BS ISO/IEC 23003-3:2012

BSI Standards Publication

Information technology — MPEG audio technologies

Part 3: Unified speech and audio coding

bsi.

...making excellence a habit.™

Document Sharing Hub

Full document details: BS ISO/IEC 23003-3:2012

National foreword

This British Standard is the UK implementation of ISO/IEC 23003-3:2012, incorporating corrigenda September 2012, September 2013 and December 2014.

Due to the technical nature of this standard these three corrigenda have been appended at the end of this document.

The UK participation in its preparation was entrusted to Technical Committee IST/37, Coding of picture, audio, multimedia and hypermedia information.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2015.
Published by BSI Standards Limited 2015

ISBN 978 0 580 73349 9

ICS 35.040

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 May 2015.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23003-3:2012

INTERNATIONAL STANDARD

ISO/IEC 23003-3

First edition
2012-04-01

Information technology — MPEG audio technologies —

Part 3:
Unified speech and audio coding

Technologies de l'information — Technologies audio MPEG —

Partie 3: Discours unifié et codage audio

Reference number
ISO/IEC 23003-3:2012(E)

Documents
Document Sharing Hub
© ISO/IEC 2012

Full document details: BS ISO/IEC 23003-3:2012

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Documents
Document Sharing Hub
© ISO/IEC 2012 – All rights reserved

Full document details: BS ISO/IEC 23003-3:2012

Contents

Page

Foreword iv

Introduction v

1 Scope 1

2 Normative references 1

3 Terms, definitions, symbols and abbreviated terms 1
3.1 Terms and definitions 1
3.2 Symbols and abbreviated terms 2

4 Technical Overview 2
4.1 Decoder block diagram 2
4.2 Overview of the decoder tools 5
4.3 Combination of USAC with MPEG Surround and SAOC 8
4.4 Interface between USAC and Systems 9
4.5 USAC Profiles and Levels 9

5 Syntax 12
5.1 General 12
5.2 Decoder configuration (UsacConfig) 12
5.3 USAC bitstream payloads 17

6 Data Structure 50
6.1 USAC configuration 50
6.2 USAC payload 63

7 Tool Descriptions 81
7.1 Quantization 81
7.2 Noise Filling 82
7.3 Scalefactors 84
7.4 Spectral Noiseless coding 84
7.5 enhanced SBR Tool (eSBR) 90
7.6 Inter-subband-sample Temporal Envelope Shaping (inter-TES) 139
7.7 Joint Stereo Coding 142
7.8 TNS 149
7.9 Filterbank and block switching 151
7.10 Time-Warped Filterbank and Blockswitching 159
7.11 MPEG Surround for Mono to Stereo upmixing 167
7.12 AVQ decoding 180
7.13 LPC-filter 186
7.14 ACELP 193
7.15 MDCT based TCX 202
7.16 Forward Aliasing Cancellation (FAC) tool 206
7.17 Post-processing of the synthesis signal 208

Annex A (normative) **Tables** 211

Annex B (informative) **Encoder Tools** 216

Annex C (normative) **Tables for Arithmetic Decoder** 254

Annex D (normative) **Tables for Predictive Vector Coding** 260

Annex E (informative) **Adaptive Time / Frequency Post-Processing** 269

Annex F (informative) **Audio/Systems Interaction** 275

Annex G (informative) **Patent Statements** 277

Bibliography 278

Documents
Document Sharing Hub

© ISO/IEC 2012 – All rights reserved

Full document details: BS ISO/IEC 23003-3:2012

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 23003-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

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

— *Part 1: MPEG Surround*

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

— *Part 3: Unified speech and audio coding*

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Introduction

As mobile appliances become multi-functional, multiple devices converge into a single device. Typically, a wide variety of multimedia content is required to be played on or streamed to these mobile devices, including audio data that consists of a mix of speech and music.

This part of ISO/IEC 23003 Unified Speech and Audio Coding (USAC) is a new audio coding standard that allows for coding of speech, audio or any mixture of speech and audio with a consistent audio quality for all sound material over a wide range of bitrates. It supports single and multi-channel coding at high bitrates and provides perceptually transparent quality. At the same time, it enables very efficient coding at very low bitrates while retaining the full audio bandwidth.

Where previous audio codecs had specific strengths in coding either speech or audio content, USAC is able to encode all content equally well, regardless of the content type.

In order to achieve equally good quality for coding audio and speech, the developers of USAC employed the proven MDCT-based transform coding techniques known from MPEG-4 audio and combined them with specialized speech coder elements like ACELP. Parametric coding tools such as MPEG-4 spectral band replication (SBR) and MPEG-D MPEG surround were enhanced and tightly integrated into the codec. The result delivers highly efficient coding and operates down to the lowest bit rates.

The main focus of this codec are applications in the field of typical broadcast scenarios, multi-media download to mobile devices, user-generated content such as podcasts, digital radio, mobile TV, audio books, etc.

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 patents.

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

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

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

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23003-3:2012

Information technology — MPEG audio technologies —

Part 3:
Unified speech and audio coding

1 Scope

This part of ISO/IEC 23003 specifies a unfied speech and audio codec which is capable of coding signals having an arbitrary mix of speech and audio content. The codec has a performance comparable to or better than the best known coding technology that might be tailored specifically to coding of either speech or general audio content. The codec supports single and multi-channel coding at high bitrates and provides perceptually transparent quality. At the same time, it enables very efficient coding at very low bitrates while retaining the full audio bandwidth.

This part of ISO/IEC 23003 incorporates several perceptually-based compression techniques developed in previous MPEG standards: perceptually shaped quantization noise, parametric coding of the upper spectrum region and parametric coding of the stereo sound stage. However, it combines these well-known perceptual techniques with a source coding technique: a model of sound production, specifically that of human speech.

2 Normative references

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

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

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

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14496-3, ISO/IEC 23003-1 and the following apply.

3.1.1
algebraic codebook
fixed codebook where an algebraic code is used to populate the excitation vectors (innovation vectors)

NOTE The excitation contains a small number of nonzero pulses with predefined interlaced sets of potential positions. The amplitudes and positions of the pulses of the kth excitation codevector can be derived from its index k through a rule requiring no or minimal physical storage, in contrast with stochastic codebooks whereby the path from the index to the associated codevector involves look-up tables.

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23003-3:2012

3.1.2
AVQ
Algebraic Vector Quantizer
process associating, to an input block of 8 coefficients, the nearest neighbour from an 8-dimensional lattice and a set of binary indices to represent the selected lattice point

NOTE The above definition describes the encoder. At the decoder, AVQ describes the process to obtain, from the received set of binary indices, the 8-dimensional lattice point that was selected at the encoder.

3.1.3
closed-loop pitch
result of the adaptive codebook search, a process of estimating the pitch (lag) value from the weighted input speech and the long-term filter state

NOTE In the closed-loop search, the lag is searched using error minimization loop (analysis-by-synthesis). In USAC, closed-loop pitch search is performed for every subframe.

3.1.4
fractional pitch
set of pitch lag values having sub-sample resolution

NOTE In the LPD USAC, a sub-sample resolution of $1/4^{th}$ or $1/2^{nd}$ of a sample is used.

3.1.5
ZIR
zero input response
output of a filter due to past inputs, i.e. due to the present state of the filter, given that an input of zeros is applied

3.2 Symbols and abbreviated terms

For the purposes of this document, the symbols and abbreviated terms given in ISO/IEC 14496-3 and the following apply.

ACELP Algebraic Code-Excited Linear Predictor

PVC Predictive Vector Coding

uclbf unary code, left bit first

NOTE "left bit first" refers to the order in which the unary codes are received. The value is encoded using a conventional unary code, where any decimal value d is represented by d '1' bits followed by one '0' stop-bit.

USAC Unified Speech and Audio Coding

4 Technical Overview

4.1 Decoder block diagram

The block diagram of the USAC decoder as shown in Figure 1 reflects the general structure of MPEG-D USAC which can be described as follows (from bottom to top): There is a common pre/postprocessing stage consisting of an MPEG Surround functional unit to handle stereo processing (MPS212) and an enhanced SBR (eSBR) unit which handles the parametric representation of the higher audio frequencies in the input signal. Then there are two branches, one consisting of a modified Advanced Audio Coding (AAC) tool path (frequency domain, "FD") and the other consisting of a linear prediction coding (LP or LPC domain, "LPD") based path. The latter can use either a frequency domain representation or a time domain representation of the LPC residual. All transmitted spectra for both FD and LPD path are represented in MDCT domain. The quantized spectral coefficients are coded using a context adaptive arithmetic coder. The time domain representation uses an ACELP excitation coding scheme.

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

In case of transmitted spectral information the decoder shall reconstruct the quantized spectra, process the reconstructed spectra through whatever tools are active in the bitstream payload in order to arrive at the actual signal spectra as described by the input bitstream payload, and finally convert the frequency domain spectra to the time domain. Following the initial reconstruction and scaling of the spectrum, there are optional tools that modify one or more of the spectra in order to provide more efficient coding.

In case of transmitted time domain signal representation, the decoder shall reconstruct the quantized time signal, process the reconstructed time signal through whatever tools are active in the bitstream payload in order to arrive at the actual time domain signal as described by the input bitstream payload.

For each of the optional tools that operate on the signal data, the option to "pass through" is retained, and in all cases where the processing is omitted, the spectra or time samples at its input are passed directly through the tool without modification.

In places where the bitstream changes its signal representation from time domain to frequency domain representation or from LP domain to non-LP domain or vice versa, the decoder shall facilitate the transition from one domain to the other by means of an appropriate transition mechanism.

eSBR and MPS212 processing is applied in the same manner to both coding paths after transition handling.

The USAC specification offers in some instances multiple decoding options that serve to provide different quality / complexity trade-offs.

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

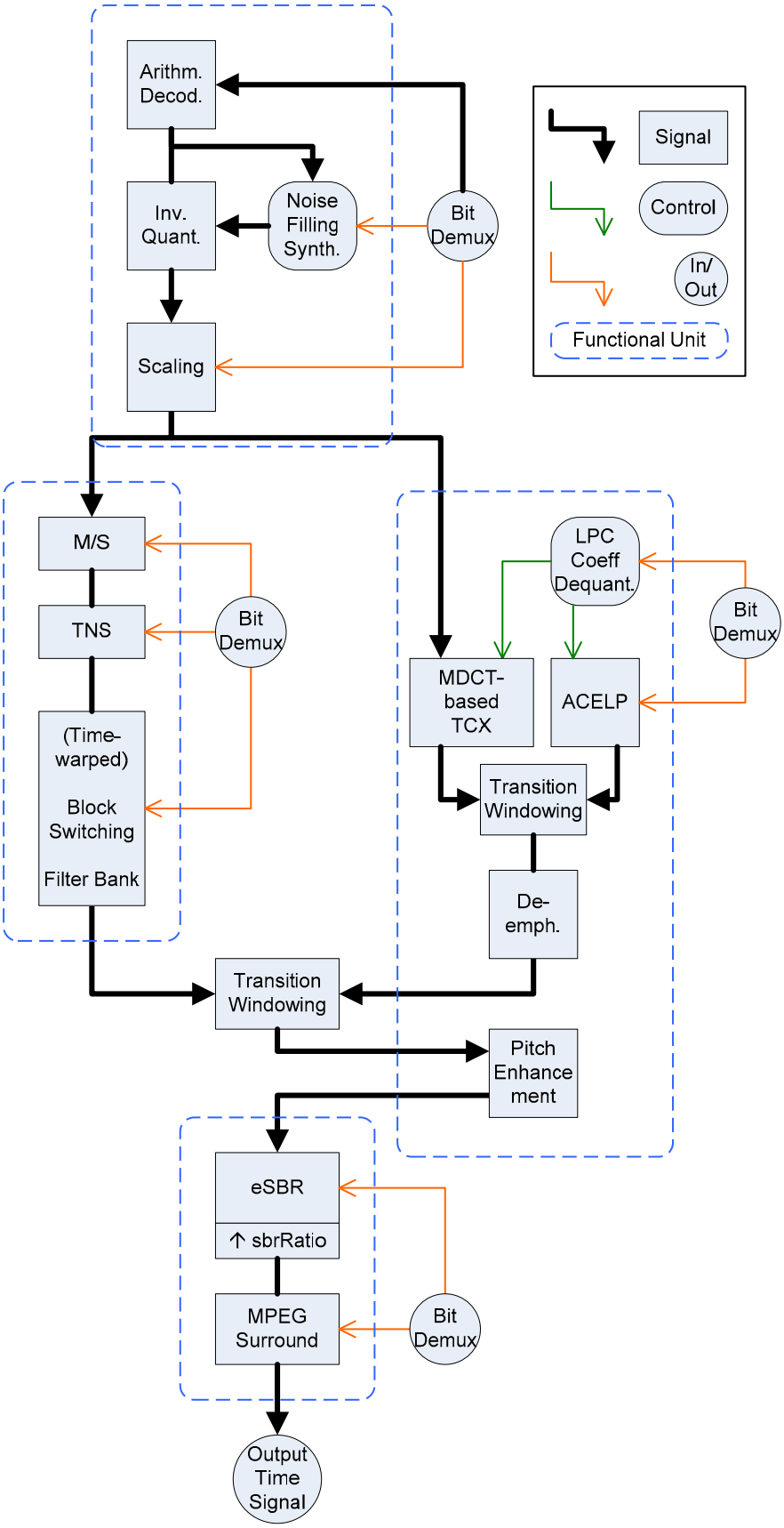

Figure 1 — Simplified block diagram of the typical USAC decoder configuration

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

4.2 Overview of the decoder tools

The input to the bitstream payload demultiplexer tool is the MPEG-D USAC bitstream payload. The demultiplexer separates the bitstream payload into the parts for each tool, and provides each of the tools with the bitstream payload information related to that tool.

The outputs from the bitstream payload demultiplexer tool are:

- Depending on the core coding type in the current frame either:
  - Tthe quantized and noiselessly coded spectra represented by
    - Scalefactor information
    - Arithmetically coded spectral lines
  - or: linear prediction (LP) parameters together with an excitation signal represented by either:
    - Quantized and arithmetically coded spectral lines (transform coded excitation, TCX) or
    - ACELP coded time domain excitation
- The spectral noise filling information (optional)
- The M/S decision information (optional)
- The temporal noise shaping (TNS) information (optional)
- The filterbank control information
- The time unwarping (TW) control information (optional)
- The enhanced spectral bandwidth replication (eSBR) control information (optional)
- The MPEG Surround 2-1-2 (MPS212) control information (optional)

The scalefactor noiseless decoding tool takes information from the bitstream payload demultiplexer, parses that information, and decodes the Huffman and DPCM coded scalefactors.

The input to the scalefactor noiseless decoding tool is:

- The scalefactor information for the noiselessly coded spectra

The output of the scalefactor noiseless decoding tool is:

- The decoded integer representation of the scalefactors:

The spectral noiseless decoding tool takes information from the bitstream payload demultiplexer, parses that information, decodes the arithmetically coded data, and reconstructs the quantized spectra.The input to this noiseless decoding tool is:

- The noiselessly coded spectra

The output of this noiseless decoding tool is:

- The quantized values of the spectra

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

The inverse quantizer tool takes the quantized values for the spectra, and converts the integer values to the non-scaled, reconstructed spectra. This quantizer is a companding quantizer, whose companding factor depends on the chosen core coding mode.

The input to the Inverse Quantizer tool is:

— The quantized values for the spectra

The output of the inverse quantizer tool is:

— The un-scaled, inversely quantized spectra

The noise filling tool is used to fill spectral gaps in the decoded spectra, which occur when spectral value are quantized to zero e.g. due to a strong restriction on bit demand in the encoder. The use of the noise filling tool is optional.

The inputs to the noise filling tool are:

— The un-scaled, inversely quantized spectra

— Noise filling parameters

— The decoded integer representation of the scalefactors

The outputs to the noise filling tool are:

— The un-scaled, inversely quantized spectral values for spectral lines which were previously quantized to zero.

— Modified integer representation of the scalefactors

The rescaling tool converts the integer representation of the scalefactors to the actual values, and multiplies the un-scaled inversely quantized spectra by the relevant scalefactors.

The inputs to the scalefactors tool are:

— The decoded integer representation of the scalefactors

— The un-scaled, inversely quantized spectra

The output from the scalefactors tool is:

— The scaled, inversely quantized spectra

For an overview over the M/S tool, please refer to ISO/IEC 14496-3:2009, 4.1.1.2.

For an overview over the temporal noise shaping (TNS) tool, please refer to ISO/IEC 14496-3:2009, 4.1.1.2.

The filterbank / block switching tool applies the inverse of the frequency mapping that was carried out in the encoder. An inverse modified discrete cosine transform (IMDCT) is used for the filterbank tool. The IMDCT can be configured to support 120, 128, 240, 256, 480, 512, 960 or 1024 spectral coefficients.

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

The inputs to the filterbank tool are:

— The (inversely quantized) spectra

— The filterbank control information

The output(s) from the filterbank tool is (are):

— The time domain reconstructed audio signal(s).

The time-warped filterbank / block switching tool replaces the normal filterbank / block switching tool when the time warping mode is enabled. The filterbank is the same (IMDCT) as for the normal filterbank, but in addition the windowed time domain samples are mapped from the warped time domain to the linear time domain by time-varying resampling.

The inputs to the time-warped filterbank tools are:

— The inversely quantized spectra

— The filterbank control information

— The time-warping control information

The output(s) from the filterbank tool is (are):

— The linear time domain reconstructed audio signal(s).

The enhanced SBR (eSBR) tool regenerates the highband of the audio signal. It is based on replication of the sequences of harmonics, truncated during encoding. It adjusts the spectral envelope of the generated high-band and applies inverse filtering, and adds noise and sinusoidal components in order to recreate the spectral characteristics of the original signal.

The input to the eSBR tool is:

— The quantized envelope data

— Control data

— A time domain signal from the frequency domain core decoder or the ACELP/TCX core decoder

The output of the eSBR tool is either:

— A time domain signal or

— A QMF-domain representation of a signal, e.g. in case MPS212 is used.

The MPEG Surround 2-1-2 (MPS212) tool produces multiple signals from one input signal by applying a sophisticated upmix procedure to the input signal controlled by appropriate spatial parameters. In the USAC context MPS212 is used for coding a stereo signal, by transmitting parametric side information alongside a transmitted downmixed signal.

The input to the MPS212 tool is:

— A downmixed time domain signal or

— A QMF-domain representation of a downmixed signal from the eSBR tool

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

The output of the MPS212 tool is:

— A two-channel time domain signal

The ACELP tool provides a way to efficiently represent a time domain excitation signal by combining a long term predictor (adaptive codebook codeword) with a pulse-like sequence (innovation codebook codeword). The reconstructed excitation is sent through an LP synthesis filter to form a time domain signal.

The input to the ACELP tool is:

— Adaptive and innovation codebook indices

— Adaptive and innovation codes gain values

— Other control data

— Inversely quantized and interpolated LPC filter coefficients

The output of the ACELP tool is:

— The time domain reconstructed audio signal

The MDCT based TCX decoding tool is used to turn the weighted LP residual representation from an MDCT-domain back to the time domain and outputs a time domain signal in which weighted LP synthesis filtering has been applied. The IMDCT can be configured to support 256, 512, or 1024 spectral coefficients.

The input to the TCX tool is:

— The (inversely quantized) MDCT spectra

— Inversely quantized and interpolated LPC filter coefficients

The output of the TCX tool is:

— The time domain reconstructed audio signal

4.3 Combination of USAC with MPEG Surround and SAOC

The output of the USAC decoder can be further processed by MPEG Surround (MPS) (ISO/IEC 23003-1) or Spatial Audio Object Coding (SAOC) (ISO/IEC 23003-2). If the SBR tool in USAC is active, a USAC decoder can typically be efficiently combined with a subsequent MPS/SAOC decoder by connecting them in the QMF domain in the same way as it is described for HE-AAC in ISO/IEC 23003-1:2007, 4.4. If a connection in the QMF domain is not possible, they need to be connected in the time domain.

If MPS/SAOC side information is embedded into a USAC bitstream by means of the usacExtElement mechanism (with usacExtElementType being ID_EXT_ELE_MPEGS or ID_EXT_ELE_SAOC), the time-alignment between the USAC data and the MPS/SAOC data assumes the most efficient connection between the USAC decoder and the MPS/SAOC decoder. If the SBR tool in USAC is active and if MPS/SAOC employs a 64 band QMF domain representation (see ISO/IEC 23003-1:2007, 6.6.3), the most efficient connection is in the QMF domain. Otherwise, the most efficient connection is in the time domain. This corresponds to the time-alignment for the combination of HE-AAC and MPS as defined in ISO/IEC 23003-1:2007, 4.4, 4.5, and 7.2.1.

The additional delay introduced by adding MPS decoding after USAC decoding is given by ISO/IEC 23003-1:2007, 4.5 and depends on whether HQ MPS or LP MPS is used, and whether MPS is connected to USAC in the QMF domain or in the time domain.

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

4.4 Interface between USAC and Systems

This subclause clarifies the interface between USAC and MPEG Systems. Every access unit delivered to the audio decoder from the systems interface shall result in a corresponding composition unit delivered from the audio decoder to the systems interface, i.e., the compositor. This shall include start-up and shut-down conditions, i.e., when the access unit is the first or the last in a finite sequence of access units.

For an audio composition unit, ISO/IEC 14496-1:2010, 7.1.3.5 Composition Time Stamp (CTS) specifies that the composition time applies to the *n*-th audio sample within the composition unit. For USAC, the value of *n* is always 1. Note that this applies to the output of the USAC decoder itself. In the case that a USAC decoder is, for example, being combined with an MPS decoder as described in 4.3, the additional delay caused by the MPS decoding process (see 4.3 and ISO/IEC 23003-1:2007, 4.5) needs to be taken into account for the composition units delivered at the output of the MPS decoder.

4.5 USAC Profiles and Levels

4.5.1 Introduction

This subclause defines profiles and their levels for Unified Speech and Audio Coding.

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

4.5.2 MPEG-4 HE AACv2 Compatibility

Large parts of the USAC codec are inherited from the codec tools and structure subsumed in the MPEG-4 HE AAC v2 profile. A few of these tools have been adopted into USAC as is. Many more have been adopted into USAC and greatly enhanced in terms of performance, capability and flexibility. Others were substituted with tools which provide a range of advantages over their MPEG-4 counterparts. As a result, USAC retains all *functionalities and performance features* that the AAC family of technologies – AAC, HE AAC, HE AAC v2 – can provide. However, it does not adopt all *tools*.

If a decoder is intended to provide full AAC family functionality, including the legacy MPEG-4 AAC tools, all coding tools listed in Table 1 shall be considered.

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

The following tools listed in Table 1 are normatively referenced in USAC:

Table 1 — Summary of the Location of and Normative Reference to the Definitions of all AAC, HE-AAC and USAC Coding Tools as employed in the Extended High Efficiency AAC profile

Tool / Module		defined in ISO/IEC	sub-clause	USAC	AAC LC	SBR	PS
block switching		14496-3	4.6.11	X	X		
window shapes	AAC based	14496-3	4.6.11	X	X		
	additional USAC	23003-3		X			
filter bank	standard	14496-3	4.6.11	X	X		
	time-warped	23003-3		X			
TNS		14496-3	4.6.9	X	X		
intensity		14496-3	4.6.8.2	NOTE 1	X		
coupling		14496-3	4.6.8.3		X		
perceptual noise synthesis	PNS	14496-3	4.6.13	NOTE 2	X		
	noise filling	23003-3		X			
MS	basic mid/side coding	14496-3	4.6.8.1	X	X		
	MDCT based complex prediction	23003-3		X			
quantization	non-uniform	14496-3	4.6.1	X	X		
	uniform	23003-3		X			
entropy coding	Huffman	14496-3	4.6.3	NOTE 3	X		
	context adaptive arithmetic coding	23003-3		X			
SBR	base	14496-3	4.6.18	X		X	X
	enhanced	23003-3		X			
parametric stereo extension	Parametric Stereo	14496-3	8.6.4 / 8.A	NOTE 4			X
	MPEG Surround 2-1-2 (incl. residual coding)	23003-3		X			
ACELP		23003-3		X			
frequency domain noise shaping	scale factor based	14496-3	4.6.2	X	X		
	LPC based	23003-3		X			

NOTE 1: Functionality of the AAC LC intensity tool is fully provided by the MDCT based complex prediction tool of USAC
NOTE 2: Functionality of the PNS tool is largely provided by the noise filling tool of USAC
NOTE 3: Functionality of the AAC LC Huffman coding tool is fully provided by the context adaptive arithmetic coding tool of USAC
NOTE 4: Functionality of the Parametric Stereo tool is fully provided by the MPEG Surround 2-1-2 tool of USAC

4.5.3 Baseline USAC Profile

In the Baseline USAC profile the following coding tools shall not be employed:

— Time warped filterbank

— DFT based harmonic transposer in enhanced spectral band replication

— Fractional delay decorrelator in MPEG Surround for mono to stereo upmixing (MPS212)

Four different hierarchical levels are defined with increasing number of audio channels and increasing complexity. The definition of the four levels of the Baseline USAC profile is given in Table 2.

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 2 — Levels for the Baseline USAC profile

Level	Max. channels	Max. sampling rate [kHz]	Max. PCU	Max. RCU
1	1	48	7	6
2	2	48	12	11
3	5.1	48	31	28
4	5.1	96	62	28

4.5.4 Extended High Efficiency AAC Profile

The Extended HE AAC profile contains the audio object types 42 (USAC), 5 (SBR), 29 (PS) and 2 (AAC LC) as defined in ISO/IEC 14496-3. In order for a decoder to support the Extended HE AAC profile it shall implement all modules listed in Table 1.

The Extended HE AAC profile is compatible with the MPEG-4 High Efficiency AAC v2 profile as defined in ISO/IEC 14496-3. It warrants decodability of HE AAC v2 profile compliant bit streams by Extended HE AAC profile decoders.

Four different hierarchical levels are defined with increasing number of audio channels and increasing complexity. All four levels include Level 2 of the Baseline USAC profile. The definition of the four levels of the Extended HE AAC profile is given in Table 3. All notes in Table 3 and all restrictions listed in the columns 2, 3, 4, and 5 (“Max. channels/object”, “Max. AAC sampling rate, SBR not present [kHz]”, “Max. AAC sampling rate, SBR present [kHz]”, “Max. SBR sampling rate [kHz] (in/out)”) of Table 3 apply only when decoding HE AAC v2 profile compliant bit streams.

Table 3 — Levels for the Extended HE AAC profile

Level (NOTE 1)	Max. channels/ object	Max. AAC sampling rate, SBR not present [kHz]	Max. AAC sampling rate, SBR present [kHz]	Max. SBR sampling rate [kHz] (in/out)	Max. PCU	Max. RCU	Max. PCU HQ / LP SBR (NOTE 5)	Max. RCU HQ / LP SBR (NOTE 5)
1	NA	NA	NA	NA	NA	NA	NA	NA
2	2	48	24	24/48	12	11	12	11
3	2	48	24/48 (NOTE 3)	48/48 (NOTE 2)	15	11	15	11
4	5	48	24/48 (NOTE 4)	48/48 (NOTE 2)	25	28	20	23
5	5	96	48	48/96	49	28	39	23

NOTE 1: Level 2, 3, and 4 Extended HE AAC profile decoders implement the baseline version of the parametric stereo tool. A level 5 decoder shall not be limited to the baseline version of the parametric stereo tool.
NOTE 2: For level 3 and level 4 decoders, it is mandatory to operate the SBR tool in downsampled mode if the sampling rate of the AAC core is higher than 24kHz. Hence, if the SBR tool operates on a 48kHz signal, the internal sampling rate of the SBR tool will be 96kHz, however, the output signal will be downsampled by the SBR tool to 48kHz.
NOTE 3: If Parametric Stereo data is present the maximum AAC sampling rate is 24kHz, if Parametric Stereo data is not present the maximum AAC sampling rate is 48kHz.
NOTE 4: For one or two channels the maximum AAC sampling rate, with SBR present, is 48kHz. For more than two channels the maximum AAC sampling rate, with SBR present, is 24kHz.
NOTE 5: The PCU/RCU number are given for a decoder operating the LP SBR tool whenever applicable.

For the MPEG-4 audio object type 2 (AAC LC), mono or stereo mixdown elements are not permitted.

For MPEG-4 audio object types 2, 5, and 29 the following restrictions apply:

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

— An Extended HE AAC profile decoder shall operate the HQ SBR tool for bitstreams containing Parametric Stereo data.

— For bitstreams not containing Parametric Stereo data, the Extended HE AAC profile decoder may operate the HQ SBR tool, or the LP SBR tool.

— Only bitstreams consisting of exactly one AAC single channel element may contain Parametric Stereo data. Bitstreams containing more than one channel in the AAC part shall not contain Parametric Stereo data.

5 Syntax

5.1 General

The bit stream syntax shall be based on ISO/IEC 14496-3:2009, 4.4.

The USAC bit stream syntax is shown below:

5.2 Decoder configuration (UsacConfig)

Table 4 — Syntax of UsacConfig()

Syntax	No. of bits	Mnemonic
UsacConfig()		
{		
usacSamplingFrequencyIndex;	**5**	**bslbf**
if (usacSamplingFrequencyIndex == 0x1f) {		
usacSamplingFrequency;	**24**	**uimsbf**
}		
coreSbrFrameLengthIndex;	**3**	**uimsbf**
channelConfigurationIndex;	**5**	**uimsbf**
if (channelConfigurationIndex == 0) {		
UsacChannelConfig();		
}		
UsacDecoderConfig();		
if (**usacConfigExtensionPresent**==1) {	**1**	**uimsbf**
UsacConfigExtension();		
}		
}		

Table 5 — Syntax of UsacChannelConfig()

Syntax	No. of bits	Mnemonic
UsacChannelConfig()		
{		
numOutChannels = escapedValue(5,8,16);		
for (i=0; i<numOutChannels; i++) {		
bsOutputChannelPos[i];	**5**	**uimsbf**
}		
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 6 — Syntax of UsacDecoderConfig()

Syntax	No. of bits	Mnemonic
UsacDecoderConfig()		
{		
numElements = escapedValue(4,8,16) + 1**;**		
for (elemIdx=0; elemIdx<numElements; ++elemIdx) {		
usacElementType[elemIdx**]**	**2**	**uimsbf**
switch (usacElementType[elemIdx]) {		
case: ID_USAC_SCE		
UsacSingleChannelElementConfig(sbrRatioIndex);		
break;		
case: ID_USAC_CPE		
UsacChannelPairElementConfig(sbrRatioIndex);		
break;		
case: ID_USAC_LFE		
UsacLfeElementConfig();		
break;		
case: ID_USAC_EXT		
UsacExtElementConfig();		
break;		
}		
}		
NOTE: UsacSingleChannelElementConfig(), UsacChannelPairElementConfig(), UsacLfeElement-Config() and UsacExtElementConfig() signaled at position elemIdx refer to the corresponding elements in UsacFrame() at the respective position elemIdx.		

Table 7 — Syntax of UsacSingleChannelElementConfig()

Syntax	No. of bits	Mnemonic
UsacSingleChannelElementConfig(sbrRatioIndex)		
{		
UsacCoreConfig();		
if (sbrRatioIndex > 0) {		
SbrConfig();		
}		
}		

Table 8 — Syntax of UsacChannelPairElementConfig()

Syntax	No. of bits	Mnemonic
UsacChannelPairElementConfig(sbrRatioIndex)		
{		
UsacCoreConfig();		
if (sbrRatioIndex > 0) {		
SbrConfig();		
stereoConfigIndex;	**2**	**uimsbf**
}		
else {		
stereoConfigIndex = 0;		
}		
if (stereoConfigIndex > 0) {		
Mps212Config(stereoConfigIndex);		
}		
}		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 9 — Syntax of UsacLfeElementConfig()

Syntax	No. of bits	Mnemonic
UsacLfeElementConfig()		
{		
tw_mdct = 0;		
noiseFilling = 0;		
}		

Table 10 — Syntax of UsacCoreConfig()

Syntax	No. of bits	Mnemonic
UsacCoreConfig()		
{		
tw_mdct;	**1**	**bslbf**
noiseFilling;	**1**	**bsblf**
}		

Table 11 — Syntax of SbrConfig()

Syntax	No. of bits	Mnemonic
SbrConfig()		
{		
harmonicSBR;	**1**	**bsblf**
bs_interTes;	**1**	**bsblf**
bs_pvc;	**1**	**bsblf**
SbrDfltHeader();		
}		

Table 12 — Syntax of SbrDfltHeader()

Syntax	No. of bits	Mnemonic
SbrDfltHeader()		
{		
dflt_start_freq;	**4**	**uimsbf**
dflt_stop_freq;	**4**	**uimsbf**
dflt_header_extra1;	**1**	**uimsbf**
dflt_header_extra2;	**1**	**uimsbf**
if (dflt_header_extra1 == 1) {		
dflt_freq_scale;	**2**	**uimsbf**
dflt_alter_scale;	**1**	**uimsbf**
dflt_noise_bands;	**2**	**uimsbf**
}		
if (dflt_header_extra2 == 1) {		
dflt_limiter_bands;	**2**	**uimsbf**
dflt_limiter_gains;	**2**	**uimsbf**
dflt_interpol_freq;	**1**	**uimsbf**
dflt_smoothing_mode;	**1**	**uimsbf**
}		
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 13 — Syntax of Mps212Config()

Syntax	No. of bits	Mnemonic
Mps212Config(stereoConfigIndex)		
{		
bsFreqRes;	**3**	**uimsbf**
bsFixedGainDMX	**3**	**uimsbf**
bsTempShapeConfig;	**2**	**uimsbf**
bsDecorrConfig;	**2**	**uimsbf**
bsHighRateMode;	**1**	**uimsbf**
bsPhaseCoding;	**1**	**uimsbf**
bsOttBandsPhasePresent;	**1**	**uimsbf**
if (bsOttBandsPhasePresent) {		NOTE 1
bsOttBandsPhase;	**5**	**uimsbf**
}		
if (bsResidualCoding) {		**NOTE 2**
bsResidualBands;	**5**	**uimsbf**
bsOttBandsPhase = max(bsOttBandsPhase,bsResidualBands);		
bsPseudoLr;	**1**	**uimsbf**
}		
if (bsTempShapeConfig == 2) {		
bsEnvQuantMode;	**1**	**uimsbf**
}		
}		
NOTE 1: if bsOttBandsPhasePresent==0 bsOttBandsPhase is initialized according toTable 104.		
NOTE 2: bsResidualCoding depends on stereoConfigIndex according to Table 72		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 14 — Syntax of UsacExtElementConfig()

Syntax	No. of bits	Mnemonic
UsacExtElementConfig()		
{		
usacExtElementType = escapedValue(4,8,16);		
usacExtElementConfigLength = escapedValue(4,8,16);		
usacExtElementDefaultLengthPresent;	**1**	**uimsbf**
if (usacExtElementDefaultLengthPresent) {		
usacExtElementDefaultLength = escapedValue(8,16,0) + 1;		
} else {		
usacExtElementDefaultLength = 0;		
}		
usacExtElementPayloadFrag;	**1**	**uimsbf**
switch (usacExtElementType) {		
case ID_EXT_ELE_FILL:		
break;		
case ID_EXT_ELE_MPEGS:		
SpatialSpecificConfig();		
break;		
case ID_EXT_ELE_SAOC:		
SaocSpecificConfig();		
break;		
default:	NOTE	
while (usacExtElementConfigLength--) {		
tmp;	**8**	**uimsbf**
}		
break;		
}		
}		
NOTE: The default entry for the usacExtElementType is used for unknown extElementTypes so that legacy decoders can cope with future extensions.		

Table 15 — Syntax of UsacConfigExtension()

Syntax	No. of bits	Mnemonic
UsacConfigExtension()		
{		
numConfigExtensions = escapedValue(2,4,8) + 1;		
for (confExtIdx=0; confExtIdx<numConfigExtensions; confExtIdx++) {		
usacConfigExtType[confExtIdx] = escapedValue(4,8,16);		
usacConfigExtLength[confExtIdx] = escapedValue(4,8,16);		
switch (usacConfigExtType[confExtIdx]) {		
case ID_CONFIG_EXT_FILL:		
while (usacConfigExtLength[confExtIdx]--) {		
fill_byte[i]; /* should be '10100101' */	**8**	**uimsbf**
}		
break;		
default:		
while (usacConfigExtLength[confExtIdx]--) {		
tmp;	**8**	**uimsbf**
}		
break;		
}		
}		
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 16 — Syntax of escapedValue()

Syntax	No. of bits	Mnemonic
escapedValue(nBits1, nBits2, nBits3)		
{		
value;	**nBits1**	**uimsbf**
if (value == 2^{nBits1}-1) {		
value += **valueAdd;**	**nBits2**	**uimsbf**
if (valueAdd == 2^{nBits2}-1) {		
value += **valueAdd;**	**nBits3**	**uimsbf**
}		
}		
return value;		
}		

5.3 USAC bitstream payloads

5.3.1 Payloads for audio object type USAC

Table 17 — Syntax of UsacFrame(), top level payload for audio object type USAC

Syntax	No. of bits	Mnemonic
UsacFrame()		
{		
usacIndependencyFlag;	**1**	**uimsbf**
for (elemIdx=0; elemIdx<numElements; ++elemIdx) {		
switch (usacElementType[elemIdx]) {		
case: ID_USAC_SCE		
UsacSingleChannelElement(usacIndependencyFlag);		
break;		
case: ID_USAC_CPE		
UsacChannelPairElement(usacIndependencyFlag);		
break;		
case: ID_USAC_LFE		
UsacLfeElement(usacIndependencyFlag);		
break;		
case: ID_USAC_EXT		
UsacExtElement(usacIndependencyFlag);		
break;		
}		
}		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 18 — Syntax of UsacSingleChannelElement()

Syntax	No. of bits	Mnemonic

UsacSingleChannelElement(indepFlag)
{
  UsacCoreCoderData(1, indepFlag);
 	
  if (sbrRatioIndex > 0) {
  UsacSbrData(1, indepFlag);
  }
}		

Table 19 — Syntax of UsacChannelPairElement()

Syntax	No. of bits	Mnemonic

UsacChannelPairElement(indepFlag)
{
  if (stereoConfigIndex == 1) {
  nrCoreCoderChannels = 1;
  } else {
  nrCoreCoderChannels = 2;
  }
 	
  UsacCoreCoderData(nrCoreCoderChannels, indepFlag);
 	
  if (sbrRatioIndex > 0) {
  if (stereoConfigIndex == 0 || stereoConfigIndex == 3) {
  nrSbrChannels = 2;
  } else {
  nrSbrChannels = 1;
  }
  UsacSbrData(nrSbrChannels, indepFlag);
  }
 	
  if (stereoConfigIndex > 0) {
  Mps212Data(indepFlag);
  }
}		

Table 20 — Syntax of UsacLfeElement()

Syntax	No. of bits	Mnemonic

UsacLfeElement(indepFlag)
{
  fd_channel_stream(0,0,0,0, indepFlag);
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 21 — Syntax of UsacExtElement()

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

Table 22 — Syntax of ics_info()

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

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

5.3.2 Subsidiary payloads

Table 23 — Syntax of UsacCoreCoderData()

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

Table 24 — Syntax of StereoCoreToolInfo()

Syntax	No. of bits	Mnemonic
StereoCoreToolInfo(core_mode)		
{		
if (core_mode[0] == 0 && core_mode[1] == 0) {		
tns_active;	**1**	**uimsbf**
common_window;	**1**	**uimsbf**
if (common_window) {		
ics_info();		
common_max_sfb;	**1**	**uimsbf**
if (common_max_sfb == 0) {		
if (window_sequence == EIGHT_SHORT_SEQUENCE) {		
max_sfb1;	**4**	**uimsbf**
} else {		
max_sfb1;	**6**	**uimsbf**
}		
} else {		
max_sfb1 = max_sfb;		
}		
max_sfb_ste = max(max_sfb, max_sfb1);		
ms_mask_present;	**2**	**uimsbf**
if (ms_mask_present == 1) {		
for (g = 0; g < num_window_groups; g++) {		
for (sfb = 0; sfb < max_sfb; sfb++) {		
ms_used[g][sfb];	**1**	**uimsbf**

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

```
                    }
                }
            }
            if (ms_mask_present == 3) {
                cplx_pred_data();
            } else {
                alpha_q_re[g][sfb] = 0;
                alpha_q_im[g][sfb] = 0;
            }
        }
        if (tw_mdct) {
            common_tw;                                          1       uimsbf
            if ( common_tw ) {
                tw_data();
            }
        }
        if (tns_active) {
            if (common_window) {
                common_tns;                                     1       uimsbf
            } else {
                common_tns = 0;
            }
            tns_on_lr;                                          1       uimsbf
            if (common_tns) {
                tns_data();
                tns_data_present[0] = 0;
                tns_data_present[1] = 0;
            } else {
                tns_present_both;                               1       uimsbf
                if (tns_present_both) {
                    tns_data_present[0] = 1;
                    tns_data_present[1] = 1;
                } else {
                    tns_data_present[1];                        1       uimsbf
                    tns_data_present[0] = 1 - tns_data_present[1];
                }
            }
        } else {
            common_tns = 0;
            tns_data_present[0] = 0;
            tns_data_present[1] = 0;
        }
    } else {
        common_window = 0;
        common_tw = 0;
    }
}
```

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 25 — Syntax of fd_channel_stream()

Syntax	No. of bits	Mnemonic
fd_channel_stream(common_window, common_tw, tns_data_present, noiseFilling, indepFlag)		
{		
global_gain;	**8**	**uimsbf**
if (noiseFilling) {		
noise_level;	**3**	**uimsbf**
noise_offset;	**5**	**uimsbf**
}		
else {		
noise_level = 0;		
}		
if (!common_window) {		
ics_info();		
}		
if (tw_mdct) {		
if (!common_tw) {		
tw_data();		
}		
}		
scale_factor_data ();		
if (tns_data_present) {		
tns_data ();		
}		
ac_spectral_data(indepFlag);		
fac_data_present;	**1**	**uimsbf**
if (fac_data_present) {		
fac_length = (window_sequence==EIGHT_SHORT_SEQUENCE) ? ccfl/16 : ccfl/8;		
fac_data(1, fac_length);		
}		
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 26 — Syntax of cplx_pred_data()

Syntax	No. of bits	Mnemonic
cplx_pred_data(max_sfb_ste, indepFlag)		
{		
cplx_pred_all;	**1**	**uimsbf**
if (cplx_pred_all == 0) {		
for (g = 0; g < num_window_groups; g++) {		
for (sfb = 0; sfb < max_sfb_ste; sfb += SFB_PER_PRED_BAND) {		
cplx_pred_used[g][sfb];	**1**	**uimsbf**
if ((sfb+1) < max_sfb_ste) {		
cplx_pred_used[g][sfb+1] = cplx_pred_used[g][sfb];		
}		
}		
}		
}		
else {		
for (g = 0; g < num_window_groups; g++) {		
for (sfb = 0; sfb < max_sfb_ste; sfb++) {		
cplx_pred_used[g][sfb] = 1;		
}		
}		
}		
pred_dir;	**1**	**uimsbf**
complex_coef;	**1**	**uimsbf**
if (complex_coef) {		
if (indepFlag) {		
use_prev_frame = 0;		
} else {		
use_prev_frame;	**1**	**uimsbf**
}		
}		
if (indepFlag) {		
delta_code_time = 0;		
} else {		
delta_code_time;	**1**	**uimsbf**
}		
for (g = 0; g < num_window_groups; g++) {		
for (sfb = 0; sfb < max_sfb_ste; sfb += SFB_PER_PRED_BAND) {		
if (cplx_pred_used[g][sfb]) {		
hcod_sf[dpcm_alpha_q_re[g][sfb]];	**1..19**	**vlclbf**
if (complex_coef) {		
hcod_sf[dpcm_alpha_q_im[g][sfb]];	**1..19**	**vlclbf**
}		
else {		
alpha_q_im[g][sfb] = 0;		
dpcm_alpha_q_im[g][sfb] = 60;		
}		
}		
else {		
alpha_q_re[g][sfb] = 0;		
alpha_q_im[g][sfb] = 0;		
}		
}		
}		
}		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 27 — Syntax of tw_data()

Syntax	No. of bits	Mnemonic
tw_data( )		
{		
tw_data_present;	**1**	**uimsbf**
if (tw_data_present == 1) {		
for (i = 1 ; i < NUM_TW_NODES ; i++) {		
tw_ratio[i];	**3**	**uimsbf**
}		
}		
}		

Table 28 — Syntax of scale_factor_data()

Syntax	No. of bits	Mnemonic
scale_factor_data()		
{		
dpcm_sf[0][0] = 60;		
for (g = 0; g < num_window_groups; g++) {		
for (sfb = 0; sfb < max_sfb; sfb++) {		
if (g > 0 || sfb > 0) {		
hcod_sf[dpcm_sf[g][sfb]**];**	**1..19**	**vlclbf**
}		
}		
}		
}		

Table 29 — Syntax of tns_data()

Syntax	No. of bits	Mnemonic
tns_data()		
{		
for (w = 0; w < num_windows; w++) {		
n_filt[w];	**1..2**	**uimsbf**
if (n_filt[w]) {		
coef_res[w];	**1**	**uimsbf**
}		
for (filt = 0; filt < n_filt[w]; filt++) {		
length[w][filt];	**{4;6}**	**uimsbf**
order[w][filt];	**{3;4}**	**uimsbf**
if (order[w][filt]) {		
direction[w][filt];	**1**	**uimsbf**
coef_compress[w][filt];	**1**	**uimsbf**
for (i = 0; i < order[w][filt]; i++) {		
coef[w][filt][i];	**2..4**	**uimsbf**
}		
}		
}		
}		
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 30 — Syntax of ac_spectral_data()

Syntax	No. of bits	Mnemonic
ac_spectral_data(indepFlag)		
{		
if (indepFlag) {		
arith_reset_flag = 1;		
} else {		
arith_reset_flag;	**1**	**uimsbf**
}		
for (win = 0; win < num_windows; win++) {		NOTE 1
arith_data(lg, arith_reset_flag && (win==0));		
}		
}		
NOTE 1: num_windows indicates the number of windows in the current window_sequence. In case window_sequence is EIGHT_SHORT_SEQUENCE num_windows equals 8. In all other cases num_windows equals 1		

Table 31 — Syntax of lpd_channel_stream()

Syntax	No. of bits	Mnemonic
lpd_channel_stream(indepFlag)		
{		
acelp_core_mode;	**3**	**uimsbf**
lpd_mode;	**5**	**uimsbf,** NOTE 1
bpf_control_info	**1**	**uimsbf**
core_mode_last;	**1**	**uimsbf**
fac_data_present;	**1**	**uimsbf**
first_lpd_flag = !core_mode_last;		
first_tcx_flag=TRUE;		
k = 0;		
if (first_lpd_flag) { last_lpd_mode = -1; }		NOTE 2
while (k < 4) {		
if (k==0) {		
if ((core_mode_last==1) && (fac_data_present==1)) {		
fac_data(0, ccfl/8);		
}		
} else {		
if ((last_lpd_mode==0 && mod[k]>0) ||		
(last_lpd_mode>0 && mod[k]==0)) {		
fac_data(0, ccfl/8);		
}		
}		
if (mod[k] == 0) {		
acelp_coding(acelp_core_mode);		
last_lpd_mode=0;		
k += 1;		
}		
else {		
tcx_coding(lg(mod[k]) , first_tcx_flag, indepFlag);		NOTE 3
last_lpd_mode=mod[k];		
k += (1 << (mod[k]-1));		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

first_tcx_flag=FALSE;		
}		
}		
lpc_data(first_lpd_flag);		
if ((core_mode_last==0) && (fac_data_present==1)) {		
short_fac_flag;	**1**	**uimsbf**
fac_length = short_fac_flag ? ccfl/16 : ccfl/8;		
fac_data(1, fac_length);		
}		
}		

NOTE 1: **lpd_mode** defines the contents of the array mod[] as described in 6.2.10.2, Table 89.
NOTE 2: first_lpd_flag is defined in 6.2.10.2.
NOTE 3: The number of spectral coefficients, lg, depends on mod[k] according to Table 148.

Table 32 — Syntax of lpc_data()

Syntax	No. of bits	Mnemonic
lpc_data(first_lpd_flag, mod[])		
{		
lpc_set = 4;		
mode_lpc = get_mode_lpc(lpc_set);		
lpc_first_approximation_index[lpc_set]	**8**	**uimsbf**
code_book_indices(lpc_set, nk_mode, 2);	NOTE 1	
if (first_lpd_flag) {		
lpc_set = 0;		
mode_lpc = get_mode_lpc(lpc_set);		
if (mode_lpc == 0) { **lpc_first_approximation_index[lpc_set]** }	**8**	**uimsbf**
code_book_indices(0, nk_mode, 2);	NOTE 1	
}		
if (mod[0] != 3) {		
lpc_set = 2;		
mode_lpc = get_mode_lpc(lpc_set);		
if (mode_lpc == 0) { **lpc_first_approximation_index[lpc_set]** }	**8**	**uimsbf**
code_book_indices(lpc_set, nk_mode, 2);	NOTE 1	
}		
if (mod[0] < 2) {		
lpc_set = 1;		
mode_lpc = get_mode_lpc(lpc_set);		
if (mode_lpc == 0) { **lpc_first_approximation_index[lpc_set]** }	**8**	**uimsbf**
if (mode_lpc != 1) {		
code_book_indices(lpc_set, nk_mode, 2);	NOTE 1	
}		
}		
if (mod[2] < 2) {		
lpc_set = 3;		
mode_lpc = get_mode_lpc(lpc_set);		
if (mode_lpc == 0) { **lpc_first_approximation_index[lpc_set]** }	**8**	**uimsbf**
code_book_indices(lpc_set, nk_mode, 2);	NOTE 1	
}		
}		

NOTE 1: nk_mode is determined by the number of the currently decoded LPC Filter set, lpc_set, and the LPC quantization mode, mode_lpc, according to Table 143.

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 33 — Syntax of qn_data()

Syntax	No. of bits	Mnemonic
qn_data(nk_mode, no_qn)		
{		
switch (nk_mode) {		
case 1:		
for (k=0; k<no_qn; k++) {		
qn[k]	**1..n**	**uclbf**
if (qn[k] > 0) { qn[k] += 1 }		
}		
break;		
case 0:		
case 2:		
case 3:		
for (k=0; k<no_qn; k++) {		
qn_base	**2**	**uimsbf**
qn[k] = qn_base + 2;		
}		
if (nk_mode == 2) {		
for (k=0; k<no_qn; k++) {		
if (qn[k] > 4) {		
qn[k]	**1..n**	**uclbf**
if (qn[k] > 0) { qn[k] += 4 }		
}		
}		
} else {		
for (k=0; k<no_qn; k++) {		
if (qn[k] > 4) {		
qn_ext	**1..n**	**uclbf**
switch (qn_ext) {		
case 0: qn[k] = 5; break;		
case 1: qn[k] = 6; break;		
case 2: qn[k] = 0; break;		
default: qn[k] = qn_ext + 4; break;		
}		
}		
}		
}		
break;		
}		
}		
}		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 34 — Syntax of get_mode_lpc()

Syntax	No. of bits	Mnemonic
get_mode_lpc(lpc_set)		
{		
switch (lpc_set) {		
case 4: mode_lpc=0; break;		
case 0:		
case 2:		
mode_lpc = **binary_code**	**1**	
break;		
case 1:		
switch (**binary_code**) {	**1..2**	**vlclbf**
case ‘0_2’: mode_lpc = 2; break;		
case ‘10_2’: mode_lpc = 0; break;		
case ‘11_2’: mode_lpc = 1; break;		
}		
break;		
case 3:		
switch (**binary_code**) {	**1..3**	**vlclbf**
case ‘0_2’: mode_lpc = 1; break;		
case ‘10_2’: mode_lpc = 0; break;		
case ‘110_2’: mode_lpc = 2; break;		
case ‘111_2’: mode_lpc = 3; break;		
}		
break;		
}		
return mode_lpc;		
}		
NOTE: The mapping of binary code to mode_lpc can also be deduced from Table 143		

Table 35 — Syntax code_book_indices ()

Syntax	No. of bits	Mnemonic
code_book_indices(idx, nk_mode, no_qn)	NOTE 1	
{		
for (k=0; k<no_qn; k++) {		
qn_data(nk_mode, no_qn)		
}		
for (k=0; k<no_qn; k++)		
{		
if (qn[k] > 4) {		
nk = (qn[k]-3)/2;		
n = qn[k] – nk*2;		
} else {		
nk = 0;		
n = qn[k];		
}		
code_book_index[idx][k]	**4*n**	**uimsbf**
kv[idx][k][0]	**nk**	**uimsbf**
kv[idx][k][1]	**nk**	**uimsbf**
kv[idx][k][2]	**nk**	**uimsbf**
kv[idx][k][3]	**nk**	**uimsbf**

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

kv[idx][k][4]	**nk**	**uimsbf**
kv[idx][k][5]	**nk**	**uimsbf**
kv[idx][k][6]	**nk**	**uimsbf**
kv[idx][k][7]	**nk**	**uimsbf**
}		
}		

NOTE 1: idx can take values from 0 to 4 in case the syntax element is used in context of lpc_data(). In case of the use in the context of fac_data() idx can take values from 0 to 7 or from 0 to 15 depending on fac_length.

Table 36 — Syntax of acelp_coding()

Syntax	No. of bits	Mnemonic
acelp_coding(acelp_core_mode)		
{		
mean_energy;	**2**	**uimsbf**
nb_subfr = coreCoderFrameLength/256		NOTE
for (sfr=0; sfr<nb_subfr ; sfr++) {		
if ((sfr==0) || ((nb_subfr==4) && (sfr==2))) {		
acb_index[sfr];	**9**	**uimsbf**
} else {		
acb_index[sfr];	**6**	**uimsbf**
}		
ltp_filtering_flag[sfr];	**1**	**bmsbf**
switch (acelp_core_mode) {		
case 0		
icb_index[sfr];	**20**	**uimsbf**
break;		
case 1		
icb_index[sfr];	**28**	**uimsbf**
break;		
case 2		
icb_index[sfr];	**36**	**uimsbf**
break;		
case 3		
icb_index[sfr];	**44**	**uimsbf**
break;		
case 4		
icb_index[sfr];	**52**	**uimsbf**
break;		
case 5		
icb_index[sfr];	**64**	**uimsbf**
break;		
}		
gains[sfr];	**7**	**uimsbf**
}		

NOTE: coreCoderFrameLength designates the core frame length in samples and is equal to either 1024 or 768. See also 6.1.1.2.

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 37 — Syntax of tcx_coding()

Syntax	No. of bits	Mnemonic
tcx_coding(lg, first_tcx_flag, indepFlag)		
{		
noise_factor;	**3**	**uimsbf**
global_gain;	**7**	**uimsbf**
if (first_tcx_flag) {		
if (indepFlag) {		
arith_reset_flag = 1;		
} else {		
arith_reset_flag;	**1**	**uimsbf**
}		
}		
else {		
arith_reset_flag=0;		
}		
arith_data(lg, arith_reset_flag);		
}		

Table 38 — Syntax of arith_data()

Syntax	No. of bits	Mnemonic
arith_data(lg, arith_reset_flag)		
{		
c = arith_map_context(N, arith_reset_flag);		
for (i=0; i<lg/2; i++) {		
/* MSB decoding */		
c = arith_get_context (c,i,N);		
for (lev=esc_nb=0;;) {		
pki = arith_get_pk(c+esc_nb<<17)		
acod_m[pki][m];	**1..20**	**vlclbf**
if (m != ARITH_ESCAPE)		
break;		
lev += 1;		
if ((esc_nb=lev)>7)		
esc_nb=7;		
}		
b = m>>2;		
a = m – (b<<2);		
/* ARITH_STOP symbol detection */		
if (m==0 && lev>0)		
break;		
/* LSB decoding */		
for (l=lev; l>0; l--) {		
lsbidx = (a==0)?1:((b==0)?0:2);		
acod_r[lsbidx][r];	**1..20**	**vlclbf**
a=(a<<1)|(r&1);		
b=(b<<1)|((r>>1)&1);		
}		
x_ac_dec[2*i] = a;		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

x_ac_dec[2*i+1] = b;		
arith_update_context(i,a,b);		
}		
arith_finish(x_ac_dec, i,N);		
/* Signs decoding */		
for (i=0; i<lg; i++) {		
if (x_ac_dec[i] != 0) {		
s;	**1**	**uimsbf**
if (s==0) { x_ac_dec[i] *= -1; }		
}		
}		
}		

Table 39 — Syntax of fac_data()

Syntax	No. of bits	Mnemonic
fac_data(useGain, fac_length)		
{		
if (useGain) {		
fac_gain;	**7**	**uimsbf**
}		
for (i=0; i<fac_length/8; i++) {		
code_book_indices (i, 1, 1);		
}		
}		
NOTE 1: This value is encoded using a modified unary code, where qn=0 is represented by one "0" bit, and any value qn greater or equal to 2 is represented by qn-1 "1" bits followed by one "0" stop bit. Note that qn=1 cannot be signaled, because the codebook Q_1 is not defined.		

5.3.3 Payloads for enhanced SBR

Table 40 — Syntax of UsacSbrData()

Syntax	No. of bits	Mnemonic
UsacSbrData(numberSbrChannels, indepFlag)		
{		
if (indepFlag) {		
sbrInfoPresent = 1;		
sbrHeaderPresent = 1;		
} else {		
sbrInfoPresent;	**1**	**uimsbf**
if (sbrInfoPresent) {		
sbrHeaderPresent;	**1**	**uimsbf**
} else {		
sbrHeaderPresent = 0;		
}		
}		
if (sbrInfoPresent) {		
SbrInfo();		
}		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

if (sbrHeaderPresent) {		
sbrUseDfltHeader;	**1**	**uimsbf**
if (sbrUseDfltHeader) {		
/* copy all SbrDfltHeader() elements		
dlft_xxx_yyy to bs_xxx_yyy */		
} else {		
SbrHeader();		
}		
}		
sbr_data(bs_amp_res, numberSbrChannels, indepFlag);		
}		

Table 41 — Syntax of SbrInfo

Syntax	No. of bits	Mnemonic
SbrInfo()		
{		
bs_amp_res;	**1**	**uimsbf**
bs_xover_band;	**4**	**uimsbf**
bs_sbr_preprocessing;	**1**	**uimsbf**
if (bs_pvc) {		
bs_pvc_mode;	**2**	**uimsbf**
}		
}		

Table 42 — Syntax of SbrHeader()

Syntax	No. of bits	Mnemonic
SbrHeader()		
{		
bs_start_freq;	**4**	**uimsbf**, NOTE 1
bs_stop_freq;	**4**	**uimsbf**, NOTE 1
bs_header_extra_1;	**1**	**uimsbf**
bs_header_extra_2;	**1**	**uimsbf**
if (bs_header_extra_1) {		NOTE 2
bs_freq_scale;	**2**	**uimsbf**
bs_alter_scale;	**1**	**uimsbf**
bs_noise_bands;	**2**	**uimsbf**
}		
if (bs_header_extra_2) {		NOTE 2
bs_limiter_bands;	**2**	**uimsbf**
bs_limiter_gains;	**2**	**uimsbf**
bs_interpol_freq;	**1**	**uimsbf**
bs_smoothing_mode;	**1**	**uimsbf**
}		
}		

NOTE 1: bs_start_freq and bs_stop_freq shall define a frequency band that does not exceed the limits defined in 7.5.5 and ISO/IEC 14496-3:2009, 4.6.18.3.6.
NOTE 2: If this bit is not set the default values for the underlying data elements shall be used disregarded any previous value.

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 43 — Syntax of sbr_data()

Syntax	No. of bits	Mnemonic
sbr_data(bs_amp_res, numberSbrChannels, indepFlag)		
{		
switch (numberSbrChannels) {		
case 1:		
sbr_single_channel_element(bs_amp_res, bs_pvc_mode, indepFlag);		
break;		
case 2:		
sbr_channel_pair_element(bs_amp_res, indepFlag);		
break;		
}		
}		

Table 44 — Syntax of sbr_single_channel_element()

Syntax	No. of bits	Mnemonic
sbr_single_channel_element(bs_amp_res, bs_pvc_mode, indepFlag)		
{		
if (harmonicSBR) {		
if (**sbrPatchingMode[0]** == 0) {	**1**	**uimsbf**
sbrOversamplingFlag[0];	**1**	**uimsbf**
if (**sbrPitchInBinsFlag[0]**)	**1**	**uimsbf**
sbrPitchInBins[0];	**7**	**uimsbf**
else		
sbrPitchInBins[0] = 0;		
} else {		
sbrOversamplingFlag[0] = 0;		
sbrPitchInBins[0] = 0;		
}		
}		
sbr_grid(0, bs_pvc_mode);		
sbr_dtdf(0, bs_pvc_mode, indepFlag);		
sbr_invf(0);		
if (bs_pvc_mode==0) {		
sbr_envelope(0, 0, bs_amp_res);		
} else {		
pvc_envelope(indepFlag);		
}		
sbr_noise(0, 0);		
if (**bs_add_harmonic_flag**[0]) {	**1**	**uimsbf**
sbr_sinusoidal_coding(0, bs_pvc_mode);		
}		
}		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 45 — Syntax of sbr_channel_pair_element()

Syntax	No. of bits	Mnemonic
sbr_channel_pair_element(bs_amp_res, indepFlag)		
{		
if (**bs_coupling**) {	**1**	**uimsbf**
if (harmonicSBR) {		
if (**sbrPatchingMode[0,1]** == 0) {	**1**	**uimsbf**
sbrOversamplingFlag[0,1];	**1**	**uimsbf**
if (**sbrPitchInBinsFlag[0,1]**)	**1**	**uimsbf**
sbrPitchInBins[0,1];	**7**	**uimsbf**
else		
sbrPitchInBins[0,1] = 0;		
} else {		
sbrOversamplingFlag[0,1] = 0;		
sbrPitchInBins[0,1] = 0;		
}		
}		
sbr_grid(0, 0);		
sbr_dtdf(0, 0, indepFlag);		
sbr_dtdf(1, 0, indepFlag);		
sbr_invf(0);		
sbr_envelope(0,1, bs_amp_res);		
sbr_noise(0,1);		
sbr_envelope(1,1, bs_amp_res);		
sbr_noise(1,1);		
} else {		
if (harmonicSBR) {		
if (**sbrPatchingMode[0]** == 0) {	**1**	**uimsbf**
sbrOversamplingFlag[0];	**1**	**uimsbf**
if (**sbrPitchInBinsFlag[0]**)	**1**	**uimsbf**
sbrPitchInBins[0];	**7**	**uimsbf**
else		
sbrPitchInBins[0] = 0;		
} else {		
sbrOversamplingFlag[0] = 0;		
sbrPitchInBins[0] = 0;		
}		
if (**sbrPatchingMode[1]** == 0) {	**1**	**uimsbf**
sbrOversamplingFlag[1];	**1**	**uimsbf**
if (**sbrPitchInBinsFlag[1]**)	**1**	**uimsbf**
sbrPitchInBins[1];	**7**	**uimsbf**
else		
sbrPitchInBins[1] = 0;		
} else {		
sbrOversamplingFlag[1] = 0;		
sbrPitchInBins[1] = 0;		
}		
}		
sbr_grid(0, 0);		
sbr_grid(1, 0);		
sbr_dtdf(0,0, indepFlag);		
sbr_dtdf(1,0, indepFlag);		
sbr_invf(0);		
sbr_invf(1);		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

sbr_envelope(0,0, bs_amp_res);		
sbr_envelope(1,0, bs_amp_res);		
sbr_noise(0,0);		
sbr_noise(1,0);		
}		
if (**bs_add_harmonic_flag**[0]) {	**1**	**uimsbf**
sbr_sinusoidal_coding(0, 0);		
}		
if (**bs_add_harmonic_flag**[1]) {	**1**	**uimsbf**
sbr_sinusoidal_coding(1, 0);		
}		
}		

Table 46 — Syntax of sbr_grid()

Syntax	No. of bits	Mnemonic
sbr_grid(ch, bs_pvc_mode)		
{		
if (bs_pvc_mode == 0) {		
switch (**bs_frame_class**) {	**2**	**uimsbf**
case FIXFIX		
bs_num_env[ch] = 2^ **tmp**;	**2**	**uimsbf**, NOTE 1
if (bs_num_env[ch] == 1)		
bs_amp_res = 0;		
bs_freq_res[ch][0];	**1**	**uimsbf**
for (env = 1; env < bs_num_env[ch]; env++)		
bs_freq_res[ch][env] = bs_freq_res[ch][0];		
break;		
case FIXVAR		
bs_var_bord_1[ch];	**2**	**uimsbf**
bs_num_env[ch] = **bs_num_rel_1**[ch] + 1;	**2**	**uimsbf**
for (rel = 0; rel < bs_num_env[ch]-1; rel++)		
bs_rel_bord_1[ch][rel] = 2* **tmp** + 2;	**2**	**uimsbf**
ptr_bits = ceil (log (bs_num_env[ch] + 1) / log (2));		NOTE 2
bs_pointer[ch];	ptr_bits	**uimsbf**
for (env = 0; env < bs_num_env[ch]; env++)		
bs_freq_res[ch][bs_num_env[ch] – 1 – env];	**1**	**uimsbf**
break;		
case VARFIX		
bs_var_bord_0[ch];	**2**	**uimsbf**
bs_num_env[ch] = **bs_num_rel_0**[ch] + 1;	**2**	**uimsbf**
for (rel = 0; rel < bs_num_env[ch]-1; rel++)		
bs_rel_bord_0[ch][rel] = 2* **tmp** + 2;	**2**	**uimsbf**
ptr_bits = ceil (log (bs_num_env[ch] + 1) / log (2));		NOTE 2
bs_pointer[ch];	ptr_bits	**uimsbf**
for (env = 0; env < bs_num_env[ch]; env++)		
bs_freq_res[ch] [env];	**1**	**uimsbf**
break;		
case VARVAR		
bs_var_bord_0[ch];	**2**	**uimsbf**
bs_var_bord_1[ch];	**2**	**uimsbf**

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

```
                bs_num_rel_0[ch];                                          2          uimsbf
                bs_num_rel_1[ch];                                          2          uimsbf
                bs_num_env[ch] = bs_num_rel_0[ch] + bs_num_rel_1[ch] + 1;             NOTE 1
                for (rel = 0; rel < bs_num_rel_0[ch]; rel++)
                    bs_rel_bord_0[ch][rel] = 2* tmp + 2;                   2          uimsbf
                for (rel = 0; rel < bs_num_rel_1[ch]; rel++)
                    bs_rel_bord_1[ch][rel] = 2* tmp + 2;                   2          uimsbf
                ptr_bits = ceil (log(bs_num_env[ch] + 1) / log (2));                  NOTE 2
                bs_pointer[ch];                                            ptr_bits   uimsbf
                for (env = 0; env < bs_num_env[ch]; env++)
                    bs_freq_res[ch][env];                                  1          uimsbf
                break;
            }

            if (bs_num_env[ch] > 1)      { bs_num_noise[ch] = 2; }
            else                         { bs_num_noise[ch] = 1; }

        } else {
            bs_noise_position[ch];                                         4          uimsbf
            bs_var_len_hf[ch];                                             1,3        uimsbf
            if (bs_noise_position[ch] == 0) {
                bs_num_env[ch] = 1;
                bs_num_noise[ch] = 1;
                bs_freq_res[ch][0] = 0;
            } else {
                bs_num_env[ch] = 2;
                bs_num_noise[ch] = 2;
                for (env = 0; env < bs_num_env[ch]; env++) {
                    bs_freq_res[ch][env] = 0;
                }
            }
        }
}
```

NOTE 1: bs_num_env is restricted according to 7.5.1.3
NOTE 2: the division (/) is a float division without rounding or truncation.

Table 47 — Syntax of sbr_envelope()

Syntax	No. of bits	Mnemonic
sbr_envelope(ch, bs_coupling, bs_amp_res)		
{		
if (bs_coupling) {		
if (ch) {		
if (bs_amp_res) {		
t_huff = t_huffman_env_bal_3_0dB;		
f_huff = f_huffman_env_bal_3_0dB;		
} else {		
t_huff = t_huffman_env_bal_1_5dB;		
f_huff = f_huffman_env_bal_1_5dB;		
}		
} else {		
if (bs_amp_res) {		
t_huff = t_huffman_env_3_0dB;		
f_huff = f_huffman_env_3_0dB;		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

```
                } else {
                    t_huff = t_huffman_env_1_5dB;
                    f_huff = f_huffman_env_1_5dB;
                }
            }
        } else {
            if (bs_amp_res) {
                t_huff = t_huffman_env_3_0dB;
                f_huff = f_huffman_env_3_0dB;
            } else {
                t_huff = t_huffman_env_1_5dB;
                f_huff = f_huffman_env_1_5dB;
            }
        }

        for (env = 0; env < bs_num_env[ch]; env++) {
            if (bs_df_env[ch][env] == 0) {
                if (bs_coupling && ch) {
                    if (bs_amp_res)
                        bs_data_env[ch][env][0] = bs_env_start_value_balance;      5      uimsbf
                    else
                        bs_data_env[ch][env][0] = bs_env_start_value_balance;      6      uimsbf
                } else {
                    if (bs_amp_res)
                        bs_data_env[ch][env][0] = bs_env_start_value_level;        6      uimsbf
                    else
                        bs_data_env[ch][env][0] = bs_env_start_value_level;        7      uimsbf
                }
                for (band = 1;  band < num_env_bands[bs_freq_res[ch][env]]; band++)          NOTE 1
                    bs_data_env[ch][env][band] = sbr_huff_dec(f_huff, bs_codeword);  1..18   NOTE 2
            } else {
                for (band = 0; band < num_env_bands[bs_freq_res[ch][env]]; band++)           NOTE 1
                    bs_data_env[ch][env][band] = sbr_huff_dec(t_huff, bs_codeword);  1..18   NOTE 2
            }
            if (bs_interTes) {
                bs_temp_shape[ch][env];                                            1      uimsbf
                If (bs_temp_shape[ch][env]) {
                    bs_inter_temp_shape_mode[ch][env];                             2      uimsbf
                }
            }
        }
}
```

NOTE 1: num_env_bands[bs_freq_res[ch][env]] is derived from the header according to ISO/IEC 14496-3:2009, 4.6.18.3 and is named **n**.
NOTE 2: sbr_huff_dec() is defined in ISO/IEC 14496-3:2009, 4.A.6.1.

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 48 — Syntax of sbr_dtdf()

Syntax	No. of bits	Mnemonic
sbr_dtdf (ch, bs_pvc_mode, indepFlag)		
{		
if (bs_pvc_mode == 0) {		
if (indepFlag) {		
bs_df_env[ch][0] = 0		
} else {		
bs_df_env[ch][0];	**1**	
}		
for (env = 1; env < bs_num_env[ch]; env++) {		
bs_df_env[ch][env];	**1**	
}		
}		
if (indepFlag) {		
bs_df_noise[ch][0] = 0		
} else {		
bs_df_noise[ch][0];	**1**	
}		
for (noise = 1; noise < bs_num_noise[ch]; noise++) {		
bs_df_noise[ch][noise];	**1**	
}		
}		

Table 49 — Syntax of sbr_sinusoidal_coding()

Syntax	No. of bits	Mnemonic
sbr_sinusoidal_coding(ch, bs_pvc_mode)		
{		
for (n = 0; n < num_high_res[ch]; n++)		
bs_add_harmonic [ch][n];	**1**	
if (bs_pvc_mode != 0) {		
bs_sinusoidal_position = 31;		
bs_sinusoidal_position_flag;	**1**	
if (bs_sinusoidal_position_flag == 1)		
bs_sinusoidal_position;	**5**	**uimsbf**
}		
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 50 — Syntax of pvc_envelope

Syntax	No. of bits	Mnemonic
pvc_envelope(indepFlag)		
{		
divMode;	**3**	**uimsbf**
nsMode;	**1**	**uimsbf**
if (divMode<=3) {		
num_length = divMode;		
if (indepFlag) {		
reuse_pvcID = 0;		
} else {		
reuse_pvcID;	**1**	**uimsbf**
}		
if (reuse_pvcID) {		
pvcID[0]=pvcID[-1];		
} else {		
pvcID[0];	**7**	**uimsbf**
}		
k=1;		
if (num_length) {		
sum_length=0;		
for (i=0; i<num_length; i++) {		
if (sum_length >= 13) {		
length;	**1**	**uimsbf**
} else if (sum_length >= 11) {		
length;	**2**	**uimsbf**
} else if (sum_length >= 7) {		
length;	**3**	**uimsbf**
} else {		
length;	**4**	**uimsbf**
}		
length += 1;		
sum_length += length;		
for (j=1; j<length; j++, k++) {		
pvcID[k] = pvcID[k-1];		
}		
pvcID[k++];	**7**	**uimsbf**
}		
}		
for (; k<16; k++) {		
pvcID[k]=pvcID[k-1];		
}		
} else {		
switch (divMode) {		
case 4:		
num_grid_info=2;		
fixed_length=8;		
break;		
case 5:		
num_grid_info=4;		
fixed_length=4;		
break;		
case 6:		
num_grid_info=8;		
fixed_length=2;		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

break;		
case 7:		
num_grid_info=16;		
fixed_length=1;		
break;		
}		
for (i=0, k=0; i<num_grid_info; i++) {		
if (indepFlag && i==0) {		
grid_info = 1;		
} else {		
grid_info;	**1**	**uimsbf**
}		
if (grid_info) {		
pvcID[k++];	**7**	**uimsbf**
} else {		
pvcID[k++] = pvcID[k-1];		
}		
for (j=1; j<fixed_length; j++, k++) {		
pvcID[k] = pvcID[k-1];		
}		
}		
}		
}		

Table 51 — References to SBR syntactic elements

Syntax of	Please see
sbr_invf()	ISO/IEC 14496-3:2009, 4.4.2.8, Table 4.71
sbr_noise()	ISO/IEC 14496-3:2009, 4.4.2.8, Table 4.73

5.3.4 Payloads for MPEG Surround

Table 52 — Syntax of Mps212Data()

Syntax	No. of bits	Mnemonic
Mps212Data(indepFlag)		
{		
FramingInfo();		
if (indepFlag) {		
bsIndependencyFlag = 1;		
} else {		
bsIndependencyFlag;	**1**	**uimsbf**
}		
OttData();		
SmgData();		
TempShapeData();		
if (bsTsdEnable == 1) {		
TsdData();		
}		
}		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 53 — Syntax of FramingInfo()

Syntax	No. of bits	Mnemonic
FramingInfo()		
{		
if (bsHighRateMode) {		
bsFramingType;	**1**	**uimsbf**
bsNumParamSets;	**3**	**uimsbf**
} else {		
bsFramingType = 0;		
bsNumParamSets = 1;		
}		
numParamSets = bsNumParamSets + 1;		
nBitsParamSlot = ceil(log2(numSlots));		
if (bsFramingType) {		
for (ps=0; ps<numParamSets; ps++) {		
bsParamSlot[ps];	**nBitsParamSlot**	**uimsbf**
}		
}		
}		

Table 54 — Syntax of OttData()

Syntax	No. of bits	Mnemonic
OttData()		
{		
EcData(CLD, 0, 0, numBands);		NOTE 1
EcData(ICC, 0, 0, numBands);		NOTE 1
if (bsPhaseCoding) {		
bsPhaseMode;	**1**	**uimsbf**
if (bsPhaseMode) {		
bsOPDSmoothingMode;	**1**	**uimsbf**
EcData(IPD, 0, 0, bsOttBandsPhase);		
}		
}		
}		
NOTE 1: numBands is defined in ISO/IEC 23003-1:2007, 5.2, Table 39 and depends on bsFreqRes.		

© ISO/IEC 2012 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012

Table 55 — Syntax of SmgData()

Syntax	No. of bits	Mnemonic
SmgData()		
{		
if (bsHighRateMode) {		
for (ps=0; ps<numParamSets; ps++) {		NOTE 1
bsSmoothMode[ps**]**;	**2**	**uimsbf**
if (bsSmoothMode[ps] >= 2) {		
bsSmoothTime[ps**]**;	**2**	**uimsbf**
}		
if (bsSmoothMode[ps] == 3) {		
bsFreqResStrideSmg[ps**]**;	**2**	**uimsbf**
dataBands = (numBands-1)/pbStride+1;		NOTE 2
for (pg=0; pg<dataBands; pg++) {		
bsSmgData[ps][pg];	**1**	**uimsbf**
}		
}		
}		
} else {		
for (ps=0; ps<numParamSets; ps++) {		
bsSmoothMode[ps] = 0;		
}		
}		
}		
NOTE 1: numParamSets is defined by numParamSets = bsNumParamSets + 1.		
NOTE 2: pbStride is defined in ISO/IEC 23003-1:2007, 5.2, Table 70 and depends on bsFreqResStrideSmg. The division shall be interpreted as ANSI C integer division. numBands is defined in ISO/IEC 23003-1:2007, 5.2, Table 39 and depends on bsFreqRes.		

Table 56 — Syntax of TempShapeData()

Syntax	No. of bits	Mnemonic
TempShapeData()		
{		
bsTsdEnable = 0;		
if (bsTempShapeConfig == 3) {		
bsTsdEnable;	**1**	**uimsbf**
} else if ((bsTempShapeConfig == 1) || (bsTempShapeConfig == 2)) {		
bsTempShapeEnable;	**1**	**uimsbf**
if (bsTempShapeEnable) {		
for (ch=0; ch< numTempShapeChan; ch++) {		NOTE 1
bsTempShapeEnableChannel[ch];	**1**	**uimsbf**
}		
if (bsTempShapeConfig == 2) {		
EnvelopeReshapeHuff(bsTempShapeEnableChannel);		
}		
}		
}		
}		
NOTE 1: numTempShapeChan is 2 as defined 6.2.13.2.		

© ISO/IEC 2012 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23003-3:2012