
**Information technology — MPEG
audio technologies —**

**Part 2:
Spatial Audio Object Coding (SAOC)**

*Technologies de l'information — Technologies audio MPEG —
Partie 2: Codage d'objet audio spatial (SAOC)*



Reference number
ISO/IEC 23003-2:2018(E)

Documents
Document Sharing Hub
© ISO/IEC 2018



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2018

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

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

Published in Switzerland

Contents

Page

Foreword.....	v
Introduction.....	vi
1 Scope.....	1
2 Normative references.....	1
3 Terms and definitions	1
4 Notations and abbreviated terms	3
4.1 Notation	3
4.2 Operations	3
4.3 Constants.....	3
4.4 Variables	3
4.5 Abbreviated terms.....	6
5 SAOC overview	7
5.1 General.....	7
5.2 Basic structure of the SAOC transcoder/decoder	8
5.3 Tools and functionality	10
5.4 Delay and synchronization.....	11
5.5 SAOC Profiles and levels.....	17
6 Syntax	20
6.1 Payloads for SAOC.....	20
6.2 Definition	35
7 SAOC processing	43
7.1 Compressed data stream decoding and dequantization of SAOC data.....	43
7.2 Compressed data stream encoding and quantization of MPS data.....	46
7.3 Time/frequency transforms	47
7.4 Signals and parameters	47
7.5 SAOC transcoding/decoding modes for baseline and LD profiles	51
7.6 EAO processing for baseline and LD profiles.....	64
7.7 SAOC-DE profile decoding modes.....	73
7.8 DCU processing	75
7.9 Modification range control for SAOC-DE processing modes.....	79
7.10 MBO processing.....	80
7.11 MCU Combiner.....	81
7.12 Effects	83
7.13 Low power SAOC processing.....	86
7.14 Low delay SAOC processing.....	87
8 Transport of SAOC side information	89
8.1 Overview	89
8.2 Transport and signalling in an MPEG environment.....	89
8.3 Transport of SAOC data over PCM channels.....	93
9 Transport of predefined rendering information	94
9.1 General.....	94
9.2 Rendering information description file format.....	95
10 Conformance testing.....	96
10.1 General.....	96
10.2 Terms and definitions	96
10.3 SAOC conformance testing.....	96
10.4 Bitstreams.....	96

10.5	SAOC decoder/transcoder	105
11	Reference software	119
11.1	Reference software structure.....	119
Annex A	(normative) Tables.....	121
Annex B	(normative) Low delay MPEG surround	150
Annex C	(informative) Effects processing.....	161
Annex D	(informative) Encoder.....	163
Annex E	(informative) Guidelines for rendering matrix specification	167
Annex F	(informative) MCU combiner	169
Annex G	(informative) Reference software	171

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see <http://patents.iec.ch>).

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see: www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 23003-2:2010), which has been technically revised. It also incorporates the Amendments ISO/IEC 23003-2:2010/Amd 1:2015, ISO/IEC 23003-2:2010/Amd 2:2015, ISO/IEC 23003-2:2010/Amd 3:2015, ISO/IEC 23003-2:2010/Amd 4:2016 and ISO/IEC 23003-2:2010/Amd 5:2016 and the Technical Corrigenda ISO/IEC 23003-2:2010/Cor 1:2012 and ISO/IEC 23003-2:2010/Cor 2:2014.

The main changes compared to the previous edition are as follows:

- clarifications on SAOC-DE profile description;
- corrections to SAOC-DE profile specification;
- corrections to SAOC-DE profile;
- corrections to MPEG SAOC IS text;
- corrections to the low power mode.

A list of all parts in the ISO/IEC 23003 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

In the preferred modes of operating, the SAOC system, the transmitted signal can be either mono, stereo or 3-channel. The audio objects can be represented by a mono, stereo, or 3-channel signal or have the MPEG surround (MPS) multi-channel background object (MBO) format. The additional parametric data exhibits a significantly lower data rate than required for transmitting all objects individually, making the coding very efficient. At the same time, this ensures compatibility of the transmitted signal with legacy devices.

When a multi-channel rendering setup (e.g. a 5.1 loudspeaker setup) is required, the SAOC system acts as a transcoder, converting the additional parametric data to MPS parameters, and interfaces to the MPS decoder that acts as rendering device. For certain rendering setups (e.g. a binaural or plain stereo setup), the SAOC system behaves as a decoder, using its own rendering engine. Another key feature is that the SAOC parametric data from different streams can be merged at parameter level to allow for the combination of SAOC streams, similar to the functionality of a multi-point control unit (MCU).

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

ISO and IEC take no position concerning the evidence, validity and scope of this patent right. The holder of this patent right has assured ISO and IEC that he/she is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO and IEC. Information may be obtained from:

Qualcomm Incorporated
6455 Lusk Blvd
US-San Diego, CA 92121-2779

Fraunhofer Institute for Integrated Circuits IIS
Leonrodstrasse 68
DE-80636 München

LG Electronics
16 Woomyeon-Dong Seocho-Gu
KR-Seoul 137-724

Koninklijke Philips Electronics N.V.
High Tech Campus 44
NL-5656 AE, Eindhoven

Electronics and Telecommunications Research Institute
161 Gajeong-dong Yuseong-gu
KR-Daejeon 305-350

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

Information technology — MPEG audio technologies — Part 2: Spatial Audio Object Coding (SAOC)

1 Scope

This document specifies the reference model of the spatial audio object coding (SAOC) technology that is capable of recreating, modifying and rendering a number of audio objects based on a smaller number of transmitted channels and additional parametric data.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

3 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1

audio object

input audio signal consisting of one, two or multiple channels, including multi-channel background object (MBO)

3.2

frame

time segment (3.15) to which SAOC processing is applied according to the data conveyed in the corresponding SAOCFrame() or SAOCDEFrame() syntax elements

3.3

hybrid filterbank

structure, consisting of a quadrature mirror filter (QMF) bank and oddly modulated Nyquist filter banks, used to transform time domain signals into *hybrid subband* (3.5) samples

3.4

hybrid filtering

filtering step on a quadrature mirror filter (QMF) subband signal resulting in multiple *hybrid subbands* (3.5)

Note 1 to entry: The resulting hybrid subbands can be non-consecutive in frequency.

3.5

hybrid subband

subband obtained after *hybrid filtering* (3.4) of a quadrature mirror filter (QMF) subband

Note 1 to entry: The hybrid subband can have the same time/frequency resolution as a QMF subband.

3.6

input channel

input audio channel corresponding to the channels of an *audio object* (3.1)

3.7

output channel

audio channel corresponding to a specific speaker

Note 1 to entry: Channel abbreviations and loudspeaker positions are given in Table 1.

3.8

parameter band

one or more *hybrid subbands* (3.5) applicable to one parameter

3.9

parameter time slot

specific *time slot* (3.16) for which the parameter is defined

3.10

parameter set

parameters associated with a specific *parameter time slot* (3.9)

3.11

parameter subset

parameters associated with a specific *parameter time slot* (3.9) and a specific one-to-two (OTT) box or two-to-three (TTT) box

3.12

processing band

one or more *hybrid subbands* (3.5) defining the finest frequency resolution that could be controlled by the parameters

3.13

QMF bank

bank of complex exponentially modulated filters

3.14

QMF subband

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

3.15

time segment

group of consecutive *time slots* (3.16)

3.16

time slot

finest resolution in time for spatial *audio object* (3.1) coding (SAOC) time borders

Note 1 to entry: One time slot equals one subsample in the hybrid quadrature mirror filter (QMF) domain.

4 Notations and abbreviated terms

4.1 Notation

The description of the SAOC system uses the following notations:

- vectors are indicated by bold lower-case names, e.g. **vector**;
- matrices (and vectors of vectors) are indicated by bold upper-case single letter names, e.g. **M**;
- variables are indicated by italic, e.g. *variable*;
- functions are indicated as *func(x)*.

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

4.2 Operations

4.2.1 Scalar operations

- x^* is the complex conjugate of x .
- $y = \log_{10}(x)$ is the base-10 logarithm of x .
- $y = \min(\dots)$ is the minimum value in the argument list.
- $y = \max(\dots)$ is the maximum value in the argument list.
- $\exp(x)$ is the exponential function of x .

4.2.2 Vector and matrix operations

- $\mathbf{m} = \text{diag}(\mathbf{M})$ is main diagonal of matrix, **M**.
- $\mathbf{y} = \text{sort}(\mathbf{x})$ is equal to the sorted vector **x**, where the elements of **x** are sorted in ascending order.
- $y = \text{trace}(\mathbf{M})$ is sum of all diagonal elements of matrix, **M**.
- $\mathbf{M}^*$ is the complex conjugate transpose of **M**.

4.3 Constants

- ε is a constant to avoid division by and logarithm of zero, e.g. $\varepsilon = 10^{-9}$.
- $\mathbf{0}_{A \times B}$ is a matrix of size $A \times B$ consisting of zeros.
- $\mathbf{I}_A$ is an identity matrix of size $A \times A$.

4.4 Variables

- $a_{i,y}^{l,m}$ is the virtual speaker transfer function, defined for binaural output channel, i , audio object, y , and all parameter time slots, l , and processing bands, m .
- D** is the downmix matrix.

$\mathbf{D}_{\text{CLD}}$	is the three-dimensional matrix holding the dequantized, and mapped CLD data for every OTT box, every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{ICC}}$	is the three-dimensional matrix holding the dequantized, and mapped ICC data for every OTT or TTT box, every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{CPC}_1}, \mathbf{D}_{\text{CPC}_2}$	are the three-dimensional matrices holding the dequantized, and mapped first and second CPC data for every TTT box, every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{CLD}_1}, \mathbf{D}_{\text{CLD}_2}$	are the three-dimensional matrices holding the dequantized, and mapped first and second CLD data for every TTT box, every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{DCLD}}$	is the matrix holding the dequantized, and mapped DCLD data for every input channel and every parameter set.
$\mathbf{D}_{\text{DMG}}$	is the matrix holding the dequantized, and mapped DMG data for every input channel and every parameter set. If DMG data contains information for more than one downmix channel, $\mathbf{D}_{\text{DMG}}$ is a three-dimensional matrix holding the dequantized, and mapped DMG data for every input channel, every downmix channel and every parameter set.
$\mathbf{D}_{\text{IOC}}$	is the four-dimensional matrix holding the dequantized, and mapped IOC data for every input channel pair, every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{NRG}}$	is the two-dimensional matrix holding the dequantized, and mapped NRG data for the highest energy within every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{OLD}}$	is the three-dimensional matrix holding the dequantized, and mapped OLD data for every input channel, every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{PDG}}$	is the three-dimensional matrix holding the dequantized, and mapped PDG data for every downmix channel, every parameter set and M_{proc} bands.
$\mathbf{D}_{\text{BGO}}$	is the downmix sub-matrix for BGOs.
$\mathbf{D}_{\text{FGO}}$	is the downmix sub-matrix for FGOs.
$H_{i,\{L,R\}}^m$	is the HRTF parameter which represents the average level with respect to the left and right ear $\{L, R\}$ for the HRTF database index i , and all processing bands m .
$\text{idxXXX}(\dots, \dots)$	is a three-dimensional matrix holding the Huffman, and delta decoded indices. XXX can be any of OLD, IOC, NRG, DCLD, DMG, PDG.
K	is the number of hybrid subbands.
L	is the number of parameter sets.
M	is the number of downmix channels.
M_{proc}	is the number of processing bands.
M_{QMF}	is the number of QMF subbands depending on sampling frequency.

$\mathbf{M}^{l,m}$	is the OTN/TTN upmix matrix for the prediction mode of operation.
$\mathbf{M}_{\text{Energy}}^{l,m}$	is the OTN/TTN upmix matrix for the energy mode of operation.
$\mathbf{M}_1^{n,k}, \mathbf{M}_2^{n,k}$	are the time and frequency variant pre-matrices, defined for all time slots, n , and all hybrid subbands, k .
$\mathbf{M}_{\text{ren}}^{l,m}$	is the time and frequency variant rendering matrix, defined for all parameter time slots, l , and all processing bands, m .
$\mathbf{G}_{\text{DE}}^{l,m}$	is the time and frequency variant parametric processing matrix, defined for all parameter time slots, l , and all processing bands, m .
$\mathbf{M}_{\text{DE}}^{l,m}$	is the time and frequency variant residual processing matrix, defined for all parameter time slots, l , and all processing bands, m .
m_{BGO}	is the modification gain for BGOs.
m_{FGO}	is the modification gain for FGOs.
m_{G}	is the decoder limited modification gain.
$m_{\text{G}}^{\text{input}}$	is the input modification gain.
N	is the number of SAOC input channels of audio objects.
N_{FGO}	is the number of FGOs.
N_{EAO}	is the number of EAO channels.
N_{MPS}	is the number of MPS output channels.
N_{HRTF}	is the number of different HRTFs in the HRTF database.
N_{g}	is the number of groups of downmix signals.
N_{g}^q	is the number of downmix signals assigned to group $\mathbf{g}_q$, defined for all group indices, q .
$\mathbf{g}_q$	is a vector with the indices of the downmix signals assigned to the same group, defined for all group indices, q .
P	is the frame length.
$\mathbf{W}_{\text{ADG}}^{l,m}$	is the time and frequency variant matrix including ADGs, defined for all parameter time slots, l , and all processing bands, m .
$\mathbf{W}_h^{l,m}$	is the time and frequency variant sub-rendering matrix, defined for OTT box, h , (of the MPS “5-1-5” tree-structure), all parameter time slots, l , and all processing bands, m .

$\mathbf{W}_{\text{PDG}}^{l,m}$	is the time and frequency variant matrix including PDGs, defined for all parameter time slots, l , and all processing bands, m .
$\mathbf{s}^{n,k}$	is a vector with the hybrid subband (encoder) input channels, defined for all time slots, n , and all hybrid subbands, k .
$\mathbf{x}^{n,k}$	is a vector with the hybrid subband (transcoder/decoder) input signals (downmix and residuals), defined for all time slots, n , and all hybrid subbands, k .
$\mathbf{y}^{n,k}$	is a vector with the (transcoder/decoder) output hybrid subband signals, which are fed into the hybrid synthesis filter banks, defined for all time slots, n , and all hybrid subbands, k .
ϕ_i^m	is the HRTFs parametric representation of the average phase difference, defined for the HRTF database index, i , and all processing bands, m .

4.5 Abbreviated terms

ADG	arbitrary downmix gain
BGO	background object
CLD	channel level difference; describes the energy difference between two channels
CPC	channel prediction coefficient; used for recreating three or more channels from two channels
DCLD	downmix channel level difference; describes the gain differences of objects contributing to the left and right downmix channel in case of a stereo downmix
DCU	distortion control unit
DE	dialogue enhancement
DMG	downmix gain; gains applied to each object before downmixing
EAO	enhanced audio object
FGO	foreground object
HRTF	head related transfer function
ICC	inter channel correlation; describes the correlation between two channels
IOC	inter object correlation; describes the correlation between two channels of audio objects
LD	low delay
MBO	multi-channel background object
MCU	multi-point control unit
MPS	mpeg surround

N/A	not applicable
NRG	absolute object energy; specifies the absolute energy of the object with the highest energy for the corresponding parameter band
OLD	object level difference, describes intensity differences between one object and the object with the highest energy for the corresponding parameter band
OTN	conceptual "One-To-N" unit that takes one channel as input and produces N channels as output
OTT	conceptual "One-To-Two" unit that takes one channel as input and produces two channels as output
PDG	post(processing) downmix gains; describes intensity differences between the encoder-generated downmix and the post(processed) downmix for the corresponding parameter band
QMF	quadrature mirror filter
SAC	spatial audio coding
SAOC	spatial audio object coding
TTN	conceptual "Two-To-N" unit that takes two channels as input and produces N channels as output
TTT	conceptual "Two-To-Three" unit that takes two channels as input and produces three channels as output

Table 1 — Channel abbreviations and loudspeaker positions

Channel abbreviation	Loudspeaker position	Figure
L	Left front	
R	Right front	
C	Center front	
LFE	Low frequency enhancement	
Ls	Left surround	
Rs	Right surround	

5 SAOC overview

5.1 General

Spatial audio object coding (SAOC) is a parametric multiple object coding technique. It is designed to transmit a number of audio objects in an audio signal that comprises M channels. Together with this backwards compatible downmix signal, object parameters are transmitted that allow for recreation and manipulation of the original object signals. An SAOC encoder produces a downmix of the object signals at its input and extracts these object parameters. The number of objects that can be handled is in principle not limited.

The object parameters are quantized and coded efficiently into an SAOC bitstream.

The downmix signal can be compressed and transmitted without the need to update existing coders and infrastructures. The object parameters, or SAOC side information, are transmitted in a low bitrate side channel, e.g. the ancillary data portion of the downmix bitstream.

On the decoder side, the input objects are reconstructed and at the same time rendered to a certain number of playback channels. The rendering information containing reproduction level and panning position for each object is user supplied or can be extracted from the SAOC bitstream (e.g. preset information). The rendering information can be time variant. Output scenarios can range from mono to multi-channel (e.g. 5.1) and are independent from both, the number of input objects and the number of downmix channels. Binaural rendering of objects is possible including azimuth and elevation of virtual object positions. An optional effects interface allows for advanced manipulation of object signals, besides level and panning modification.

The objects themselves can be mono signals, stereophonic signals, as well as multi-channel signals (e.g. 5.1 channels). Typical downmix configurations are mono and stereo.

5.2 Basic structure of the SAOC transcoder/decoder

The SAOC transcoder/decoder module described below may act either as a stand-alone decoder or as a transcoder from an SAOC to an MPS bitstream, depending on the intended output channel configuration. Table 2 illustrates the differences between the two modes of operation.

Table 2 — Operation modes of the SAOC

Output signal configuration	# of output channels	# of input channels	SAOC module mode	SAOC module output	MPS decoder required
Mono/stereo/binaural/ 3-channel configuration	1, 2 or 3	1, 2 or 3	Decoder	PCM output	No
Multi-channel configuration	>2	1 or 2	Transcoder	MPS bitstream, downmix signal	Yes

Figure 1 shows the basic structure of the SAOC transcoder/decoder architecture. The residual processor extracts the EAOs from the incoming downmix using the residual information contained in the SAOC bitstream. The downmix pre-processor processes the regular audio objects. The EAOs and processed regular audio objects are combined to the output signal for the SAOC decoder mode or to the MPS downmix signal for the SAOC transcoder mode. The detailed descriptions of these processing blocks are given in the corresponding subclauses, namely, 7.5 and 7.5.4, which describe the SAOC transcoder/decoder functionality and 7.6 explains handling of enhanced audio objects and residual processing.

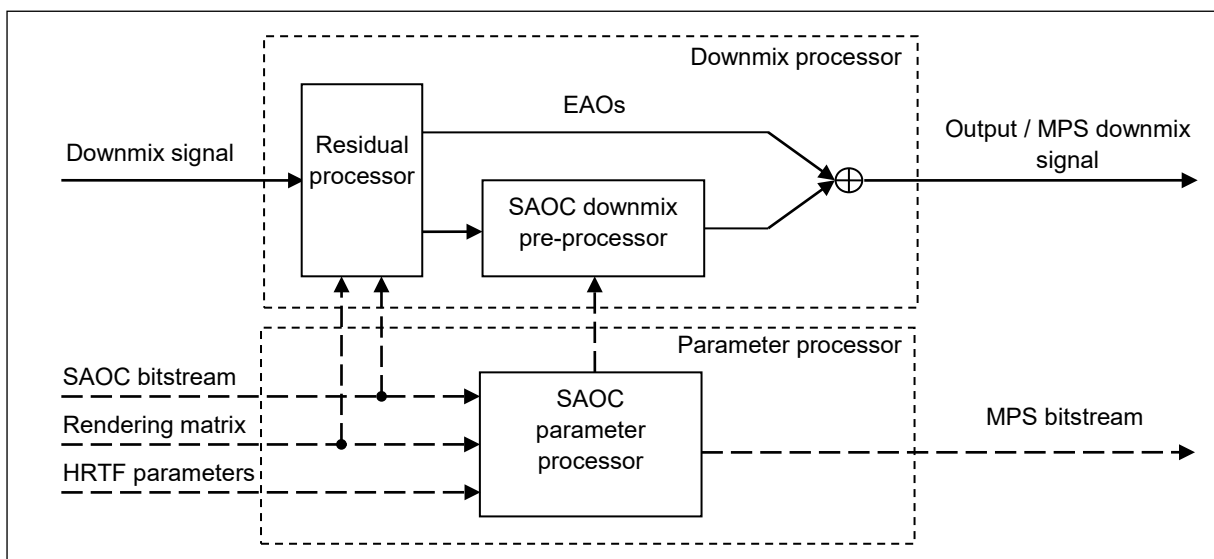


Figure 1 — Overall structure of the SAOC transcoder/decoder architecture

Figure 2 (left) shows a block diagram of an SAOC transcoder unit. It consists of an SAOC parameter processor and a downmix processor module. The SAOC parameter processor decodes the SAOC bitstream and has furthermore a user interface from which it receives additional input in form of generally time variant rendering information. It provides steering information for the downmix processor. The SAOC transcoder outputs an MPS bitstream and downmix signal, as an input to the MPS decoder. In case of a mono downmix, the downmix pre-processor leaves the downmix signal unchanged. However, in case of a stereo downmix, it is functional to pre-process the downmix signal to allow more flexible object panning than is supported by the MPS rendering engine alone. In case of a mono/stereo/binaural/multi-channel output configuration, the SAOC system works in decoder mode and MPS decoding is omitted [see Figure 2 (right)]. Here, the downmix processing module directly provides the output signal.

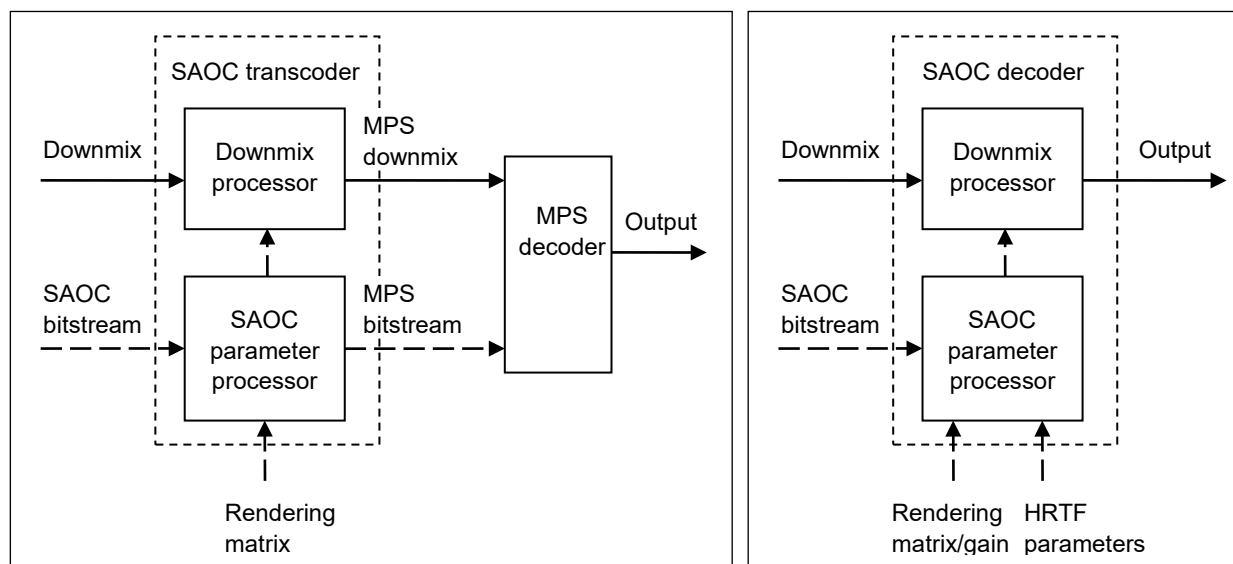


Figure 2 — Block diagrams of the SAOC transcoder (left) and decoder (right) processing modes

5.3 Tools and functionality

5.3.1 General SAOC tools

5.3.1.1 Overview

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

5.3.1.2 Binaural decoding

The SAOC system can be operated in a binaural mode. This enables a multi-channel impression over headphones by means of head related transfer function (HRTF) filtering.

5.3.1.3 Efficient multipoint control unit support

In order to use the SAOC concept for teleconferencing applications, a multipoint control unit (MCU) functionality of combining the signals of several communication partners without decoding/re-encoding the corresponding audio objects is provided. The MCU combines the input SAOC side information streams into one common SAOC bitstream in a way that the parameters representing all audio objects from the input bitstreams are included in the resulting output bitstream. These calculations are performed in the parameter domain without the need to analyse the downmix signals and, therefore, introduce no additional delay in the signal processing chain.

5.3.1.4 External downmix

The SAOC system is capable of handling not only encoder-generated downmixes but also post(processed) downmixes supplied to the encoder in addition to the input audio object signals. In this case, post downmix gains (PDGs) are calculated in the encoder and conveyed as a part of the SAOC bitstream. The difference of the downmix signals is compensated for at the SAOC decoder side.

5.3.1.5 Multichannel background object

The audio input to a SAOC encoder can contain a so-called multi-channel background object (MBO). Generally, the MBO can be considered as a complex sound scene involving a large and often unknown number of sound sources, for which no controllable rendering functionality is required. The MBO is represented by a downmix of the MPS encoded complex sound scene and corresponding MPS parameters.

5.3.1.6 Enhanced audio object processing

A special "Karaoke-type" application scenario requires a total suppression of specific objects, typically the lead vocals, while keeping the perceptual quality of the background sound scene unharmed. High sound quality is assured by the incorporation of residual coding enabling a better separation of the background object and foreground objects. The EAO processing mode supports reproduction of both EAO and regular objects exclusively and arbitrary mixtures of these object groups.

5.3.1.7 Distortion control unit

The distortion control unit is incorporated into the SAOC system in order to provide a flexible control for users and audio content providers over the SAOC rendering functionality and audio output quality.

5.3.1.8 Predefined rendering information

The SAOC system is capable of starting playback with some initial predefined settings which can be stored and/or transmitted in SAOC bitstream. These settings can be dynamically updated. The SAOC system allows instantaneous switching between them if more than one set of predefined settings is available.

5.3.1.9 Effects interface

The SAOC effects interface operates on the downmix and therefore is part of the downmix processor of the SAOC transcoder or decoder. The effects interface allows objects or linear combinations of objects to be extracted from the downmix for effects processing. There are two types of effects processing interfaces. The first type, referred to as the insert effects interface, allows effects processing to individual objects in the downmix. The second type, referred to as the send effects interface, allows effect processing on individual objects or linear combinations thereof.

5.3.2 High quality, low power and low delay

The SAOC decoder can be implemented in a high quality (HQ) version, a low power (LP) version and a low delay (LD) version. The main differences are outlined by Table 3 and given in detail in 7.13 and 7.14.

Table 3 — Outline of difference between the HQ, LP and LD SAOC system

Tool or functionality	HQ system	LP system	LD system
Filterbank	Complex valued QMF	Partially complex valued QMF	LD QMF, no Nyquist filterbank
Aliasing reduction	Not applicable	Tool operates on the real-valued part of the frequency range	Not applicable
Residual coding	Supported over the entire frequency range	Only supported over the complex valued part of the frequency range	Not supported
Decorrelators	Supported	Not supported for stereo downmix	Supported

5.4 Delay and synchronization

5.4.1 Overview

The SAOC decoder introduces a delay when processing the time domain signal coming from a downmix decoder. Depending on whether the SAOC module is working as a decoder or as a transcoder for a multichannel renderer (MC-SAOC), i.e. MPS decoder, two different cases are to be taken into account.

For all the different cases described in this subclause, the transmission of the SAOC side information with respect to the transmission of the coded downmix signal is done in such a manner that there is no need to further delay the downmix signal before the SAOC processing. This means that synchronization of the spatial data and downmix is achieved at the SAOC decoder/transcoder, following the temporal relationships described in Clause 8.

5.4.2 High quality and low power processing

5.4.2.1 SAOC decoding mode

The SAOC decoder (mono, stereo, 3-channel or binaural up-mix modes) introduces a total delay of 1 281 time domain samples for the high quality (HQ) mode and 1 601 samples for the low power (LP)

mode. As shown in Figure 3, the analysis filterbank as outlined in 7.3 introduces a delay of 704 samples, while the synthesis filterbank introduces 257 for HQ and 577 for LP. The analysis filterbank processing delay consists of the QMF and hybrid processing delays, 320 and 384 time domain samples, respectively. If no real to complex conversion is performed, it shall be replaced by a delay line. This leads to 320 time samples which are added on top the analysis processing delay for both HQ and LP decoders. The synthesis filterbank introduces 257 samples delay which is introduced by the QMF synthesis filtering. The hybrid synthesis does not introduce further delay. For the LP decoder, the complex to real conversion adds 320 time domain samples to the synthesis processing delay.

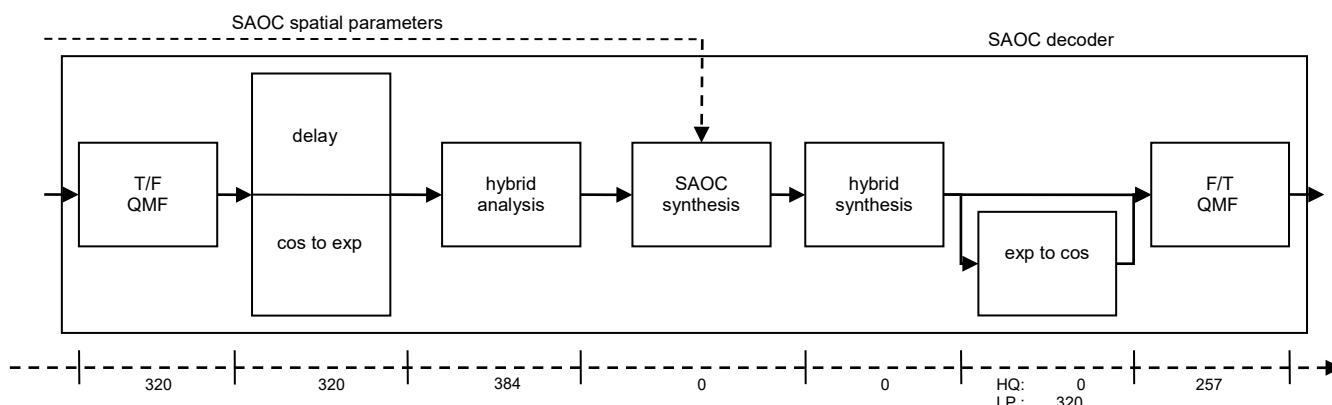


Figure 3 — Delay for the different parts of the SAOC decoder

5.4.2.2 SAOC transcoding mode

The MPS renderer introduces a delay when processing the downmix from the SAOC transcoder. Two cases may be distinguished: mono and stereo downmix.

In case of a mono downmix, the signal from the downmix decoder is passed directly to the MPS decoder as no further processing is applied to the downmix, as depicted in Figure 4. The MC-SAOC processing delay shall be equal to the one given by the SAOC decoder, as the SAOC to MPS parameter processing does not introduce any additional delay.

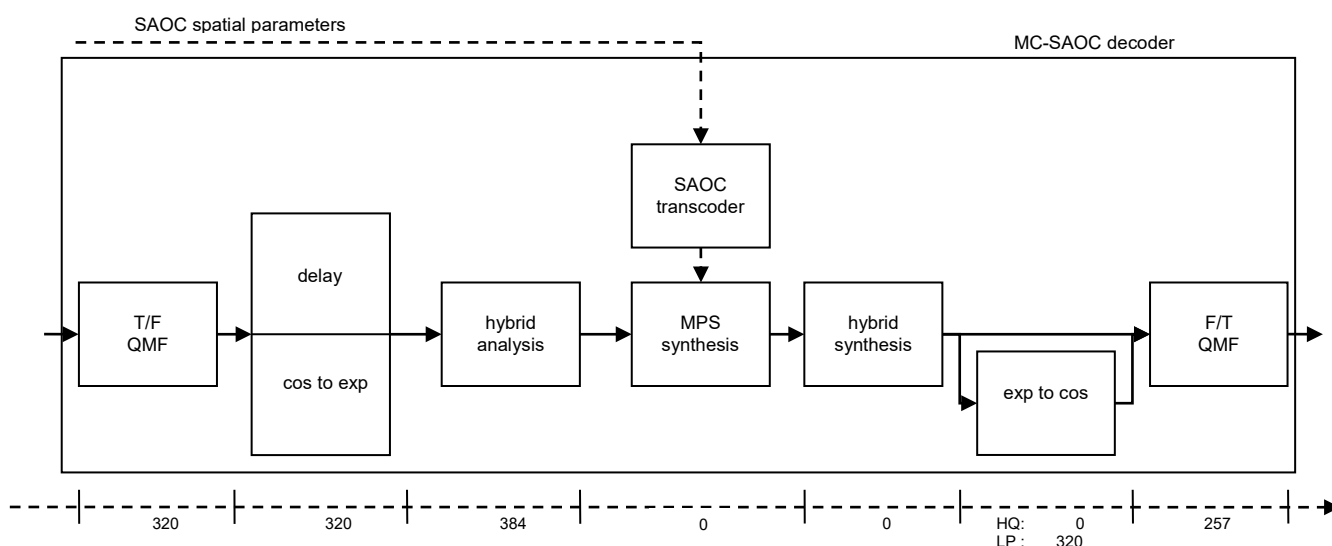


Figure 4 — Delay for the different parts of the MC-SAOC decoder with mono downmix signal

In case of a stereo downmix, the SAOC transcoder processes the core coder signal to adapt it to the subsequent MPS decoding, as shown in Figure 5. The delay of the MPS renderer shall be added on top of the SAOC processing delay. Both the SAOC transcoder and the MPS decoder introduce a delay of

1 281 samples for the HQ mode or of 1 601 samples for the LP mode. Their analysis/synthesis delay is distributed as described above for the SAOC decoder.

If both modules are not integrated, i.e. they interface via the time domain, the total processing delay shall be the sum of the delays introduced by each module plus buffering needed for synchronization of the MPS parameters to the downmix.

The size of buffer B (spatial parameters buffer) is a multiple of the frame length. The downmix buffer A is needed to synchronize the delayed bitstream and processed downmix. To achieve synchronization, Formula (1) should be met; both buffer sizes are given in time domain samples:

$$B = N \cdot \text{Frame length so that } B \geq 1\,281 \text{ (HQ) or } B \geq 1\,601 \text{ (LP); } N=0,1,2,3,\dots \quad (1)$$

$$A(\text{HQ}) = B - 1281 \text{ or } A(\text{LP}) = B - 1601$$

Given an interface via the hybrid QMF domain, the overall processing delay of the multi-channel rendering is equal to the delay of the SAOC decoding mode.

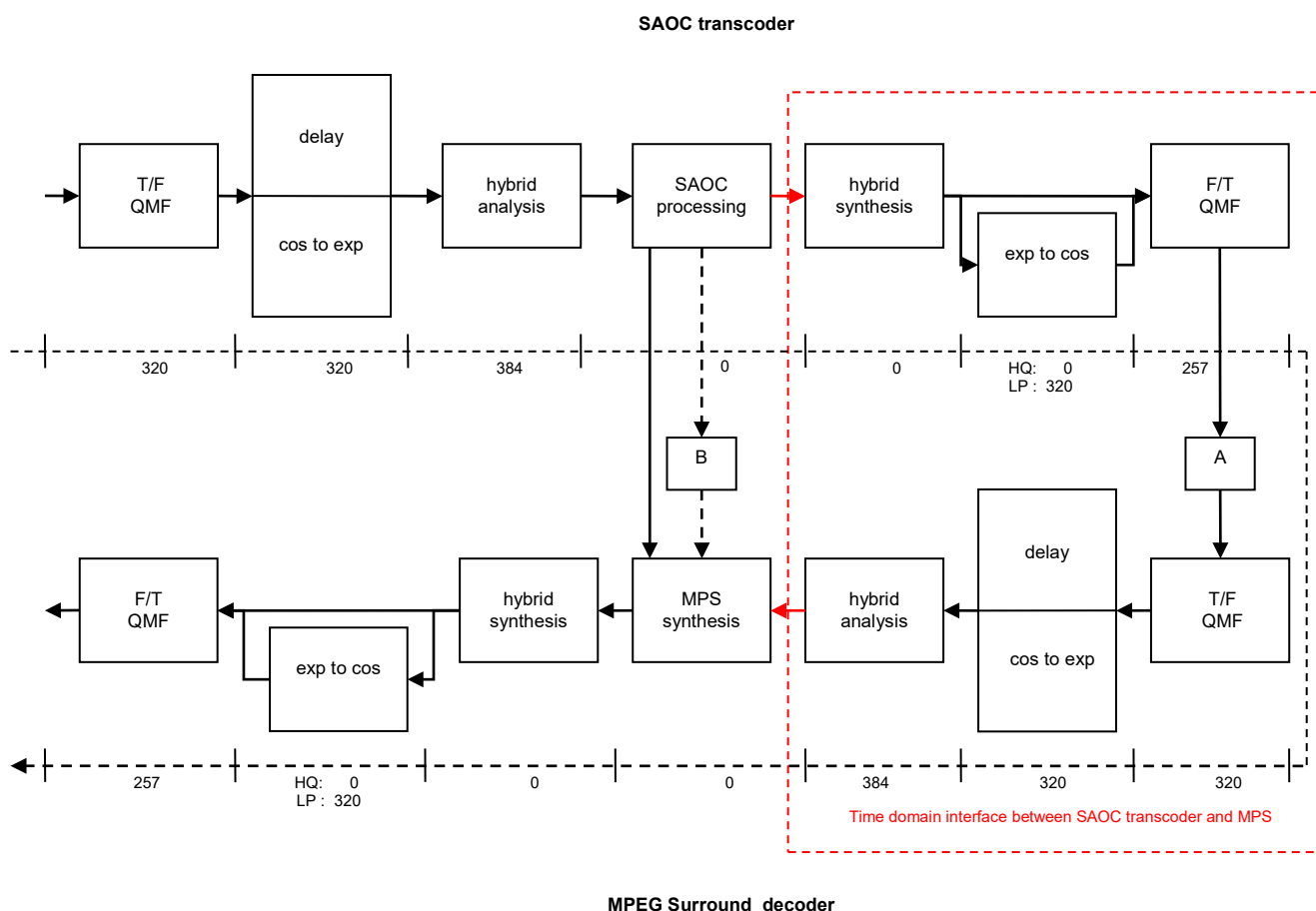


Figure 5 — Delay for the different parts of the MC-SAOC decoder with stereo downmix signal

5.4.2.3 Connection to an arbitrary core coder

If the (MC-)SAOC decoder is connected with an arbitrary downmix coder (including HE-AAC) via the time domain, as shown in Figure 6, the additional delay introduced by the (MC-)SAOC decoder processing is described in the previous subclauses.

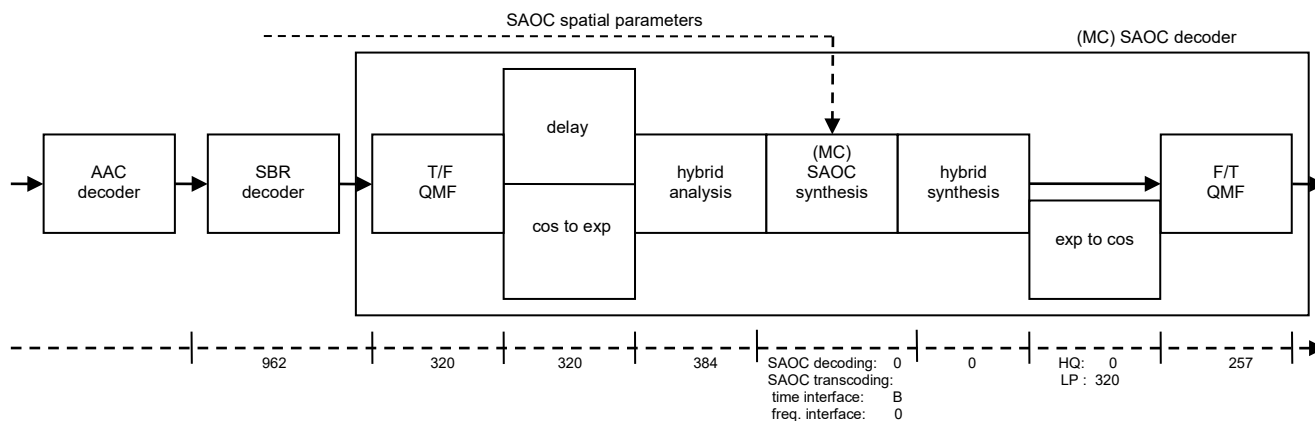


Figure 6 — Delay when connecting (MC-)SAOC in the time-domain for an arbitrary core codec (including HE-AAC)

If the (MC-)SAOC decoder is directly connected with a High Efficiency AAC decoder via the QMF domain, as shown in Figure 7, the delay shall be reduced as outlined in ISO/IEC 23003-1:2007, 4.5 for a MPS decoder. The only additional delay is introduced by the real to complex converter for a LP decoder or the delay compensation for a HQ decoder. No hybrid analysis delay is introduced due to the fact that the look-ahead of 384 time domain samples is already available on the SBR tool of HE-AAC, as described in ISO/IEC 14496-3:2009, 8.A.3.

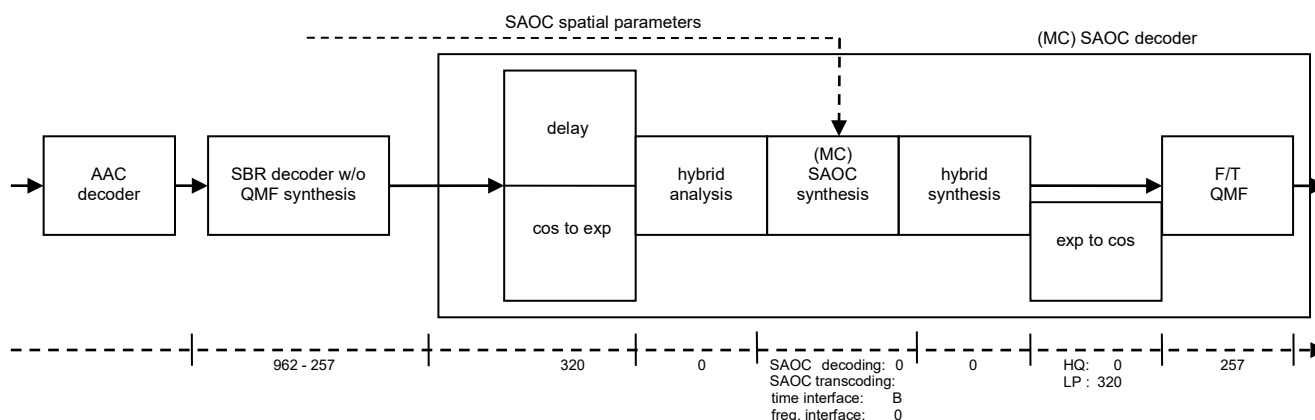


Figure 7 — Delay when connecting (MC-)SAOC with HE-AAC in the QMF domain

5.4.3 LD processing

5.4.3.1 SAOC Decoding

As outlined in 7.13 for the LD-SAOC processing, the Hybrid QMF filterbank is substituted by a LD QMF. In addition, no LP mode is allowed and hereafter no complex conversion is to be performed. As a result, the analysis and synthesis processing delays shall only depend on the LD QMF filtering, i.e. 160 time domain samples for analysis and 96 for synthesis. This is shown in Figure 8.

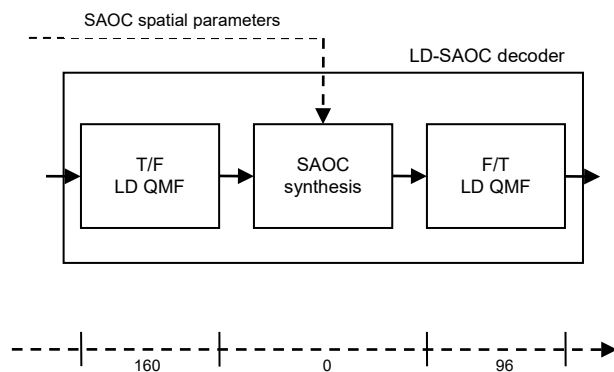


Figure 8 — Delay for the different parts of the LD-SAOC decoder

5.4.3.2 SAOC Transcoding

Equally to the non-LD MC-decoders, depending on the number of downmix channels (mono/stereo), the processing for the LD-MC-SAOC rendering scenarios is performed differently. Due to the delay constraints for a LD-SAOC system, no time domain interface is allowed for the stereo processing case. Hereafter, both mono and stereo downmix cases may be equally described with respect to the systems delay, matching the SAOC decoder delay distribution, as depicted in Figure 9 and Figure 10.

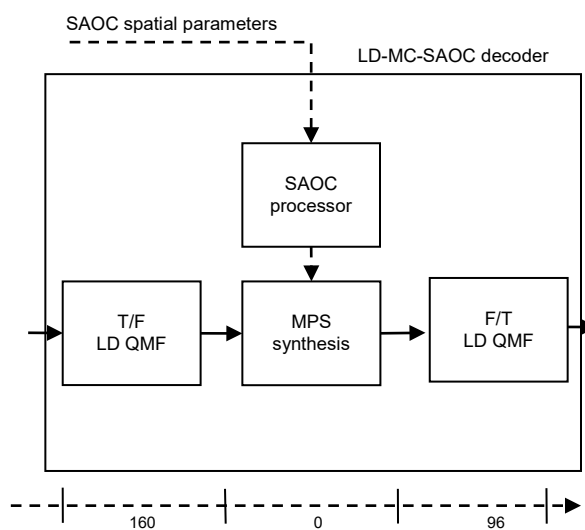


Figure 9 — Delay for the different parts of the LD-MC-SAOC decoder with mono downmix signal

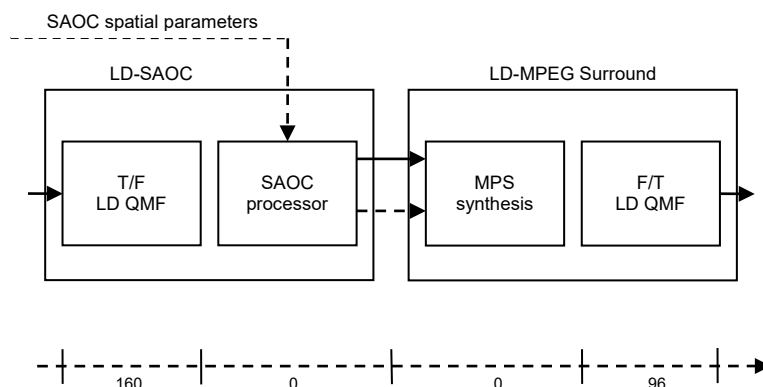


Figure 10 — Delay for the different parts of the LD-MC-SAOC decoder with stereo downmix signal

5.4.3.3 LD-(MC-)SAOC connection to LD core codec

If the LD-(MC-)SAOC decoder is connected with a LD downmix coder (e.g. AAC-LD or AAC-ELD, with/-out SBR) via the time domain, as shown in Figure 11, the additional delay introduced by the LD-(MC-)SAOC decoder processing is described above.

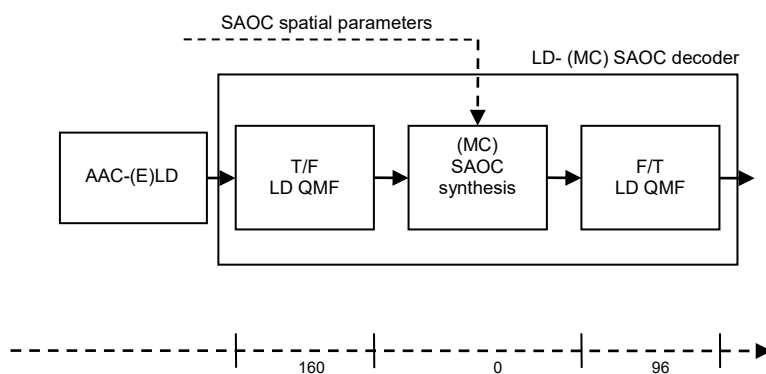


Figure 11 — Delay when connecting LD-(MC-)SAOC in the time-domain for a LD core codec (e.g. AAC-LD or AAC-ELD)

If the LD-(MC-)SAOC decoder is connected with a LD core coder via the frequency domain (e.g. AAC-ELD with SBR), as shown in Figure 12, the additional delay introduced by the LD-(MC-)SAOC decoder processing shall be reduced with respect to the description above, as no analysis is performed in the LD-(MC-)SAOC decoder.

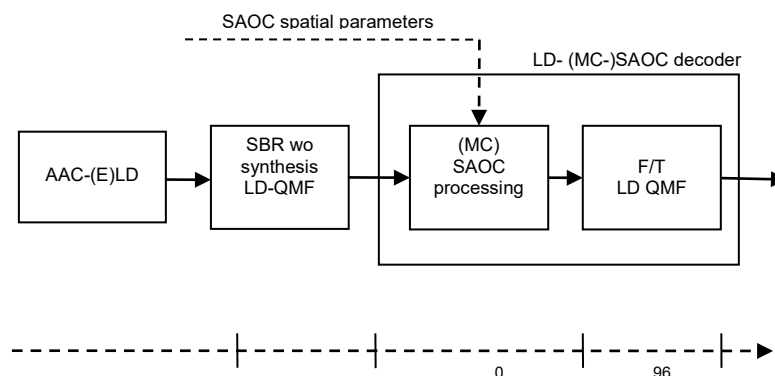


Figure 12 — Delay when connecting LD-(MC-) SAOC in the frequency-domain for a LD core codec (e.g. AAC-ELD with SBR)

5.5 SAOC Profiles and levels

5.5.1 General

This subclause defines profiles and their levels for SAOC.

Complexity units are defined to give an approximation of the decoder complexity in terms of processing power and RAM usage required for the SAOC decoding/transcoding process. The approximated processing power is given in “processor complexity units” (PCU), specified in MOPS. The approximated RAM usage is given in “ram complexity units” (RCU), specified in kWords (1 000 words).

Note that the following three profiles are independent from each other and are not intended to be interoperable on a bitstream or decoder level. Due to the underlying concept of SAOC, the number of output channels is independent of the bitstream. Depending on the decoder capabilities, the output channel configuration is selected via the user interface of the decoder.

5.5.2 Baseline profile

The baseline profile is the generic SAOC profile for all applications that are not delay critical. Within this profile, all SAOC decoders use the hybrid QMF filterbank and allow rendering to at least two output channels, based on the maximum number of supported downmix channels of two. Rendering to more than two output channels is optional.

An envisioned application scenario for the baseline profile includes, e.g. the following interactive re-mix application: a stereo (or mono) compressed signal or “track” can be provided to the consumer along with SAOC data describing the objects present in the track. With this, the user can create his or her own “re-”mix of the music or the sounds in the stereo (or mono) track, by playing back such object-based tracks with a control similar to that of a multi-track mixing desk and are therefore free to adjust relative level, spatial position, etc. of instruments, sounds or dialogue according to their preferences.

NOTE ISO/IEC 23000-12 defines several brands that refer to the SAOC baseline profile.

5.5.3 LD profile

The SAOC-LD profile is targeted at applications that require low-delay operation. Within this profile, all SAOC decoders use the LD-QMF filterbank and allow rendering to at least two output channels. Rendering to more than two output channels is optional, with one exception: Level 3 requires the support for at least 5 output channels and includes a low-delay version of MPS as the rendering engine.

An envisioned application scenario for the LD profile includes, e.g. the following telecommunication application: existing monophonic infrastructures for telecommunication may be extended in their functionality easily by introducing SAOC with a single downmix channel. Telecommunication terminals equipped with an SAOC extension are able to pick up several sound sources (objects), mix them into a single (monophonic) downmix signal which is transmitted in a compatible way by using the existing coders (e.g. speech coders). The side information is conveyed in a hidden, backward compatible way. While such advanced terminals produce an output object stream containing several objects, legacy terminals will simply be considered as producing an object stream with a single embedded object.

5.5.4 Dialogue enhancement (SAOC-DE) profile

The dialogue enhancement profile is targeted at broadcast applications that desire dialogue enhancement functionality. Within this profile, all SAOC decoders use the hybrid QMF filterbank and allow rendering to mono, stereo or 3-channel output. The number of output channels is equal with the number of downmix channels.

An envisioned application scenario for the SAOC-DE profile includes, e.g. the following broadcasting application: by adding only a moderate bitrate overhead to a usual broadcast transport bitstream (the dialogue and background signals are already mixed together), the user will be provided with the option to enhance or attenuate the dialogue level in relation to all other signals contained in the mix. The side information is conveyed in a backwards-compatible way together with the encoded audio signal (e.g. using MPEG-4 AAC or HE-AAC) and therefore the legacy decoders will simply playback the default broadcast mixed signal.

The definition of the profiles and levels of SAOC is given in Table 4.

Table 4 — SAOC profiles and levels

Profiles	Baseline profile				SAOC-DE profile		LD profile		
Levels	1	2	3	4	1	2	1	2	3
Hybrid QMF bank	X	X	X	X	X	X	—	—	—
LD-QMF bank	—	—	—	—	—	—	X	X	X
Max number of residual channels	0	2	4	4	0	3	—	—	—
Max sampling rate [kHz]	48	48	48	96	48	48	48	48	48
Max number of objects	8	16	32	32	6	6	8	32	32
Max number of downmix channels	2	2	2	2	3	3	1	2	2
Min number of required output channels ^a	2	2	2	2	1	1	2	2	5
Use of decorrelator	yes	yes	yes	yes	no	no	yes	yes	yes
PCU HQ decoder	12.2	20.4	33.9	67.8	12.4	22.1	8.4	20.7	39.3 ^b
PCU LP decoder	6.6	12.2	23.0	46.0	11.4	21.0	N/A	N/A	N/A
PCU addition for transcoding	1.1	1.1	1.1	2.3	N/A	N/A	0.7	1.1	N/A
PCU reduction for integrated transcoding	-6.8	-6.8	-6.8	-6.8	N/A	N/A	-3.6	-6.5	N/A
RCU HQ decoder	5.7	9.8	13.5	17.5	6.3	12.3	3.6	4.2	17.9 ^c
RCU LP decoder	4.8	5.4	5.7	10.3	7.3	7.9	N/A	N/A	N/A
RCU reduction for integrated transcoding	-1.3	-1.3	-1.3	-1.3	N/A	N/A	-0.6	-1.3	N/A
^a Every SAOC decoder has to support rendering of SAOC content for every number of output channels up to the maximum number, as selected by the user interface. If the number of output channels >2, MPS is used to render the output signals in baseline and LD profiles. In SAOC-DE profile, the number of output channels shall be equal with the number of downmix channels. ^b PCU number includes SAOC (20.7), transcoding overhead (1.1), reduction for integrated transcoding (-6.5) and an LD-MPS “5-2-5” decoder (24). ^c RCU number includes SAOC (4.2), reduction for integrated transcoding (-1.3) and an LD-MPS “5-2-5” decoder (15).									

The SAOC decoder type is defined by the four conditions:

- profile: baseline, LD or SAOC-DE profile;
- level: 1 to 4 (for baseline), 1 to 3 (for LD) or 1 to 2 (for SAOC-DE);
- HQ/LP: only applicable for baseline and SAOC-DE profiles;
- MPS transcoding support for baseline and LD profiles if the number of output channels >2.

For baseline and LD profiles:

- decoding to mono/stereo/binaural output. Transcoding to 5.1 is supported.

For dialogue enhancement (SAOC-DE) profile:

- decoding to mono/stereo/3-channel output. No transcoding to 5.1 is supported;
- multi-channel background object (MBO) processing, DCU processing, MCU processing, separation metadata and send effects interface are not supported;
- post-downmix gain processing (PDG) is supported only in combination with post(processing) re-application processing step;
- insert effects interface is supported only if no modification range control (MRC) settings are transported in the bitstream.

6 Syntax

6.1 Payloads for SAOC

Table 5 — Syntax of SAOCSpecificConfig()

Syntax	No. of bits	Mnemonic
SAOCSpecificConfig() {		
bsSamplingFrequencyIndex;	4	uimsbf
if (bsSamplingFrequencyIndex == 15) {		
bsSamplingFrequency;	24	uimsbf
}		
bsLowDelayMode;	1	uimsbf
bsFreqRes;	3	uimsbf
if (bsLowDelayMode == 0) {		
bsFrameLength;	7	uimsbf
} else {		
bsFrameLength;	5	uimsbf
}		
bsNumObjects;	5	uimsbf
for (i=0; i<bsNumObjects+1; i++) {		
bsRelatedTo[i][i] = 1;		
for(j=i+1; j<bsNumObjects+1; j++) {		
bsRelatedTo[i][j];	1	uimsbf
bsRelatedTo[j][i] = bsRelatedTo[i][j];		
}		
}		
bsTransmitAbsNrg;	1	uimsbf
bsNumDmxChannels;	1	uimsbf
if (bsNumDmxChannels == 1) {		
bsTttDualMode;	1	uimsbf
if (bsTttDualMode) {		
bsTttBandsLow;	5	uimsbf
bsTttBandsHigh = numBands;		Note
} else {		
bsTttBandsLow = numBands;		
}		
}		
bsPdgFlag;	1	uimsbf
bsOneIOC;	1	uimsbf
bsDcuFlag;	1	uimsbf
if (bsDcuFlag == 1) {		
bsDcuMandatory;	1	uimsbf
bsDcuDynamic;	1	uimsbf
if (bsDcuDynamic == 0) {		
bsDcuMode;	1	uimsbf
bsDcuParam;	4	uimsbf
}		
} else {		
bsDcuMandatory = 0;		
bsDcuDynamic = 0;		
bsDcuMode = 0;		
bsDcuParam = 0;		
}		
ByteAlign();		
SAOCExtensionConfig();		
}		

NOTE numBands is defined in Table 36 and depends on bsFreqRes.

Table 6 — Syntax of SAOCDESpecificConfig()

Syntax	No. of bits	Mnemonic
SAOCDEspecificConfig() {		
bsVersion;	4	uimbsbf
if (bsVersion == 0) {		
bsSamplingFrequencyIndex;	4	uimbsbf
if (bsSamplingFrequencyIndex == 15) {		
bsSamplingFrequency;	24	uimbsbf
}		
bsFreqRes;	3	uimbsbf
bsFrameLength;	7	uimbsbf
bsNumObjects;	3	uimbsbf
bsNumFGOs;	3	uimbsbf
for (i=0; i<bsNumObjects+1; i++) {		
bsRelatedTo[i][i] = 1;		
for(j=i+1; j<bsNumObjects+1; j++) {		
bsRelatedTo[i][j];	1	uimbsbf
bsRelatedTo[j][i] = bsRelatedTo[i][j];		
}		
}		
bsNumDmxChannels;	3	uimbsbf
bsPdgFlag;	1	uimbsbf
bsOneIOC;	1	uimbsbf
bsDeLimitFlag;	1	uimbsbf
if (bsDeLimitFlag == 1) {		
bsDeLimitFgo;	4	uimbsbf
bsDeLimitBgo;	4	uimbsbf
} else {		
bsDeLimitFgo = 0;		
bsDeLimitBgo = 0;		
}		
bsDcuFlag;	1	uimbsbf
if (bsDcuFlag == 1) {		
bsDcuMandatory;	1	uimbsbf
bsDcuDynamic;	1	uimbsbf
if (bsDcuDynamic == 0) {		
bsDcuMode;	1	uimbsbf
bsDcuParam;	4	uimbsbf
}		
} else {		
bsDcuMandatory = 0;		
bsDcuDynamic = 0;		
bsDcuMode = 0;		
bsDcuParam = 0;		
}		
ByteAlign();		
SAOCExtensionConfig();		
}		
}		
NOTE numBands is defined in Table 36 and depends on bsFreqRes.		

Table 7 — Syntax of SAOCExtensionConfig()

Syntax	No. of bits	Mnemonic
<pre> SAOCExtensionConfig() { SaocExtNum = 0; while (BitsAvailable() >= 8) { bsSaocExtType; SaocExtType[SaocExtNum] = bsSaocExtType; SaocExtNum++; cnt = bsSaocExtLen; if (cnt==15) { cnt += bsSaocExtLenAdd; } if (cnt==15+255) { cnt += bsSaocExtLenAddAdd; } bitsRead = SAOCExtensionConfigData(bsSaocExtType) nFillBits = 8*cnt-bitsRead; bsFillBits; } } </pre>	4 4 8 16 nFillBits	Note 1 uimsbf uimsbf uimsbf Note 2 bslbf
NOTE 1 The function BitsAvailable() returns the number of bits available to be read.		
NOTE 2 SAOCExtensionConfigData() returns the number of bits read.		

Table 8 — Syntax of SAOCExtensionConfigData(0)

Syntax	No. of bits	Mnemonic
SAOCEExtensionConfigData(0)		
{		
if (bsDeLimitFlag == 1) {		
bsDeLimitFgoEAO;	4	uimsbfb
bsDeLimitBgoEAO;	4	uimsbfb
} else {		
bsDeLimitFgoEAO = 0;		
bsDeLimitBgoEAO = 0;		
}		
if (bsDcuFlag == 1) {		
bsDcuFlag2;	1	uimsbfb
if ((bsDcuFlag2 == 1) && (bsDcuDynamic == 0)) {		
bsDcuMode2;	1	uimsbfb
bsDcuParam2;	4	uimsbfb
}		
}		
ResidualConfig();		
}		

Table 9 — Syntax of ResidualConfig()

Syntax	No. of bits	Mnemonic
ResidualConfig() { bsResidualSamplingFrequencyIndex ; bsResidualFramesPerSAOCFrame ; bsNumEAO ; for (i=0; i<bsNumEAO + 1; i++) { bsResidualPresent [i]; if (bsResidualPresent[i]) { bsResidualBands [i]; } bsTtnDualMode [i]; if (bsTtnDualMode[i]) { bsTtnBandsLow [i]; } else { bsTtnBandsLow[i] = numBands; } } }	 4 2 2 1 5 1 5 	 uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf Note
NOTE numBands is defined in Table 36 and depends on bsFreqRes.		

Table 10 — Syntax of SAOCExtensionConfigData(1)

Syntax	No. of bits	Mnemonic
SAOCExtensionConfigData(1) { }		

Table 11 — Syntax of SAOCExtensionConfigData(2)

Syntax	No. of bits	Mnemonic
SAOCExtensionConfigData(2) { SpatialSpecificConfig(); }		Note
NOTE SpatialSpecificConfig() is defined in ISO/IEC 23003-1:2007, Table 5.		

Table 12 — Syntax of SAOCExtensionConfigData(3)

Syntax	No. of bits	Mnemonic
SAOCExtensionConfigData(3) { }		

Table 13 — Syntax of SAOCExtensionConfigData(8)

Syntax	No. of bits	Mnemonic
SAOCExtensionConfigData(8) { ObjectMetaData(); }		

Table 14 — Syntax of ObjectMetaData()

Syntax	No. of bits	Mnemonic
ObjectMetaData() { for (i=0; i<bsNumObjects+1; i++) { bsNumByteMetaData[i]; for (j=0; j<bsNumByteMetaData[i]; j++) { bsMetaData[i][j]; } } }	8 8	uimsbf bslbf

Table 15 — Syntax of SAOExtensionConfigData(9)

Syntax	No. of bits	Mnemonic
SAOExtensionConfigData(9) { PresetConfig(); }		

Table 16 — Syntax of PresetConfig()

Syntax	No. of bits	Mnemonic
PresetConfig() { bsNumPresets; for (i=0; i<bsNumPresets+1; i++) { bsNumBytePresetLabel[i]; for (j=0; j<bsNumBytePresetLabel[i]; j++) { bsPresetLabel[i][j]; } bsPresetMatrix[i]; if (bsPresetMatrix[i]) { PresetMatrixData(); } else { PresetUserData(); } } }	4 8 8 1	uimsbf uimsbf bslbf uimsbf

Table 17 — Syntax of PresetMatrixData()

Syntax	No. of bits	Mnemonic
PresetMatrixData() { bsPresetMatrixType; for(i=0; i<bsNumObjects+1; i++) { numChannels = PresetMatrixType[bsPresetMatrixType]; for(j=0; j<numChannels; i++) { bsPresetMatrixElements[i][j]; } } }	2 5	uimsbf uimsbf

Table 18 — Syntax of PresetUserData()

Syntax	No. of bits	Mnemonic
PresetUserData() { for(i=0; i<6; i++) { bsPresetUserDataIdentifier [i]; } bsPresetUserDataLen ; PresetUserDataContainer(bsPresetUserDataIdentifier); }	8 12	bslbf uimsbf

Table 19 — Syntax of SAOExtensionConfigData(10)

Syntax	No. of bits	Mnemonic
SAOExtensionConfigData(10) { SeparationMetaData(); }		

Table 20 — Syntax of SeparationMetaData()

Syntax	No. of bits	Mnemonic
SeparationMetaData() { bsNumSeparationPairs ; for (i=0; i<bsNumSeparationPairs+1; i++) { bsSeparationMainObjectID [i]; bsSeparationSubObjectID [i]; } }	4 5 5	uimsbf uimsbf uimsbf

Table 21 — Syntax of SAOCFrame()

Syntax	No. of bits	Mnemonic
SAOCFrame() { SAOCFramingInfo(); bsIndependencyFlag ; for(i=0; i<bsNumObjects+1; i++) { idxOLD[i] = EcDataSaoc(OLD, i, numBands); } if (bsTransmitAbsNrg) { idxNRG = EcDataSaoc(NRG, 0, numBands); } k=0; iocIdx1=0; iocIdx2=0; for(i=0; i<bsNumObjects+1; i++) { idxIOC[i][i] = 0; for(j=i+1; j<bsNumObjects+1; j++) { if (bsRelatedTo[i][j] != 0) { if (bsOneIOC == 0) { idxIOC[i][j] = EcDataSaoc(IOC, k, numBands); k++; } } } } }	1	uimsbf

```

        } else {
            if ( k == 0 ) {
                idxIOC[i][j] = EcDataSaoc(IOC, k, numBands);
                k++;
                iocIdx1=i;
                iocIdx2=j;
            } else {
                idxIOC[i][j] = idxIOC[iocIdx1][iocIdx2];
            }
        }
    } else {
        idxIOC[i][j] = 5;
    }
    idxIOC[j][i] = idxIOC[i][j];
}
}
idxDMG = EcDataSaoc(DMG, 0, bsNumObjects+1);
if ( bsNumDmxChannels == 1 ) {
    idxDCLD = EcDataSaoc(DCLD, 0, bsNumObjects+1);
}
if ( bsPdgFlag == 1 ) {
    for (i=0; i<bsNumDmxChannels + 1; i++) {
        idxPDG[i] = EcDataSaoc(DCLD, i, numBands);
    }
}
if ( bsDcuFlag == 1 ) && ( bsDcuDynamic == 1 ) {
    if ( bsIndependencyFlag == 1 ) {
        bsDcuDynamicUpdate = 1;
    } else {
        bsDcuDynamicUpdate;
    }
    if ( bsDcuDynamicUpdate == 1 ) {
        bsDcuMode;
        bsDcuParam;
    }
}
ByteAlign();
SAOCExtensionFrame();
}

```

Note

1

uimsbf

1

uimsbf

4

uimsbf

NOTE numBands is defined in Table 36 and depends on bsFreqRes.

Table 22 — Syntax of SAOCDEFrame()

Syntax	No. of bits	Mnemonic
<pre> SAOCDEFrame() { SAOCDEFramingInfo(); if (bsVersion == 0) { bsIndependencyFlag; for(i=0; i<bsNumObjects+1; i++) { idxOLD[i] = EcDataSaoc(OLD, i, numBands); } k=0; iocIdx1=0; iocIdx2=0; for(i=0; i<bsNumObjects+1; i++) { idxIOC[i][i] = 0; for(j=i+1; j<bsNumObjects+1; j++) { if (bsRelatedTo[i][j] != 0) { if (bsOneIOC == 0) { idxIOC[i][j] = EcDataSaoc(IOC, k, numBands); k++; } else { if (k == 0) idxIOC[i][j] = EcDataSaoc(IOC, k, numBands); k++; iocIdx1=i; iocIdx2=j; } else { idxIOC[i][j] = idxIOC[iocIdx1][iocIdx2]; } } } idxIOC[j][i] = idxIOC[i][j]; } } for (i=0; i<bsNumDmxChannels + 1; i++) { idxDMG[i] = EcDataSaoc(DMG, i, bsNumObjects+1); } if (bsPdgFlag == 1) { for (i=0; i<bsNumDmxChannels + 1; i++) { idxPDG[i] = EcDataSaoc(DCLD, i, numBands); } } if (bsDeLimitFlag == 1) { if (bsIndependencyFlag == 1) { bsDeLimitUpdate = 1; } else { bsDeLimitUpdate; } if (bsDeLimitUpdate == 1) { bsDeLimitFgo; bsDeLimitBgo; } } if (bsDcuFlag == 1) && (bsDcuDynamic == 1) { if (bsIndependencyFlag == 1) { bsDcuDynamicUpdate = 1; } } } </pre>	1 1 4 4	uimsbf Note uimsbf uimsbf

<pre> } else { bsDcuDynamicUpdate; } if (bsDcuDynamicUpdate == 1) { bsDcuMode; bsDcuParam; } } ByteAlign(); SAOExtensionFrame(); } </pre>	1	uimsbf
	1	uimsbf
	4	uimsbf
NOTE numBands is defined in Table 36 and depends on bsFreqRes.		

Table 23 — Syntax of SAOCFramingInfo()

Syntax	No. of bits	Mnemonic
SAOCFramingInfo()		
{		
bsFramingType;	1	uimsbf
If (bsLowDelayMode == 0) {		
bsNumParamSets;	3	uimsbf
} else {		
bsNumParamSets;	1	uimsbf
}		
for (ps=0; ps<numParamSets; ps++) {		
if (bsFramingType) {		Note 1
bsParamSlot[ps];	nBitsParamSlot	uimsbf
		Note 2
} else {		
bsParamSlot[ps] = ceil(numSlots*(ps+1)/numParamSets)-1;		Notes 1 and 3
}		
}		
}		
NOTE 1 numParamSets is defined by numParamSets = bsNumParamSets + 1.		
NOTE 2 nBitsParamSlot is defined according to nBitsParamSlot = ceil[log2(numSlots)].		
NOTE 3 numSlots is defined by numSlots = bsFrameLength + 1.		

Table 24 — EcDataSaoc()

Syntax	No. of bits	Mnemonic
EcDataSaoc(dataType, paramIdx, stopIdx)		Note 1
<pre> { dataSets = 0; for (ps=0; ps<bsNumParamSets + 1; ps++) { bsXXXdataMode[paramIdx][ps]; if (bsXXXdataMode[paramIdx][ps] == 3) { dataSets++; } } setIdx = 0; while (setIdx < dataSets) { If (dataSets-setIdx > 1) { bsDataPairXXX[paramIdx][setIdx]; } else { bsDataPairXXX[paramIdx][setIdx] = 0; } bsQuantCoarseXXX[paramIdx][setIdx]; bsFreqResStrideXXX[paramIdx][setIdx]; dataDim = (stopIdx-1)/pbStride+1; SAOEcDataPair(dataType, paramIdx, setIdx, dataDim, bsDataPairXXX[paramIdx][setIdx], bsQuantCoarseXXX[paramIdx][setIdx]); if (bsDataPairXXX[paramIdx][setIdx]) { bsXXXQuantCoarse[paramIdx][setIdx+1] = bsXXXQuantCoarse[paramIdx][setIdx]; bsFreqResStrideXXX[paramIdx][setIdx+1] = bsFreqResStrideXXX[paramIdx][setIdx]; } setIdx += bsDataPairXXX[paramIdx][setIdx]+1; } stopIdxXXX[paramIdx] = stopIdx; } </pre>	2 1 1 2	uimsbf uimsbf uimsbf uimsbf Note 2
NOTE 1 XXX is to be replaced by the value of dataType (OLD, IOC, NRG, DCLD, DMG).		
NOTE 2 pbStride is defined in ISO/IEC 23003-1:2007, Table 70 and depends on bsFreqResStride[][]. Furthermore, the division shall be interpreted as ANSI C integer division.		

Table 25 — Syntax of SAOCecDataPair()

Syntax	No. of bits	Mnemonic
SAOCecDataPair(dataType, paramIdx, setIdx, dataBands, pairFlag, coarseFlag)		Note 1
<pre> { mixedTimePair_flag = 0; bsPcmCodingXXX[paramIdx][setIdx]; if (bsPcmCoding[paramIdx][setIdx]) { if (coarseFlag) { numQuantSteps = numQuantStepsXXXCoarse; } else { numQuantSteps = numQuantStepsXXXFine; } aaDataPair = GroupedPcmData(dataType, pairFlag, numQuantSteps, dataBands); } else { allowDiffTimeBack = (!bsIndependencyFlag) (setIdx>0); (aaDataPairMsbDiff, aPgOffset, mixedTimePair_flag) = SAOCDiffHuffData(dataType, pairFlag, allowDiffTimeBack, dataBands); aaDataPairLsb[0] = LsbData(dataType, coarseFlag, dataBands); if (pairFlag) { aaDataPairLsb[1] = LsbData(dataType, coarseFlag, dataBands); } } bsDiffTypeXXX[paramIdx][setIdx] = bsDiffType[0]; bsDiffTimeDirectionXXX[paramIdx][setIdx] = bsDiffTimeDirection[0]; mixedTimePairXXX[paramIdx][setIdx] = mixedTimePair_flag; if (pairFlag) { bsDiffTypeXXX[paramIdx][setIdx+1] = bsDiffType[1]; bsDiffTimeDirectionXXX[paramIdx][setIdx+1] = bsDiffTimeDirection[1]; bsPcmCodingXXX[paramIdx][setIdx+1] = bsPcmCodingXXX[paramIdx][setIdx]; mixedTimePairXXX[paramIdx][setIdx+1] = mixedTimePair_flag; } for (pg=0; pg<dataBands; pg++) { if (bsPcmCodingXXX[paramIdx][setIdx]) { bsXXXpcm[paramIdx][setIdx][pg] = aaDataPair[0][pg]; } else { bsXXXmsbDiff[paramIdx][setIdx][pg] = aaDataPairMsbDiff[0][pg]; bsXXXlsb[paramIdx][setIdx][pg] = aaDataPairLsb[0][pg]; } if (pairFlag) { if (bsPcmCodingXXX[paramIdx][setIdx+1]) { bsXXXpcm[paramIdx][setIdx+1][pg] = aaDataPair[1][pg]; } else { bsXXXmsbDiff[paramIdx][setIdx+1][pg] = aaDataPairMsbDiff[1][pg]; bsXXXlsb[paramIdx][setIdx+1][pg] = aaDataPairLsb[1][pg]; } } } } </pre>	1	uimsbf
		Note 2
		Note 2
		Note 3
		Note 4
		Note 4
NOTE 1 XXX is to be replaced by the value of dataType (OLD, IOC, NRG, DCLD, DMG, PDG). NOTE 2 numQuantStepsXXXCoarse and numQuantStepsXXXFine is defined in Table 48 and depends on dataType. NOTE 3 GroupedPcmData() is defined in ISO/IEC 23003-1:2007, Table 25. NOTE 4 LsbData() is defined in ISO/IEC 23003-1:2007, Table 31.		

Table 26 — Syntax of SAOCDiffHuffData()

Syntax	No. of bits	Mnemonic
<pre> SAOCDiffHuffData(dataType, pairFlag, allowDiffTimeBackFlag, dataBands) { mixedTimePair_flag = 0; bsDiffType[0] = DIFF_FREQ; bsDiffType[1] = DIFF_FREQ; if (pairFlag allowDiffTimeBackFlag) { bsDiffType[0]; } if (pairFlag && ((bsDiffType[0] == DIFF_FREQ) allowDiffTimeBackFlag)) { bsDiffType[1]; } bsCodingScheme; if (bsCodingScheme == HUFF_1D) { (aaHuffData[0]) = SAOCHuffData1D(dataType, aDiffType[0],dataBands); if (pairFlag) { (aaHuffData[1]) = SAOCHuffData1D(dataType, aDiffType[1],dataBands); } } else { bsPairing = FREQ_PAIR; (aaHuffData[0]) = SAOCHuffData2DFreqPair(dataType, aDiffType[0], dataBands); If (pairFlag) { (aaHuffData[1]) = SAOCHuffData2DFreqPair(dataType, aDiffType[1], dataBands); } } if ((bsDiffType[0] == DIFF_TIME) (bsDiffType[1] == DIFF_TIME)) { bsDiffTimeDirection[0] = BACKWARDS; if (pairFlag) { bsDiffTimeDirection[1] = BACKWARDS; } } return (aaHuffData, aPgOffset, mixedTimePair_flag); } </pre>	1 1 1	uimsbf uimsbf uimsbf

Table 27 — Syntax of SAOCHuffData1D()

Syntax	No. of bits	Mnemonic
<pre> SAOCHuffData1D(dataType, diffType, dataBands) { pgOffset = 0; if (diffType == DIFF_FREQ) { aHuffData1D[0] = 1Dhuff_dec(hcodFirstBand_XXX, bsCodeW); pgOffset = 1; } for (i=pgOffset; i<dataBands; i++) { aHuffData1D[i] = 1Dhuff_dec(hcod1D_XXX_YY, bsCodeW); if (aHuffData1D[i] != 0) { bsSign; if (bsSign) { aHuffData1D[i] = -aHuffData1D[i]; } } } return (aHuffData1D); } </pre>	1..x 1..x 1	vlclbf Notes 1 and 3 vlclbf Notes 1, 2 and 3 uimsbf
NOTE 1 XXX is to be replaced by the value of dataType (OLD, IOC, NRG, DCLD, DMG, PDG). NOTE 2 YY is to be replaced by “DF” or “DT”, depending on the value of diffType. NOTE 3 1Dhuff_dec() is defined in ISO/IEC 23003-1:2007, Annex A.1.		

Table 28 — Syntax of SAOCHuffData2DFreqPair()

Syntax	No. of bits	Mnemonic
<pre> SAOCHuffData2DFreqPair(dataType, diffType, dataBands) { LavIdx = 1Dhuff_dec(hcodLavIdx, bsCodeW); lav = lavTabXXX[LavIdx]; pgOffset = 0; if (diffType == DIFF_FREQ) { aHuffData2D[0] = 1Dhuff_dec(hcodFirstBand_XXX, bsCodeW); pgOffset = 1; } escapeCode = hcod2D_XXX_YY_FP_LL_escape; escCntr = 0; for (i=pgOffset; i<dataBands; i+=2) { (aTmp[0], aTmp[1]) = 2Dhuff_dec(hcod2D_XXX_YY_FP_LL, bsCodeW); if (bsCodeWord != escapeCode) { aTmpSym = SymmetryData(aTmp); aHuffData2D[i] = aTmpSym[0]; aHuffData2D[i+1] = aTmpSym[1]; } else { aEscList[escCntr++] = i; } } } </pre>	1..3 1..x 1..x	vlclbf NOTE 1 vlclbf Note 6 Notes 2, 3, 4 and 5 vlclbf Notes 3, 4, 5 and 6

<pre> } if (escCntr > 0) { aaEscData = GroupedPcmData(dataType, 1, 2*lav+1, escCntr); for (i=0; i<escCntr; i++) { aHuffData2D[aEscList[i]] = aaEscData[0][i] - lav; aHuffData2D[aEscList[i]+1] = aaEscData[1][i] - lav; } } if ((dataBands-pgOffset) % 2) { aHuffData2D[dataBands-1] = 1Dhuff_dec(hcod1D_XXX_YY, bsCodeW); if (aHuffData2D[dataBands-1] != 0) { bsSign; if (bsSign) { aHuffData2D[dataBands-1] = -aHuffData2D[dataBands-1]; } } } return (aHuffData2D); } </pre>			1..x	Note 7 vlclbf Notes 3, 4 and 6
			1	uimsbf
NOTE 1 lavTabXXX is defined in Table 46 and Table 47. NOTE 2 The escape code tables are defined in Table A.6, Table A.7 and Table A.8. NOTE 3 XXX is to be replaced by the value of dataType (OLD, IOC, NRG, DCLD, DMG, PDG). NOTE 4 YY is to be replaced by "DF" or "DT", depending on the value of diffType. NOTE 5 LL is to be replaced by the value of lav. NOTE 6 1Dhuff_dec() and 2Dhuff_dec() are defined in ISO/IEC 23003-1:2007, Annex A.1. NOTE 7 % denotes the modulo operator (ANSI C integer math) and returns the remainder of the division.				

Table 29 — Syntax of SAOCExtensionFrame()

Syntax	No. of bits	Mnemonic
<pre> SAOCExtensionFrame() { for (ec=0; ec<SaocExtNum; ec++) { if (SaocExtType[ec]<8) { cnt = bsSaocExtLen; if (cnt==255) { cnt += bsSaocExtLenAdd; } if (cnt>0) { bitsRead = SAOCExtensionFrameData(SaocExtType[ec]) } nFillBits = 8*cnt-bitsRead; bsFillBits; } } } </pre>		
	8	uimsbf
	16	uimsbf
		Note 1
	nFillBits	bslbf
NOTE 1 SAOCExtensionFrameData() returns the number of bits read.		

Table 30 — Syntax of SAOCExtensionFrameData(0)

Syntax	No. of bits	Mnemonic
SAOCExtensionFrameData(0)		
{		
if (bsDeLimitFlag == 1) {		
if (bsIndependencyFlag == 1) {		
bsDeLimitEaoUpdate = 1;		
} else {		
bsDeLimitEaoUpdate;	1	uimsbf
}		
if (bsDeLimitEaoUpdate == 1) {		
bsDeLimitFgoEAO;	4	uimsbf
bsDeLimitBgoEAO;	4	uimsbf
}		
}		
if ((bsDcuFlag == 1) && (bsDcuFlag2 == 1) && (bsDcuDynamic == 1)) {		
if (bsIndependencyFlag == 1) {		
bsDcuDynamicUpdate2 = 1;		
} else {		
bsDcuDynamicUpdate2;	1	uimsbf
}		
if (bsDcuDynamicUpdate2 == 1) {		
bsDcuMode2;	1	uimsbf
bsDcuParam2;	4	uimsbf
}		
}		
SAOCResidualData();		
}		

Table 31 — Syntax of SAOCResidualData()

Syntax	No. of bits	Mnemonic
<pre>SAOCResidualData() { for (i=0; i<bsNumEAO + 1; i++) { if (bsResidualPresent[i]) { individual_channel_stream(0); } } }</pre>		Note 1
NOTE 1 individual_channel_stream(0) according to MPEG-2 AAC Low Complexity profile bitstream syntax described in ISO/IEC 13818-7:2006, 6.3.		

Table 32 — Syntax of SAOCExtensionFrameData(1)

Syntax	No. of bits	Mnemonic
SAOExtensionFrameData(1) { PresetConfig(); }		

Table 33 — Syntax of SAOCExtensionFrameData(2)

Syntax	No. of bits	Mnemonic
SAOCExtensionFrameData(2) { SpatialFrame(); }		Note 1
NOTE 1 SpatialFrame() is defined in ISO/IEC 23003-1:2007, Table 15.		

Table 34 — Syntax of SAOCExtensionFrameData(3)

Syntax	No. of bits	Mnemonic
SAOCExtensionFrameData(3) { ObjectMetaData(); }		

6.2 Definition

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

SAOCSpecificConfig() Syntactic element that contains the SAOC configuration data (header).

SAOCDESpecificConfig() Syntactic element that contains the SAOC configuration data (header) in case of SAOC-DE profile.

bsVersion Defines the version of the bitstream according to Table 35.

Table 35 — bsVersion

bsVersion	Meaning
0	SAOC-DE profile, levels 1 and 2
1 ... 15	Reserved

bsSamplingFrequencyIndex

See ISO/IEC 14496-3:2009, 1.6.3.4.

bsSamplingFrequency See ISO/IEC 14496-3:2009, 1.6.3.3.**bsFreqRes**

Defines the number of parameter bands according to Table 36.

Table 36 — bsFreqRes

bsFreqRes	numBands	
	bsLowDelayMode==0	bsLowDelayMode==1
0	N/A	N/A
1	28	23
2	20	15
3	14	12
4	10	9
5	7	7
6	5	5
7	4	4

bsLowDelayMode

Indicates whether the SAOC operates in HQ or LD mode according to Table 37.

Table 37 — bsLowDelayMode

bsLowDelayMode	Meaning
0	HQ operation mode
1	LD operation mode (see 7.14)

In case of SAOC-DE profile, **bsLowDelayMode** shall be set to 0.**bsTttBandsHigh**Same as **bsTttBandsLow** but for high band range. The high band range is **bsTttBandsLow** ≤ pb < **bsTttBandsHigh**.**bsFrameLength**

Defines the number of time slots in an SAOC frame.

bsNumObjects

Defines the number of object channels for which SAOC parameters are transmitted.

In case of stereo or multi-channel objects, each channel counts as separate object.

bsNumFGOs

Defines the number of FGOs according to Table 38.

Table 38 — bsNumFGOs

bsNumFGOs	Meaning
0	$N_{\text{FGO}} = 1$
1	$N_{\text{FGO}} = 2$
2	$N_{\text{FGO}} = 3$
3,...,7	N/A

bsRelatedTo[i][j]Defines whether an object has relation with other objects. More precisely, **bsRelatedTo[i][j]** == 1 means that object *i* and object *j* are related, i.e. have correlation, whereas **bsRelatedTo[i][j]** == 0 means that this is not the case.

bsTransmitAbsNrg Defines whether absolute energy parameters are transmitted. These are necessary for delay-free merging of SAOC bitstreams in an MCU bitstream combiner.

bsNumDmxChannels Defines the number of downmix channels according to Table 39.

Table 39 — bsNumDmxChannels

bsNumDmxChannels	Meaning
0	mono downmix
1	stereo downmix
2	3-channel downmix
3,...,7	N/A

bsTttDualMode Indicates if transcoding to the TTT box operates in different modes for a low and high band range according to Table 40.

Table 40 — bsTttDualMode

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

bsTttBandsLow Defines the number of parameter bands for which TTT transcoding should be processed according to prediction or energy based scheme. Prediction based scheme should be used for the parameter band range: $0 \leq pb < \text{bsTttBandsLow}$. Energy based scheme should be used for the parameter band range: $\text{bsTttBandsLow} \leq pb < \text{numBands}$.

bsTtnDualMode Indicates if transcoding to the OTN/TTN box operates in different modes for a low and high band range according to Table 41.

Table 41 — bsTtnDualMode[i]

bsTtnDualMode[i]	Meaning
0	same OTN/TTN transcoding mode for full band range (i.e. no separate high band range)
1	different OTN/TTN transcoding modes for low and high band ranges

bsTtnBandsLow Defines the number of parameter bands for which OTN/TTN transcoding should be processed according to prediction or energy based scheme. Prediction based scheme should be used for the parameter band range: $0 \leq pb < \text{bsTtnBandsLow}$. Energy based scheme should be used for the parameter band range: $\text{bsTtnBandsLow} \leq pb < \text{numBands}$.

bsPdgFlag Defines whether PDG parameters are transmitted according to Table 42.

Table 42 — bsPdgFlag

bsPdgFlag	Meaning
0	PDG parameters are not transmitted
1	PDG parameters are transmitted

bsDcuFlag	Defines whether the values bsDcuMode and bsDcuParam are transmitted in the bitstream.
bsDcuFlag2	Defines whether the values bsDcuMode2 and bsDcuParam2 are transmitted in the bitstream.
bsDcuMandatory	If bsDcuMandatory == 1, then the DCU shall be applied using the parameters bsDcuMode and bsDcuParam as transmitted in the bitstream. If bsDcuMandatory == 0, then the DCU parameters bsDcuMode and bsDcuParam transmitted in the bitstream are only recommended values and also other DCU settings could be used.
bsDcuDynamic	Enables dynamic signaling of the values bsDcuMode and bsDcuParam .
bsDcuDynamicUpdate	Defines whether the values bsDcuMode and bsDcuParam are updated. More precisely, bsDcuDynamicUpdate == 1 means that the values bsDcuMode and bsDcuParam are updated in the current frame, whereas bsDcuDynamicUpdate == 0 means that the previously transmitted values are kept.
bsDcuDynamicUpdate2	Same as bsDcuDynamicUpdate but for application only in strict EAO mode.
bsDcuMode	Defines the distortion-free target matrix type for the DCU according to Table 43.

Table 43 — bsDcuMode

bsDcuMode	Meaning
0	Downmix-similar target matrix
1	Best-effort target matrix

bsDcuMode2	Same as bsDcuMode but for application only in strict EAO mode.
bsDcuParam	Defines the parameter value for the DCU algorithm according to Table 44.

Table 44 — bsDcuParam parameters quantization table

idx	0	1	2	3	4	5	6	7
DcuParam[idx]	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35
idx	8	9	10	11	12	13	14	15
DcuParam[idx]	0.40	0.45	0.50	0.60	0.70	0.80	0.90	1.00

bsDcuParam2	Same as bsDcuParam , but for application only in strict EAO mode.
bsOneIOC	Indicates if only a single IOC parameter is conveyed common to all objects which have relation with other, signalled by bsRelatedTo [i][j] == 1.
bsDeLimitFlag	Defines whether the values bsDeLimitFgo , bsDeLimitFgoEAO , bsDeLimitBgo , and bsDeLimitBgoEAO are transmitted in the bitstream. In case of baseline or LD profiles, bsDeLimitFlag shall be set to 0.
bsDeLimitFgo	Defines the value representing the lowest acceptable modification boundary related to the FGO for the modification range control algorithm according to Table 45.

Table 45 — bsDeLimitFgo, bsDeLimitFgoEAO, bsDeLimitBgo and bsDeLimitBgoEAO parameters quantization table

idx	0	1	2	3	4	5	6	7
DeLimit[idx]	$10^{-7.50}$	$10^{-2.25}$	$10^{-2.00}$	$10^{-1.75}$	$10^{-1.50}$	$10^{-1.25}$	$10^{-1.10}$	$10^{-0.95}$
idx	8	9	10	11	12	13	14	15
DeLimit[idx]	$10^{-0.80}$	$10^{-0.65}$	$10^{-0.50}$	$10^{-0.40}$	$10^{-0.30}$	$10^{-0.20}$	$10^{-0.10}$	1

bsDeLimitBgo	Defines the value representing the lowest acceptable modification boundary related to the BGO for the modification range control algorithm according to Table 45.
bsDeLimitFgoEAO	Same as bsDeLimitFgo but for application only in strict EAO mode.
bsDeLimitBgoEAO	Same as bsDeLimitBgo but for application only in strict EAO mode.
bsObjectMetaDataAvailable	Defines whether or not metadata information for the encoded objects is transmitted.
ObjectMetaData()	Syntactic element that contains a metadata description of the the encoded objects.
bsNumEAO	Defines the number of EAO channels.
bsResidualFramesPerSAOCFrame	Indicates the number of residual frames per SAOC frame, ranging from one to four according to ISO/IEC 23003-1:2007, Table 56.
bsNumByteMetaData[i]	Defines the number of bytes used for the metadata description of an object, <i>i</i> .
bsMetaData[i][j]	Defines the metadata description (interpreted as a string in UTF-8 encoding format).
SAOCFrame()	Syntactic element that contains the data of an SAOC frame (payload).
SAOCDEFrame()	Syntactic element that contains the data of an SAOC frame (payload) in case of SAOC-DE profile.
SAOCFramingInfo()	Syntactic element that contains information about the number of parameter sets and their associated time slots.
bsIndependencyFlag	Indicates if lossless coding of the current SAOC frame is done independently of the previous SAOC frame, i.e. whether the current SAOC frame can be decoded without knowledge about the previous SAOC frame.
bsDeLimitUpdate	Defines whether the values bsDeLimitFgo , bsDeLimitFgoEAO , bsDeLimitBgo and bsDeLimitBgoEAO are updated. More precisely, bsDeLimitUpdate == 1 means that the values bsDeLimitFgo , bsDeLimitFgoEAO , bsDeLimitBgo , and bsDeLimitBgoEAO are updated in the current frame, whereas bsDeLimitUpdate == 0 means that the previously transmitted values are kept.
bsDeLimitEaoUpdate	Defines whether the values bsDeLimitFgoEAO and bsDeLimitBgoEAO are updated. More precisely, bsDeLimitEaoUpdate == 1 means that the values bsDeLimitFgoEAO and bsDeLimitBgoEAO are updated in the current frame,

whereas **bsDeLimitEaoUpdate** == 0 means that the previously transmitted values are kept.

EcDataSaoc() Syntactic element that contains all parameter subsets of a given parameter in the SAOC frame. However, the changes described in Table 46 and Table 47 apply.

Table 46 — lavTabXXXdf

LavIdx	lavTabDCLD/DMG/PDGdf [LavIdx]	lavTabIOCdf [LavIdx]	lavTabOLDdf [LavIdx]	lavTabNRGdf [LavIdx]
0	3	1	3	3
1	5	3	6	5
2	7	5	9	7
3	9	7	12	9
NOTE This table replaces ISO/IEC 23003-1:2007, Table 76 and shall be applied only in case diffType == DIFF_FREQ.				

Table 47 — lavTabXXXdt

LavIdx	lavTabDCLD/DMG/PDGdt [LavIdx]	lavTabIOCdt [LavIdx]	lavTabOLDdt [LavIdx]	lavTabNRGdt [LavIdx]
0	3	1	3	3
1	5	3	6	6
2	7	5	9	9
3	9	7	12	12
NOTE This table replaces ISO/IEC 23003-1:2007, Table 76 and shall be applied only in case diffType == DIFF_TIME.				

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

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

bsPcmCodingXXX Indicates whether PCM coding is applied.

GroupedPcmData() Syntactic element that contains one or two temporally subsequent parameter subsets of a given parameter in the SAOC frame, where groups of quantized values are represented by a single PCM code. numQuantSteps depends on the data type XXX and whether coarse or fine quantization is used and is defined in Table 48.

Table 48 — numQuantSteps

XXX (dataType)	numQuantStepsXXXCoarse	numQuantStepsXXXFine
DCLD, DMG, PDG	15	31
IOC	4	8
OLD	8	16
NRG	32	64

SAOCExtensionConfig() Syntactic element that acts as container to carry extensions to the SAOC audio configuration. The presence of the SAOC extension of type **bsSaocExtType** (see Table 49) is signaled by the presence of a **SAOCExtensionConfigData(bsSaocExtType)** element in **SAOCExtensionConfig()**.

SaocExtNum Helper variable storing the number of the SAOC extension containers present.

bsSaocExtType Indicates type of the SAOC extension data according to Table 49.

Table 49 — bsSaocExtType

bsSaocExtType	Meaning	SAOCExtensionFrameData()
0	Residual coding data	present
1	Dynamic preset information	
2	MBO MPS data	
3	Dynamic object metadata	
4...7	N/A	
8	Static object metadata	not present
9	Static preset information	
10	Separation metadata	
11...15	N/A	

SaocExtType[i] Helper variable storing the type of spatial extension data carried in the extension container *i*.

bsSaocExtLen Number of bytes in SAOCExtensionConfigData() or SAOCExtensionFrameData().

bsSaocExtLenAdd Additional number of bytes in SAOCExtensionConfigData() or SAOCExtensionFrameData().

bsSaocExtLenAddAdd Further additional number of bytes in SAOCExtensionConfigData().

bsFillBits Fill bits, to be ignored.

SAOCExtensionConfigData(bsSaocExtType)

Instance of the SAOCExtensionConfigData that carries configuration data for the SAOC extension of type **bsSaocExtType** (see Table 49).

SAOCExtensionConfigData(0)

Syntactic element that, if present, indicates that modification range control information (in case of SAOC-DE profile), residual coding information, and corresponding DCU parameters are available.

SAOCExtensionConfigData(1)

Syntactic element that, if present, indicates that dynamic preset information is available.

SAOCExtensionConfigData(2)

Syntactic element that, if present, indicates that the MBO MPS data is available.

SAOCExtensionConfigData(3)

Syntactic element that, if present, indicates that dynamic object metadata is available.

SAOCExtensionConfigData(8)

Syntactic element that, if present, indicates that metadata information is available.

SAOCExtensionConfigData(9)

Syntactic element that, if present, indicates that static preset information is available.

SAOCExtensionConfigData(10)

Syntactic element that, if present, indicates that separation metadata is available.

SeparationMetaData()

Syntactic element that contains a metadata description of the separated objects.

bsNumSeparationPairs

Defines the number of pairs of the separated objects.

bsSeparationMainObjectID[i]

Defines the object ID of the main object which is separated from mixed signals.
Its value is in the range of 0 to **bsNumObjects**.

bsSeparationSubObjectID[i]

Defines the object ID of the sub object which is separated from mixed signals.
Its value is in the range of 0 to **bsNumObjects**.

PresetConfig()

Syntactic element that contains all preset rendering data.

bsNumPresets

Defines the number of available rendering presets.

bsNumBytePresetLabel[i]

Defines the number of bytes used for the text label of the preset *i*.

bsPresetLabel[i][j]

Defines the text label for the preset *i* (interpreted as a string in UTF-8 encoding format).

bsPresetMatrix

Defines the preset data representation type according to Table 50.

Table 50 — bsPresetMatrix

bsPresetMatrix	Meaning
0	User-defined preset representation format
1	Matrix-based preset representation format

PresetMatrixData()

Syntactic element that contains the preset rendering parameters in the matrix-based representation format.

bsPresetMatrixType

Defines the rendering configuration according to Table 51.

Table 51 — bsPresetMatrixType

bsPresetMatrixType	Meaning
0	Mono playback system
1	Stereo playback system
2	5.1 playback system
3	3.0 playback system

PresetMatrixType[**bsPresetMatrixType**]

Defines the number of upmix channels according to Table 52.

Table 52 — PresetMatrixType

bsPresetMatrixType	0	1	2	3
PresetMatrixType[bsPresetMatrixType]	1	2	5	N/A

bsPresetMatrixElements[i][j]

Defines the preset rendering matrix. Namely, the output level of the audio object channel i for the upmix channel j .

PresetUserData()

Syntactic element that contains the preset rendering parameters in the user-defined preset representation format.

bsPresetUserDataIdentifier[i]

Defines the identifier for the user-defined preset i representation format (interpreted as a string in UTF-8 encoding format).

bsPresetUserDataLen Defines the length in bytes for the preset data included into **PresetUserDataContainer()** container element.

PresetUserDataContainer()

Syntactic element that contains preset rendering data in the user-defined preset representation format and has a length of exactly **bsPresetUserDataLen** bytes.

All bitstream variables which are not explicitly described here are defined in ISO/IEC 23003-1. Any modifications and amendments to these definitions are specified in Annex A.

7 SAOC processing

7.1 Compressed data stream decoding and dequantization of SAOC data

7.1.1 General

This subclause describes the decoding and dequantization of the bitstream payload into variables that are used in the SAOC transcoder/decoder.

7.1.2 Dequantization of the SAOC parameters

The dequantization of the IOC parameters follows the same rules as defined in ISO/IEC 23003-1:2007, 6.1.8 for the MPS ICC parameters, the decoding of the DMG, DCLD and PDG parameters those of the MPS CLD parameters.

The dequantization of the OLD (i.e. relative energy) parameters is done by applying the dequantization function $deq(index, parameterType)$ as defined in ISO/IEC 23003-1:2007, 6.1.8 in combination with the parameter quantization table given in Table 53.

Table 53— OLD parameter quantization table

idx	0	1	2	3	4	5	6	7
OLD[idx]	$10^{-15.0}$	$10^{-4.5}$	$10^{-4.0}$	$10^{-3.5}$	$10^{-3.0}$	$10^{-2.5}$	$10^{-2.2}$	$10^{-1.9}$
idx	8	9	10	11	12	13	14	15
OLD[idx]	$10^{-1.6}$	$10^{-1.3}$	$10^{-1.0}$	$10^{-0.8}$	$10^{-0.6}$	$10^{-0.4}$	$10^{-0.2}$	1

The dequantization of the NRG (i.e. absolute energy) parameters is done by applying the dequantization function $deq(index, parameterType)$ as defined in ISO/IEC 23003-1:2007, 6.1.8 in combination with the parameter quantization table given in Table 54.

Table 54 — NRG parameter quantization table

idx	0	1	2	3	4	5	6	7
NRG[idx]	50·10 ^{-9.45}	50·10 ^{-9.30}	50·10 ^{-9.15}	50·10 ^{-9.00}	50·10 ^{-8.85}	50·10 ^{-8.70}	50·10 ^{-8.55}	50·10 ^{-8.40}
idx	8	9	10	11	12	13	14	15
NRG[idx]	50·10 ^{-8.25}	50·10 ^{-8.10}	50·10 ^{-7.95}	50·10 ^{-7.80}	50·10 ^{-7.65}	50·10 ^{-7.50}	50·10 ^{-7.35}	50·10 ^{-7.20}
idx	16	17	18	19	20	21	22	23
NRG[idx]	50·10 ^{-7.05}	50·10 ^{-6.90}	50·10 ^{-6.75}	50·10 ^{-6.60}	50·10 ^{-6.45}	50·10 ^{-6.30}	50·10 ^{-6.15}	50·10 ^{-6.00}
idx	24	25	26	27	28	29	30	31
NRG[idx]	50·10 ^{-5.85}	50·10 ^{-5.70}	50·10 ^{-5.55}	50·10 ^{-5.40}	50·10 ^{-5.25}	50·10 ^{-5.10}	50·10 ^{-4.95}	50·10 ^{-4.80}
idx	32	33	34	35	36	37	38	39
NRG[idx]	50·10 ^{-4.65}	50·10 ^{-4.50}	50·10 ^{-4.35}	50·10 ^{-4.20}	50·10 ^{-4.05}	50·10 ^{-3.90}	50·10 ^{-3.75}	50·10 ^{-3.60}
idx	40	41	42	43	44	45	46	47
NRG[idx]	50·10 ^{-3.45}	50·10 ^{-3.30}	50·10 ^{-3.15}	50·10 ^{-3.00}	50·10 ^{-2.85}	50·10 ^{-2.70}	50·10 ^{-2.55}	50·10 ^{-2.40}
idx	48	49	50	51	52	53	54	55
NRG[idx]	50·10 ^{-2.25}	50·10 ^{-2.10}	50·10 ^{-1.95}	50·10 ^{-1.80}	50·10 ^{-1.65}	50·10 ^{-1.50}	50·10 ^{-1.35}	50·10 ^{-1.20}
idx	56	57	58	59	60	61	62	63
NRG[idx]	50·10 ^{-1.05}	50·10 ^{-0.90}	50·10 ^{-0.75}	50·10 ^{-0.60}	50·10 ^{-0.45}	50·10 ^{-0.30}	50·10 ^{-0.15}	50

The decoding of the SAOCFrame() data or SAOCDEFrame() data results in the parameter indices idxXXX of the quantized OLD, IOC, NRG, DCLD, DMG and PDG parameters that are listed in Table 55, where

$$pi = 0 \dots N - 1, \quad ps = 0 \dots L - 1, \quad pb = 0 \dots M_{proc} - 1, \quad pd = 0 \dots M - 1.$$

Table 55 — Dimensions and value ranges of the parameter indices idxXXX

Parameter	idxOLD	idxNRG	idxIOC	idxDMG	idxDCLD	idxPDG
Dimension	[pi][ps][pb]	[ps][pb]	[pi][pi][ps][pb]	[ps][pi]	[ps][pi]	[ps][pb][pd]
Value range	0 ... 15	0 ... 63	0 ... 7	-15 ... 15	-15 ... 15	-15 ... 15

The decoding follows largely the procedure described in ISO/IEC 23003-1, taking into account the following differences. The pseudo code defining the preprocessing step in ISO/IEC 23003-1:2007, 6.1.2.3.2 is altered in the following way:

```

setIdxStart = dataSetIdx[ps];
startBand = startBandXXX[pi];
stopBand = stopBandXXX[pi];
pbStride = pbStrideTable[bsFreqResStrideXXX[pi][setIdx]];
dataBands = (stopBand - startBand - 1)/pbStride + 1; /*ANSI C integer
math*/
aGroupToBand = createMapping(startBand, stopBand, pbStride);
for (pg=0; pg<dataBands; pg++) {
    pb = aGroupToBand[pg];
    tmp = idxXXX[pi][ps-1][pb];
    switch (XXX) {
    case DCLD, DMG:
        if (bsQuantCoarseXXX[pi][setIdx]) {
            tmp = (tmp/2)+7; /* ANSI C integer math */
        }
        else {
            tmp = tmp+15;
        }
        break;
    case IOC:
        if (bsQuantCoarseXXX[pi][setIdx]) {
            tmp = tmp/2; /* ANSI C integer math */
        }
    }
}

```