and only allowed value of hf-only shall be 1 in this payload format. As the only allowed value for this parameter is 1 it is not required to include this parameter.

NOTE: There is no compact format support in this payload format, contrary to the EVS payload format in Annex A of [3] that enables the compact format by default.

ch-send: Shall not be present. The EVS modes in this payload format shall be mono-only

ch-recv: Shall not be present. The EVS modes in this payload format shall be mono-only.

The following parameters are applicable only to EVS Primary modes:

br: as defined in Annex A of [3]

br-send: as defined in Annex A of [3]

br-recv: as defined in Annex A of [3]

bw: as defined in Annex A of [3]

NOTE: Narrow-band is not supported for IVAS operation

bw-send: as defined in Annex A of [3]

bw-recv: as defined in Annex A of [3]

ch-aw-recv: as defined in Annex A of [3]

The following parameters are applicable only to EVS AMR-WB IO modes:

mode-set: as defined in Annex A of [3]

mode-change-period: see [r5].

mode-change-capability: as defined in Annex A of [3]

mode-change-neighbor: see [r5]

A.4.2 Mapping media type parameters into SDP

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [r1], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the IVAS codec, the mapping is as follows:

- The media type ("audio") goes in SDP "m=" as the media name.
- The media subtype (payload format name) goes in SDP "a=rtpmap" as the encoding name. The RTP clock rate in "a=rtpmap" shall be 16000, and the encoding parameters (number of channels) shall be omitted.
- The parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.
- Any remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the media type parameter string as a semicolon-separated list of parameter=value pairs.

Mapping to fields in SDP is specified in clause 6 of [r2].

A.4.3 Detailed Description of Usage of SDP Parameters

A.4.3.1 Offer-Answer Model Considerations

The following considerations apply when using SDP Offer-Answer procedures to negotiate the use of IVAS payload in RTP:

hf-only: Shall not be included in the SDP offer. The answerer shall include this parameter only if it is set to 1 in the SDP offer. If the value in the SDP offer is not equal to 1, the payload type shall be rejected.

ivas-mode-switch: When ivas-mode-switch is not offered for a payload type, the answerer may include ivas-mode-switch for the payload type in the SDP answer. When ivas-mode-switch is offered for a payload type and the payload type is accepted, the answerer shall not modify or remove ivas-mode-switch for the payload type in the SDP answer.

cmr: When cmr is not offered for a payload type, the answerer may include cmr for the payload type in the SDP answer. When cmr is offered for a payload type and the payload type is accepted, the answerer shall not modify or remove cmr for the payload type in the SDP answer.

ibr: When the same bitrate or bitrate range is defined for the send and the receive directions, ibr should be used but ibr-send and ibr-recv may also be used. ibr can be used even if the session is negotiated to be sendonly, recvonly, or inactive. For sendonly session, ibr and ibr-send can be interchangeably used. For recvonly session, ibr and ibr-recv can be interchangeably used. When ibr is not offered for a payload type, the answerer may include ibr for the payload type in the SDP answer. When ibr is offered for a payload type and the payload type is accepted, the answerer shall include ibr in the SDP answer which shall be identical to or a subset of ibr for the payload type in the SDP offer.

ibr-send: When ibr-send is not offered for a payload type, the answerer may include ibr-recv for the payload type in the SDP answer. When ibr-send is offered for a payload type and the payload type is accepted, the answerer shall include ibr-recv in the SDP answer, and the ibr-recv shall be identical to or a subset of ibr-send for the payload type in the SDP offer.

ibr-recv: When ibr-recv is not offered for a payload type, the answerer may include ibr-send for the payload type in the SDP answer. When ibr-recv is offered for a payload type and the payload type is accepted, the answerer shall include ibr-send in the SDP answer, and the ibr-send shall be identical to or a subset of ibr-recv for the payload type in the SDP offer.

ibw: When the same bandwidth or bandwidth range is defined for the send and the receive directions, ibw should be used but ibw-send and ibw-recv may also be used. ibw can be used even if the session is negotiated to be sendonly, recvonly, or inactive. For sendonly session, ibw and ibw-send can be interchangeably used. For recvonly session, ibw and ibw-recv can be interchangeably used. When ibw is not offered for a payload type, the answerer may include ibw for the payload type in the SDP answer. When ibw is offered for a payload type and the payload type is accepted, the answerer shall include ibw in the SDP answer, which shall be identical to or a subset of ibw for the payload type in the SDP offer.

ibw-send: When ibw-send is not offered for a payload type, the answerer may include ibw-recv for the payload type in the SDP answer. When ibw-send is offered for a payload type and the payload is accepted, the answerer shall include ibw-recv in the SDP answer, and the ibw-recv shall be identical to or a subset of ibw-send for the payload type in the SDP offer.

ibw-recv When ibw-recv is not offered for a payload type, the answerer may include ibw-send for the payload type in the SDP answer. When ibw-recv is offered for a payload type and the payload is accepted, the answerer shall include ibw-send in the SDP answer, and the ibw-send shall be identical to or a subset of ibw-recv for the payload type in the SDP offer.

cf: The SDP offer [shall] list at least one but may list several IVAS Immersive mode coded formats. The SDP answer shall include at least one IVAS Immersive mode coded format and should respond with the one most preferred coded format from the list in the SDP offer. If more than one format is present in the SDP answer, the first format shall be used at the start of a session and may only be modified by the adaptation mechanisms present in this specification. When the same IVAS Immersive mode coded formats are defined for the send and the receive directions, cf should be used but cf-send and cf-recv may also be used. For sendonly session, cf and cf-send can be interchangeably used. For recvonly session, cf and cf-recv can be interchangeably used.

cf-send: When cf-send is offered for a payload type and the payload type is accepted, the answerer shall include cf-recv in the SDP answer, and the cf-recv shall be identical to or a subset of the cf-send parameter for the payload type in the SDP offer.

cf-recv When cf-recv is offered for a payload type and the payload type is accepted, the answerer shall include cf-send in the SDP answer, and the cf-send shall be identical to or a subset of the cf-recv parameter for the payload type in the SDP offer.

pi-types: The SDP offer shall list at least one but may list several supported pi types when pi data is enabled in the offer. When one or more of the offered pi types are supported, the SDP answer shall be identical to or a subset of the pi types listed in the SDP offer. When the same pi types are defined for the send and the receive directions, pi-types should be used but pi-types-send and pi-types-recv may also be used. For sendonly session, pi-types and pi-types-send can be interchangeably used. For recvonly session, pi-types and pi-types-recv can be interchangeably used. When none of the offered pi-types is supported, the answerer shall not include pi-types in the SDP answer.

pi-types-send: When pi-types-send is offered in the SDP offer and it is accepted, the answerer shall include pi-types-recv in the SDP answer, and the pi-types-recv shall be identical to or a subset of the pi-types-send parameter in the SDP offer.

pi-types-recv: When pi-types-recv is offered in the SDP offer and it is accepted, the answerer shall include pi-types-send in the SDP answer, and the pi-types-send shall be identical to or a subset of the pi-types-recv parameter in the SDP offer.

pi-br: When the same bitrate is defined for the send and the receive directions, pi-br should be used but pi-br-send and pi-br-recv may also be used. pi-br can be used even if the session is negotiated to be sendonly, recvonly, or inactive. For sendonly session, pi-br and pi-br-send can be interchangeably used. For recvonly session, pi-br and pi-br-recv can be interchangeably used. When pi-br is not offered in the SDP offer, the answerer shall not include pi-br in the SDP answer. When pi-br is offered in the SDP offer and it is accepted, the answerer shall include pi-br in the SDP answer which shall be identical or lower than pi-br in the SDP offer.

pi-br-send: When pi-br-send is offered in the SDP offer and it is accepted, the answerer shall include pi-br-recv in the SDP answer, and the pi-br-recv shall be identical or lower than pi-br-send in the SDP offer.

pi-br-recv: When pi-br-recv is offered in the SDP offer and it is accepted, the answerer shall include pi-br-send in the SDP answer, and the pi-br-send shall be identical or lower than pi-br-recv in the SDP offer.

The offer-answer considerations for the remaining EVS parameters are as described in TS 26.445 Annex A.3.3.1 [3].

Annex B (informative): Change history

Change history							
Date	Meeting	TDoc	CR	Rev	Cat	Subject/Comment	New version
2023-11	SA4#126	S4-231842				Presented to 3GPP SA4 Audio SWG	0.0.1
2023-11	SA4#126	S4-231989				Submitted to SA4 plenary for information	0.1.0
2023-12	SA#102	SP-231301				Version 1.0.0 by MCC	1.0.0
2024-01	SA4#127	S4-240260				Version 1.0.1 presented to SA4 (Audio SWG) for agreement	1.0.1
2024-02	SA4#127	S4-240376				Version 1.1.0 presented to SA4 for agreement	1.1.0
2024-03	SA#103	SP-240030				Version 2.0.0 created by MCC	2.0.0
2024-03						Version 18.0.0 created by MCC	18.0.0
2024-06	SA#104	SP-240693	0003	2	B	Adding ISAR track-a split rendering feature	18.1.0
2024-06	SA#104	SP-240694	0002	4	F	Corrections to TS 26.253 Annex A	18.1.0
2024-06	SA#104	SP-240694	0001	3	F	Corrections to TS 26.253	18.1.0
2024-06	SA#104					Change of spec title as approved by TSG SA in SP-240917	18.1.0

History

Document history		
V18.0.0	May 2024	Publication
V18.1.0	July 2024	Publication