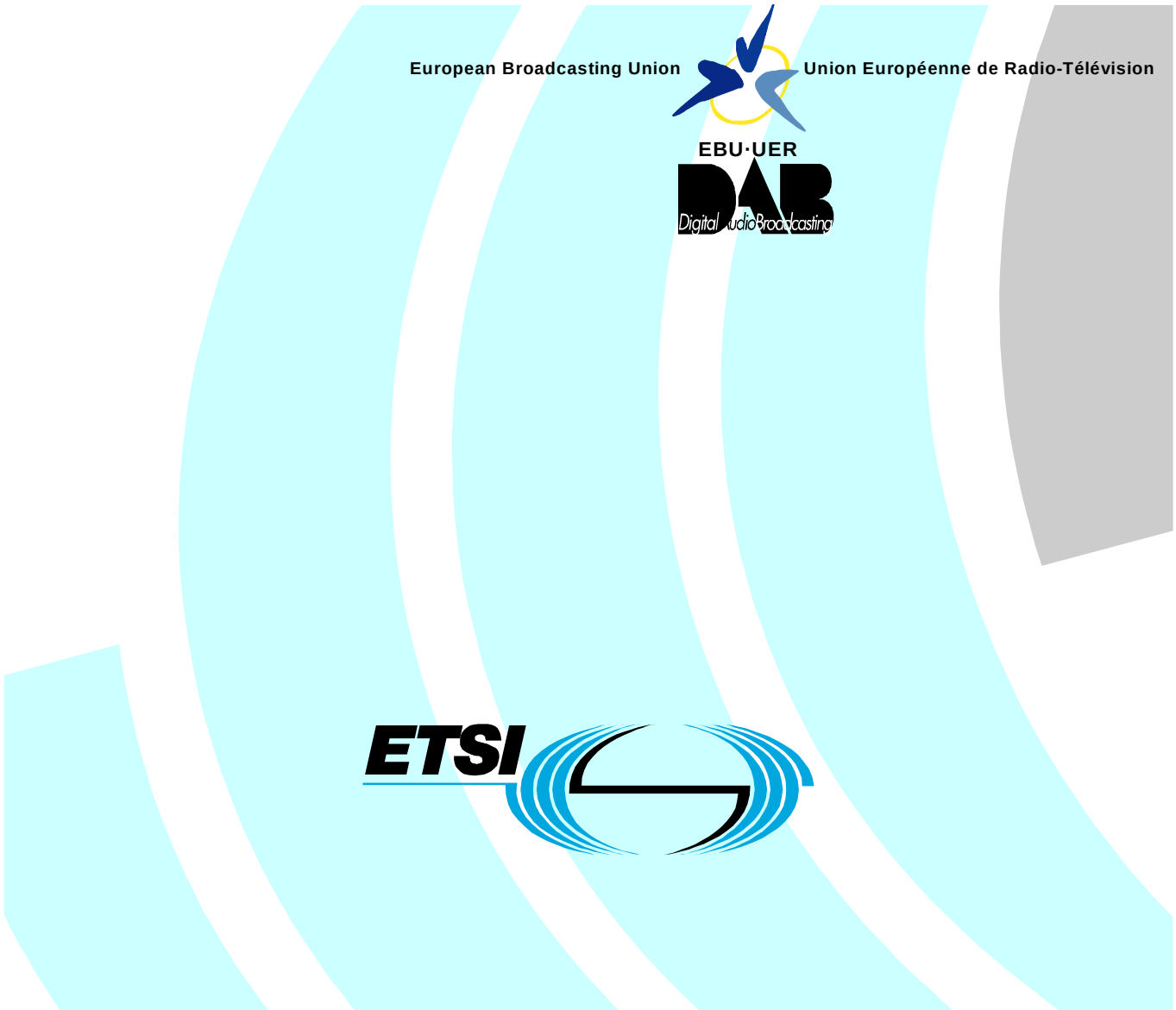
ETSI TS 102 563 V1.1.1 (2007-02)

Technical Specification

Digital Audio Broadcasting (DAB); Transport of Advanced Audio Coding (AAC) audio



Reference
DTS/JTC-DAB-49

Keywords
audio, broadcasting, coding, DAB, digital

ETSI

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

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

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

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

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

Foreword

This Technical Specification (TS) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE 1: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

The Eureka Project 147 was established in 1987, with funding from the European Commission, to develop a system for the broadcasting of audio and data to fixed, portable or mobile receivers. Their work resulted in the publication of European Standard, EN 300 401 [1], for DAB (see note 2) which now has worldwide acceptance. The members of the Eureka Project 147 are drawn from broadcasting organizations and telecommunication providers together with companies from the professional and consumer electronics industry.

NOTE 2: DAB is a registered trademark owned by one of the Eureka Project 147 partners.

1 Scope

The present document defines the method to code and transmit audio services using the HE AAC v2 [2] audio coder for Eureka-147 Digital Audio Broadcasting (DAB) (EN 300 401 [1]) and details the necessary mandatory requirements for decoders. The permitted audio modes and the data protection and encapsulation are detailed. This audio coding scheme permits the full use of the PAD channel for carrying dynamic labels and user applications.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

[1] ETSI EN 300 401: "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers".

[2] ISO/IEC 14496-3: "Information technology - Coding of audio-visual objects - Part 3: Audio".

3 Definitions, abbreviations and arithmetic operators

3.1 Definitions

For the purposes of the present document, the terms and definitions given in EN 300 401 [1] and the following apply:

access unit: access unit contains the audio samples for 20 ms, 30 ms, 40 ms or 60 ms of audio depending on the sampling rate of the AAC core, respectively 48 kHz, 32 kHz, 24 kHz or 16 kHz

audio super frame: audio super frame contains a number of AUs which together contain the encoded audio for 120 ms

subchannel_index: subchannel_index is derived from the size of the sub-channel carrying the audio service and defines the number of Reed-Solomon code words in each audio super frame

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in EN 300 401 [1] and the following apply:

AAC	Advanced Audio Coding
AU	Access Unit
DAC	Digital Analogue Converter
DMB	Digital Multimedia Broadcasting
DVB	Digital Video Broadcasting
HE AAC	High Efficiency AAC
MPS	MPEG Surround
PS	Parametric Stereo

RS	Reed-Solomon
SBR	Spectral Band Replication

3.3 Arithmetic operators

+	addition
–	subtraction
×	multiplication
÷	division
m DIV p	denotes the quotient part of the division of m by p (m and p are positive integers)
m MOD p	denotes the remainder of the division of m by p (m and p are positive integers)

$\sum_{i=p}^{q} f(i)$ denotes the sum: $f(p) + f(p + 1) + f(p + 2) ... + f(q)$

$\prod_{i=p}^{q} f(i)$ denotes the product: $f(p) \times f(p + 1) \times f(p + 2) ... \times f(q)$

4 Introduction

The DAB system standard [1] defines the way that audio (programme) services are carried when using MPEG Layer II. The present document defines the way that audio (programme) services are carried when using MPEG 4 HE AAC v2.

For Layer II audio, two sampling rates are permitted, 48 kHz and 24 kHz. Each audio frame contains samples for 24 ms or 48 ms respectively and each contains the same number of bytes. The audio frames are carried in one or two respectively DAB logical frames. For AAC, two transforms are specified. For DAB, only the 960 transform is permitted with sampling rates of 48 kHz, 32 kHz, 24 kHz and 16 kHz. Each AU (audio frame) contains samples for 20 ms, 30 ms, 40 ms or 60 ms respectively. In order to provide a similar architectural model to Layer II audio, and simple synchronization, AUs are built into audio super frames of 120ms which are then carried in five DAB logical frames. In order to provide additional error control, Reed Solomon coding and virtual interleaving is applied. The overall scheme is shown in figure 1.

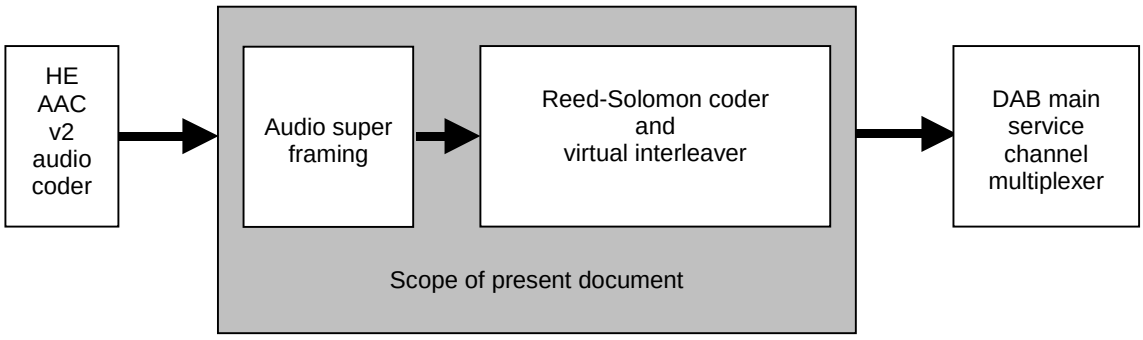


Figure 1: Conceptual diagram of the outer coder and interleaver

5 Audio

5.1 HE AAC v2 audio coding

For generic audio coding, a subset of the MPEG-4 High Efficiency Advanced Audio Coding v2 (HE AAC v2) profile chosen to best suit the DAB system environment is used. The HE AAC v2 Profile, Level 2 according to [2] shall apply with the following additional restrictions for the DAB system:

- Sampling rates: permitted output sampling rates of the HE AAC v2 decoder are 32 kHz and 48 kHz, i.e. when SBR is enabled the AAC core shall be operated at 16 kHz or 24 kHz, respectively. If SBR is disabled then the AAC core shall be operated at 32 kHz or 48 kHz respectively.
- Transform length: the number of samples per channel per AU is 960. This is required to harmonize HE AAC AU lengths to allow the combination of an integer number of AUs to build an audio super frame of 120 ms duration.
- Audio bit rates are restricted to fit within a maximum sub-channel size of 192 kbps (approximately 175 kbps for audio, assuming no PAD).
- Audio super framing: AUs are composed into audio super frames, which always correspond to 120 ms in time. The AUs in the audio super frames are encoded together such that each audio super frame is of constant length, i.e. that bit exchange between AUs is only possible within an audio super frame. The number of AUs per super frame are: two (16 kHz AAC core sampling rate with SBR enabled), three (24 kHz AAC core sampling rate with SBR enabled), four (32 kHz AAC core sampling rate) or six (48 kHz AAC core sampling rate).

Each audio super frame is carried in five consecutive logical DAB frames (see clause 7) which enables simple synchronization and management of reconfigurations. The size of the audio super frame is defined by the size of the MSC sub-channel (see [1] clause 6.2.1) which carries the audio super frame. Sub-channels are multiples of 8 kbps in size. The size of the audio super frame in bytes is given by the expressions below:

$$\text{subchannel\_index} = \text{MSC sub-channel size (kbps)} \div 8$$

$$\text{audio\_super\_frame\_size (bytes)} = \text{subchannel\_index} \times 110$$

The first byte of the audio super frame is byte 0 and the last byte is byte (audio_super_frame_size – 1).

NOTE: The subchannel_index parameter may take the values 1 to 24 due to the restriction limiting the maximum sub-channel size to 192 kbps.

5.2 Audio super framing syntax

Table 1: Syntax of he_aac_super_frame()

Syntax	No. of bits	Note

he_aac_super_frame(subchannel_index)
{
  he_aac_super_frame_header()
  for (n = 0; n < num_aus; n++) {
  au[n]
  au_crc[n]
  }
}
	
	8 × au_size[n]
	16
		
		determines num_aus

NOTE: au corresponds to one single access unit. Each au is protected by one CRC. The size of he_aac_super_frame() is equal to audio_super_frame_size.		

he_aac_super_frame_header()

The header contains the audio parameters for the audio super frame and the respective start positions of each AU within the audio super frame, along with an error protection word. The au_start values for the second and subsequent AUs are stored consecutively in the header. Depending on the number of AUs, 4 padding bits are added to achieve byte-alignment.

num_aus

The number of AUs in the audio super frame is determined by the settings of the audio parameters. num_aus may take the values 2, 3, 4 or 6 (see table 2).

au[n]

The AU contains the audio samples for 20 ms, 30 ms, 40 ms or 60 ms of audio depending on the core sampling rate, respectively 48 kHz, 32 kHz, 24 kHz or 16 kHz.

au_size[n]

This is the size in bytes of the AU.

au_crc[n]

Each AU is protected by a 16-bit CRC.

The CRC shall be generated according to the procedure defined in EN 300 401 [1] annex E. The generation shall be based on the polynomial:

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

The CRC word shall be complemented (1s complement) prior to transmission. At the beginning of each CRC word calculation, all register stages shall be initialized to "1".

Table 2: Syntax of he_aac_super_frame_header()

Syntax	No. of bits	Note
he_aac_super_frame_header()		
{		
header_firecode	**16**	
// start of audio parameters		
rfa	**1**	
dac_rate	**1**	
sbr_flag	**1**	
aac_channel_mode	**1**	
ps_flag	**1**	
mpeg_surround_config	**3**	
// end of audio parameters		
if ((dac_rate == 0) && (sbr_flag == 1)) num_aus = 2;		AAC core sampling rate 16 kHz
if ((dac_rate == 1) && (sbr_flag == 1)) num_aus = 3;		AAC core sampling rate 24 kHz
if ((dac_rate == 0) && (sbr_flag == 0)) num_aus = 4;		AAC core sampling rate 32 kHz
if ((dac_rate == 1) && (sbr_flag == 0)) num_aus = 6;		AAC core sampling rate 48 kHz
for (n = 1; n < num_aus; n++) {		
au_start[n];	**12**	AU start position
}		
if !((dac_rate == 1) && (sbr_flag == 1))		
alignment	**4**	byte-alignment
}		
NOTE: The au_start for the first AU in the audio super frame (au_start[0]) is not transmitted. The first AU always starts immediately after the he_aac_super_frame_header().		

header_firecode

The header_firecode is a 16-bit field containing a Fire code capable of detecting and correcting any single error burst of up to 6 bits. The Fire code shall be generated using the polynomial:

$$G(x) = (x^{11} + 1)(x^5 + x^3 + x^2 + x + 1) = x^{16} + x^{14} + x^{13} + x^{12} + x^{11} + x^5 + x^3 + x^2 + x + 1$$

The Fire code word shall be calculated over the nine bytes from byte 2 to byte 10 of the audio super frame.

NOTE 1: Except in the case where num_aus = 6, the Fire code calculation will include some bytes from the first AU.

At the beginning of each Fire code word calculation, all register stages shall be initialized to "0".

audio parameters

The audio parameters comprise the rfa, dac_rate, sbr_flag, aac_channel_mode, ps_flag and mpeg_surround_config fields.

NOTE 2: When the audio parameters are changed, some interruption to the audio output should be expected. Broadcasters should therefore plan audio parameter changes carefully.

rfa

The rfa is a 1-bit field reserved for future addition. This bit shall be set to zero for the currently specified application.

dac_rate

The dac_rate is a 1-bit field to signal the sampling rate of the DAC according to table 3:

Table 3: Definition of dac_rate

dac_rate	Meaning	Note
0	DAC sampling rate 32 kHz	
1	DAC sampling rate 48 kHz	

sbr_flag

The sbr_flag is a 1-bit field to signal the use of SBR according to table 4:

Table 4: Definition of sbr_flag

sbr_flag	Meaning	Note
0	SBR not used	The sampling rate of the AAC core is equal to the sampling rate of the DAC
1	SBR used	The sampling rate of the AAC core is half the sampling rate of the DAC

aac_channel_mode

The aac_channel_mode is a 1-bit field according to table 5:

Table 5: Definition of aac_channel_mode

aac_channel_mode	Meaning	Note
0	AAC (core) coding is mono	mono refers to a single_channel_element() see [2]
1	AAC (core) coding is stereo	stereo refers to a channel_pair_element() see [2]

ps_flag

The ps_flag is a 1-bit field to signal the use of PS according to table 6:

Table 6: Definition of ps_flag

ps_flag	Meaning	Note
0	PS not used	
1	PS used	only permitted when sbr_flag == 1 && aac_channel_mode == 0

mpeg_surround_config

The mpeg_surround_config is a 3-bit field according to table 7:

Table 7: Definition of mpeg_surround_config

mpeg_surround_config	Meaning	Note
000	MPEG Surround is not used	
001	MPEG Surround with 5.1 output channels is used	
010 to 111	reserved for future definition	

au_start[n]

The au_start is an unsigned integer, most significant bit first, carried in a 12-bit field that defines the start position within the audio super frame of the respective AU by giving the byte number of the first byte of the AU. The value of au_start for the first AU is not transmitted but is given by table 8:

Table 8: Definition of au_start for the first AU of the audio super frame

num_aus	value of au_start[0]	Note
2	5	
3	6	
4	8	
6	11	

The value of au_start for subsequent AUs is given by the expressions below:

- au_start[n] = au_start[n – 1] + au_size[n – 1] + 2;
- au_start[num_aus] = audio_super_frame_size.

The decoder can derive the value of au_size[n] from the received au_start[n] and au_start[n + 1].

alignment

This 4-bit field, when present, shall be set to 0 0 0 0.

5.3 MPEG Surround

5.3.1 Overview

MPEG Surround is standardized in MPEG-D, Part-1 (ISO/IEC 23003-1, (see bibliography)). It describes:

- Coding of multichannel signals based on a downmixed signal of the original multichannel signal, and associated spatial parameters. It offers lowest possible data rate for coding of multichannel signals, as well as an inherent mono or stereo downmix signal included in the data stream. Hence, a mono or stereo signal can be expanded to multi-channel by a very small additional data overhead.

- Binaural decoding of the MPEG Surround stream, enabling a surround sound experience over headphones.
- An Enhanced Matrix Mode that enables a multi-channel upmix from a stereo signal without any spatial parameters.

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

The MPEG Surround standard incorporates a number of tools that provide features which enable broad application of the standard. A key feature is the ability to scale the spatial image quality gradually from very low spatial overhead towards transparency. Another key feature is that the decoder input can be made compatible to existing matrix surround technologies.

5.3.2 Requirements for MPEG Surround encoders and decoders

The support of MPEG surround is optional for both audio encoder and decoder.

Interoperability between the different possible configurations for MPEG Surround is assured by MPEG profiles and levels. All subsequent statements towards profiles and levels refer to the definitions stated in ISO/IEC 23003-1 (see bibliography).

Table 9 defines the requirements for MPEG Surround encoders and decoders in order to support each defined configuration for MPEG Surround.

Table 9: Requirements for MPEG Surround encoders and decoders

mpeg_surround_config	Requirements for MPEG Surround encoder (if this configuration is supported by the encoder)	Requirements for MPEG Surround decoder (if this configuration is supported by the decoder)
0 0 0	None	None
0 0 1	The MPEG Surround data shall comply with level 2 or level 3 of the Baseline MPEG Surround profile. The spatial data shall be embedded in the downmix data stream, i.e. in the MPEG-4 AAC payload.	The decoder shall support either level 2 or level 3 of the Baseline MPEG Surround profile.
0 1 0 to 1 1 1	Reserved for future definition	Reserved for future definition

5.4 Programme Associated Data (PAD)

Each AU may include a number of bytes that carry Programme Associated Data (PAD). PAD is information that is synchronous to the audio and its contents may be intimately related to the audio. The PAD bytes in successive AUs constitute the PAD channel. The full range of functions provided by PAD carried in Layer II audio frames [1] are provided when using AAC AUs. The PAD is coded in exactly the same way as for Layer II audio (see [1] clause 7.4) using the F-PAD and X-PAD. The only exception is that the Dynamic Range Control mechanism provided by F-PAD is not used, see clause 5.4.2.

In each PAD field there may be two bytes called the fixed PAD (F-PAD). The F-PAD field is intended to carry control information with a strong real-time character and data with a very low bit rate. The PAD channel may be extended using an Extended PAD (X-PAD) field, intended to carry information providing additional functions to the listener, such as programme related text. The length of the X-PAD field is chosen by the service provider.

The inclusion of PAD is optional for the service provider.

The PAD bytes (if present) are always stored in a data_stream_element(), being a syntactic element of AAC [2] that can be multiplexed within a raw_data_block().

5.4.1 PAD insertion

Figure 2 shows the coding of the F-PAD and X-PAD fields within the PAD field.

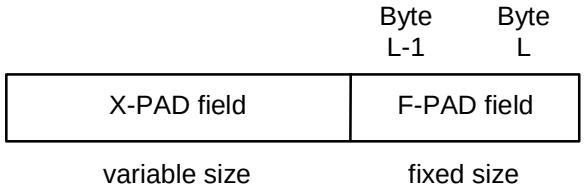


Figure 2: Coding of the PAD field

If no information is available for the X-PAD field and no information is sent in the F-PAD field, then the PAD field is empty and shall not be added to the AU.

If information is available for the X-PAD field but no information is sent in the F-PAD field, then both bytes of the F-PAD field shall be set to zero.

If no information is available for the X-PAD field but information is sent in the F-PAD field, then the PAD field only contains the F-PAD.

The PAD field carried in au[n] shall be associated with the audio carried in the following AU, au[n + 1].

All bytes of the PAD field have the same error protection. The maximum size of the X-PAD field is 196 bytes.

Table 10: Maximum bit rate of F-PAD and X-PAD data

AAC core sampling rate	Maximum bit rate for F-PAD data (2 bytes)	Maximum bit rate for X-PAD data (196 bytes)
16 kHz	267 bps	26 133 bps
24 kHz	400 bps	39 200 bps
32 kHz	533 bps	52 267 bps
48 kHz	800 bps	78 400 bps

The data_stream_element() carrying the PAD data shall be the first syntactic element of the raw_data_block().

The element_instance_tag of the data_stream_element() shall be the same as the element_instance_tag of the accompanied single_channel_element() or channel_pair_element(). Embedding a PAD field into a data_stream_element() causes an overhead of 2 bytes. There will be at most one data_stream_element() used for PAD insertion per raw_data_block().

The F-PAD channel carries a 2-bit field, "X-PAD Ind", which indicates one of three possibilities for the length of the X-PAD field:

- No X-PAD: only the F-PAD field is available. No X-PAD field is present. The length of the PAD field shall be two bytes.
- Short X-PAD: in this case the length of the X-PAD field in the current AU is four bytes. The length of the PAD field is 6 bytes.
- Variable size X-PAD: in this case the length of the X-PAD field may vary from AU to AU. The length of the X-PAD field in the current access unit can be derived by subtracting the length of the F-PAD field (2 bytes) from the length of the complete PAD field.

5.4.2 Coding of F-PAD and X-PAD

The coding of F-PAD and X-PAD shall be as specified in [1], clause 7.4 with the following exception:

In contrast to MPEG Audio Layer II, Dynamic Range Control (DRC) data shall not be carried in F-PAD. If dynamic range control is used, the DRC data shall be encoded utilizing AAC specific means: DRC data is stored in dynamic_range_info(), being contained in an extension_payload(), which in return is contained in a fill_element(), the latter being a syntactic element of AAC that can be multiplexed within a raw_data_block().

5.4.3 PAD extraction

PAD data (if present) is always located at the beginning of an AU; therefore if the AU starts with a data_stream_element(), then PAD data is available.

If no data_stream_element() is present at the beginning of the AU, or if the length of the data_stream_element() is less than two bytes (i.e. an invalid length for PAD data), then the PAD decoder shall react as if the F-PAD field had been received with both bytes set to zero and no X-PAD data is available.

The data_stream_element() explicitly indicates the length of the PAD information. The PAD decoder uses this length information to determine the size of the PAD field and also the size of the X-PAD field (if present). The size of the X-PAD field (if present) is two bytes less than the size of the PAD field.

Once the X-PAD field is extracted, decoding is the same as for X-PAD in MPEG audio layer II. The length of the X-PAD shall also be deduced from the contents information carried within the X-PAD field. If this length does not match the length of the X-PAD field derived from the length information within the data_stream_element(), then the decoder shall discard the X-PAD field.

6 Transport error coding and interleaving

Audio super frames are transported in five successive DAB logical frames with additional error protection. Reed-Solomon coding and byte-wise virtual interleaving are employed to increase the reliability of audio decoding in receivers.

The virtual interleaver can be considered as an array of dimensions *subchannel_index* rows by 120 columns. The subchannel_index is equal to the number of RS packets required to carry the audio super frame. The audio super frame data is fed into the table starting with the first row and first column and filling byte by byte from top to bottom and from left to right until the first 110 columns are completely filled. The RS parity bytes are then calculated across the rows, filling the final 10 columns. The table is transmitted, starting with the first row and first column and transmitting byte by byte from top to bottom and from left to right until all 120 columns have been sent.

In this way, the audio super frame data is transmitted in its original byte order, but the RS parity codewords are calculated from interleaved data. This increases the likelihood that any error bursts that overload the error protection scheme will only destroy one AU, rather than many.

6.1 RS coding

Reed-Solomon RS(120, 110, t = 5) shortened code (see note 1), derived from the original systematic RS(255, 245, t = 5) code, shall be applied to 110 byte portions of each audio super frame to generate an error protected packet (see figure 3).

NOTE 1: The Reed-Solomon code has length 120 bytes, dimension 110 bytes and allows the correction of up to 5 random erroneous bytes in a received word of 120 bytes.

Code generator polynomial: $G(x) = \prod_{i=0}^{9}(x + \alpha^i)$

Galois Field GF(2^8) with $\alpha^1 = 2$ using polynomial: $P(x) = x^8 + x^4 + x^3 + x^2 + 1$

NOTE 2: The Galois Field used is the same as is already used in the DMB and DVB specifications, thus allowing the same Galois Field arithmetic blocks to be used, however the code polynomial is shorter due to the fact that t = 5 in this implementation but t = 8 in the DMB/DVB specifications.

The shortened Reed-Solomon code may be implemented by adding 135 bytes, all set to zero, before the information bytes at the input of an RS(255, 245, t = 5) encoder. After the RS coding procedure these null bytes shall be discarded, leading to a RS code word of N = 120 bytes.

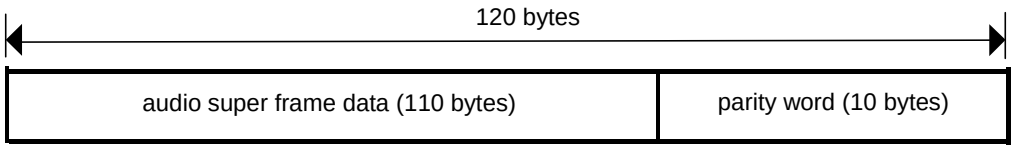


Figure 3: Error protected packet

6.2 Formation of the coding array

The elements of the coding array, $C_{[0..(s-1)],[0..109]}$, shall be filled by the audio super frame bytes, $A_{[0..(110 \times s)-1]}$. The position of bytes in the coding array, C, is defined by:

$$C_{i,j} = A_{i+j \times s}$$

where:

- $C_{i,j}$ shall be the elements of C with row index i [0..(s – 1)] and column index j [0..109];
- s shall be the subchannel_index.

6.3 Formation of the parity array

The elements of the parity array, $P_{i,[0..9]}$, shall be the 10 parity check bytes calculated for the information bytes $C_{i,[0..109]}$, where i is in the range 0 to (s – 1). The check bytes shall be calculated using the RS(120, 110, t = 5) code described in clause 6.1.

6.4 Formation of the output array

The elements of the output array, $O_{[0..(120 \times s)-1]}$, shall be filled by the audio super frame bytes, $A_{[0..(110 \times s)-1]}$ and the error protection bytes $P_{[0..(s-1)],[0..9]}$.

The position of bytes in the output array, O, is defined by:

$$O_k = \begin{cases} A_k & 0 \le k < 110 \times s \\ P_{q,r} & 110 \times s \le k < 120 \times s \end{cases}$$

where:

- A_k shall be the elements of A with index k [0..(110 × s) – 1];
- $P_{q,r}$ shall be the elements of P with row index $q = k$ MOD s and column index $r = (k$ DIV $s) - 110$;
- k shall be the index [0..(120 × s) – 1] of the output array.

6.5 Order of data transmission

Data shall be transmitted by reading out the elements of the output array $O_{[0..(120 \times s) - 1]}$. The bytes shall be read in sequence starting with O_0.

7 Signalling

7.1 FIC signalling

AAC audio services are signalled in the same way as Layer II audio services with the exception that the ASCTy carried in FIG 0/2 (see [1] clause 6.3.1) is set to the value 1 1 1 1 1 1. It is strongly recommended that only the EEP profiles are used and therefore the long form of FIG 0/1 should be used.

NOTE: The repetition rates and scheduling of FIGs as recommended in TR 101 496-2 (see bibliography) may need to be relaxed in order to accommodate the number of services carried in the ensemble.

7.2 Audio parameter signalling

Audio parameters are signalled in the audio super frame header, see clause 5.2.

8 Re-configuration

Re-configuration of the DAB multiplex may take place to allow new services to begin broadcasting, others to cease (either temporarily or permanently) or capacity to be exchanged between services (see [1] clause 6.5). Re-configuration of a sub-channel carrying an AAC audio service shall only take place at an audio super frame boundary. For a sub-channel that is reduced in capacity, the change will take place three audio super frames (15 logical frames) before the re-configuration instant. For a sub-channel that remains the same size (for example, by changing CU start address or exchanging bit-rate and protection level) or a sub-channel that is increased in capacity, the change will take place at the re-configuration instant.

NOTE 1: It is recommended that a multiplex containing AAC audio services is managed such that the audio super frames of all AAC audio services are aligned to ease the re-configuration process.

A re-configuration is only signalled when the sub-channel and/or service parameters of the MCI are changed. A change of the audio parameters signalled in the audio super frame header, which does not cause a change to the MCI, is not signalled as a re-configuration.

NOTE 2: When the audio parameters are changed, some interruption to the audio output should be expected. Broadcasters should therefore plan audio parameter changes carefully.

Annex A (normative): Error concealment

For the AAC core decoder and for the SBR and PS tools a description for concealment of erroneous bit streams is given. The error concealment provided by the DAB HE AAC v2 decoder shall provide at least the same level of performance as specified in this annex, but may be enhanced by specific implementations.

Concealment is applied when the transport layer indicates a distorted AU, i.e. the au_crc fails. If the he_aac_super_frame_header parameters cannot be correctly recovered, then the boundaries between AUs may be unavailable and one or more au_crcs may not be able to be determined. In this case concealment is applied to all affected AUs in the audio super frame. There are also various tests (plausibility checks) inside the AAC decoder and the SBR and PS tools. If such a check indicates an invalid bit stream, then concealment is applied to the according decoder stage, too.

A.1 AAC error concealment

The AAC core decoder includes a concealment function that increases the delay of the decoder by one AU.

Concealment works on the spectral data just before the final frequency to time conversion. In case a single AU is corrupted, concealment interpolates between the preceding and the following valid AUs to create the spectral data for the missing AU. If multiple AUs are corrupted, concealment implements first a fade out based on slightly modified spectral values from the last valid AU. If the decoder recovers from the error condition, the concealment algorithm performs a fade-in on valid spectral values. Fade in might be delayed (suppressed) to deal with error conditions, where only a valid AU here and there is perceived.

A.1.1 Interpolation of one corrupt AU

In the following, the actual AU is au[n], the corrupt AU to be interpolated is au[n – 1] and the last but one AU is au[n - 2]. au[n – 2] is the preceding valid AU for which spectral values have been stored during the processing in the previous call to the decoder.

The determination of window sequence and the window shape of the corrupt AU are described in table A.1.

Table A.1: Interpolated window sequences and window shapes

window sequence *n*–2	window sequence *n*	window sequence *n*–1	window shape *n*–1
ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	ONLY_LONG_SEQUENCE	0
ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	EIGHT_SHORT_SEQUENCE	LONG_START_SEQUENCE	1
EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE	1
EIGHT_SHORT SEQUENCE	ONLY_LONG_SEQUENCE or LONG_START_SEQUENCE or LONG_STOP_SEQUENCE	LONG_STOP_SEQUENCE	0

The scalefactor band energies of au[n – 2] and au[n] are calculated. If the window sequence in one of these AUs is an EIGHT_SHORT_SEQUENCE and the final window sequence for au[n – 1] is one of the long transform windows, the scalefactor band energies are calculated for long block scalefactor bands by mapping the frequency line index of short block spectral coefficients to a long block representation. The new interpolated spectrum is built on a per-scalefactorband basis by reusing the spectrum of the older AU, au[n – 2] and multiplying a factor to each spectral coefficient. An exception is made in the case of a short window sequence in au[n – 2] and a long window sequence in au[n]; here the spectrum of au[n] is modified by the interpolation factor. This factor is constant over the range of each scalefactor band and is derived from the scalefactor band energy differences of au[n – 2] and au[n]. Finally noise substitution is applied by flipping the sign of the interpolated spectral coefficients randomly.

A.1.2 Fade-out and fade-in

Fade-out and fade-in behaviour, i.e. the attenuation ramp, might be fixed or adjustable by the user. The spectral coefficients from the last AU are attenuated by a factor corresponding to the fade-out characteristics and then passed to the frequency-to-time mapping. Depending on the attenuation ramp, the concealment switches to muting after a number of consecutive invalid AUs, which means the complete spectrum will be set to 0.

After recovering from the error condition, the decoder fades in again depending on a ramp-up function possibly different from the ramp-down characteristics. If the concealment has switched to muting, fade-in might be suppressed for a configurable number of AUs to avoid annoying output of non-consecutive individual valid AUs.

A.2 SBR error concealment

The SBR error concealment algorithm is based on using previous envelope and noise-floor values with an applied decay, as a substitute for the corrupt data. In the flowchart of figure 1 the basic operation of the SBR error concealment algorithm is outlined. If the AU error flag is set, error concealment bitstream data is generated to be used instead of the corrupt bitstream data. The concealment data is generated according to the following:

The time frequency grids are set to:

- $L_E = 1$
- $\mathbf{t}_E(0) = \mathbf{t}'_E(L'_E) - numTimeSlots$
- $\mathbf{t}_E(1) = numTimeSlots$
- $\mathbf{r}(l) = HI \quad , 0 \le l < L_E$
- $bs\_pointer = 0$
- $L_Q = 1$
- $\mathbf{t}_Q = \left[\mathbf{t}_E(0), \mathbf{t}_E(1)\right]$

The delta coding direction for both the envelope data and noise-floor data are set to be in the time-direction. The envelope data is calculated according to:

$$\mathbf{E}_{Delta}(k,l) = \begin{cases} -step & , \mathbf{E}_{prev}(k,l) > target \\ step & , otherwise \end{cases} \quad , 0 \le k < \mathbf{n}\left(\mathbf{r}(l)\right), 0 \le l < L_E$$

where:

$$step = \begin{cases} 2 & , if \quad bs\_amp\_res = 1 \\ 1 & , otherwise \end{cases}$$

$$target = \begin{cases} \mathbf{panOffset}\left(bs\_amp\_res\right) & , if \quad bs\_coupling = 1 \\ 0 & , otherwise \end{cases}$$

And where *bs_amp_res* and *bs_coupling* are set to the values of the previous AU.

The noise floor data is calculated according to:

$$\mathbf{Q}_{Delta}(k,l) = 0 \quad , \begin{cases} 0 \le l < L_Q \\ 0 \le k < N_Q \end{cases}$$

Furthermore, the inverse-filtering levels in *bs_invf_mode* are set to the values of the previous AU, and all elements in *bs_add_harmonic* are set to zero.

If the AU error flag is not set, the present time grid and envelope data may need modification if the previous AU was corrupt. If the previous AU was corrupt, the time grid of the present AU is modified in order to make sure that there is a continuous transition between the AUs. The envelope data for the first envelope is modified according to:

$$\mathbf{E}_{mod}(k,0) = \mathbf{E}(k,0) + a \cdot \log_2\left(\frac{\mathbf{t}_E(1) - \mathbf{t}_E(0)}{\mathbf{t}_E(1) - estimated\_start\_pos}\right), \quad 0 \le k < \mathbf{F}\left(\mathbf{r}(l),0\right)$$

where:

$$estimated\_start\_pos = \mathbf{t}'_E\left(L'_E\right) - numberTimeSlots$$

After the delta coded data has been decoded, a plausibility check is performed to make sure that the decoded data is within reasonable limits. For the envelope data, the logarithmic values shall be within the limits:

$$\mathbf{E}(k,l) \le \begin{cases} 35 & ,ampRes = 0 \\ 70 & ,ampRes = 1 \end{cases}$$

Otherwise the AU will be considered corrupt.

The time grids are also verified according to the following rules (if any of the below is true the AU is considered to be corrupt):

- $L_E < 1$
- $L_E > 5$
- $L_Q > 2$
- $\mathbf{t}_E(0) < 0$
- $\mathbf{t}_E(0) \ge \mathbf{t}_E(L_E)$
- $\mathbf{t}_E(0) > 3$
- $\mathbf{t}_E(L_E) < 16$
- $\mathbf{t}_E(L_E) > 19$
- $\mathbf{t}_E(l) \ge \mathbf{t}_E(l+1), 0 \le l < L_E$
- $l_A > L_E$
- $L_E = 1\ AND\ L_Q > 1$
- $\mathbf{t}_Q(0) \ne \mathbf{t}_E(0)$
- $\mathbf{t}_Q(L_Q) \ne \mathbf{t}_E(L_E)$
- $\mathbf{t}_Q(l) \ge \mathbf{t}_Q(l+1), 0 \le l < L_Q$
- all elements of $\mathbf{t}_Q$ are not among the elements of $\mathbf{t}_E$

If the plausibility check fails, the AU error flag is set and the error concealment outlined above is applied.

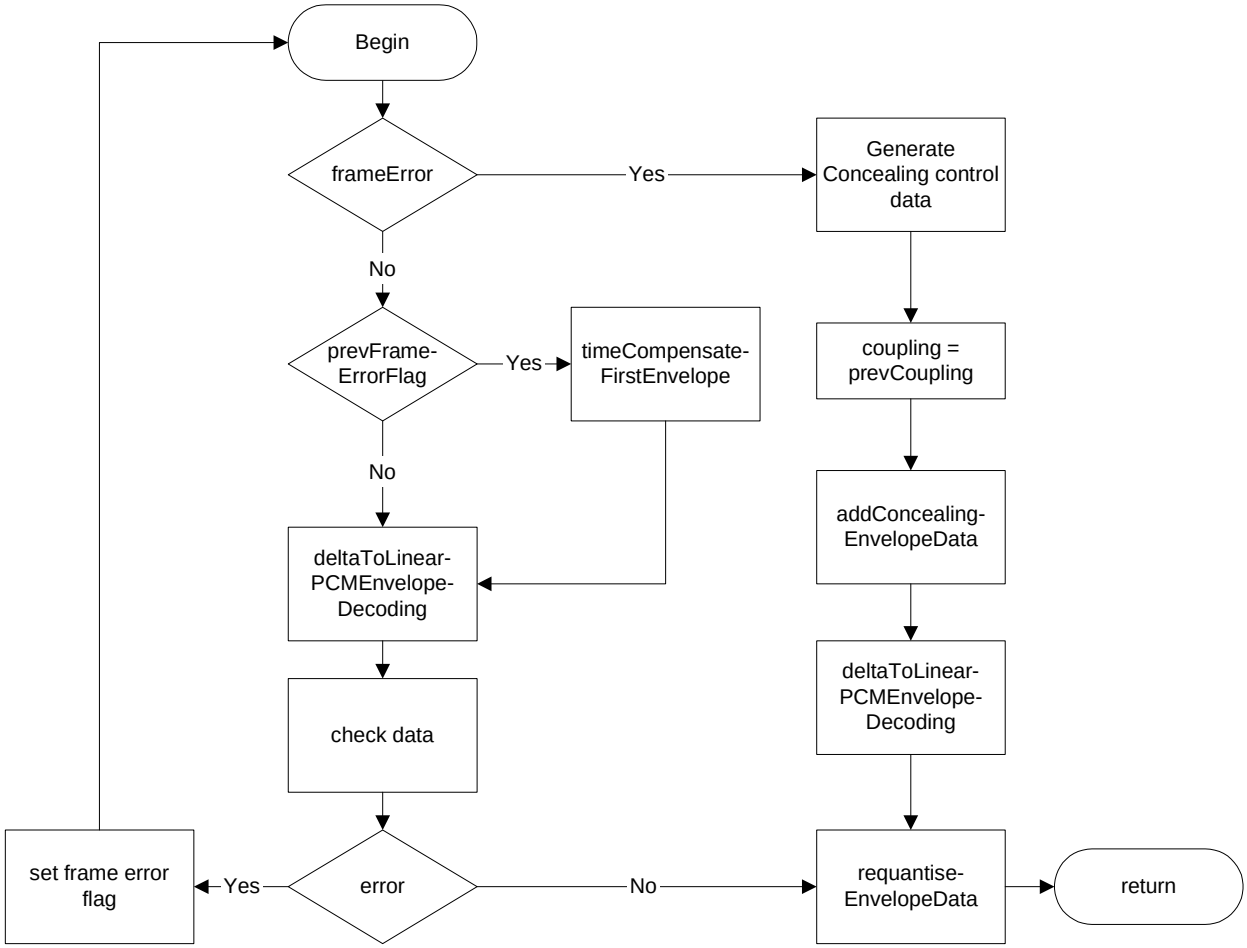


Figure A.1: SBR error concealment overview

A.3 Parametric stereo error concealment

Parametric stereo error concealment is based on the fact that the stereo image is quasi stationary. The concealment strategy keeps the Parametric Stereo settings from the last valid AU until a new set of Parametric Stereo settings can be decoded from a valid AU.

Annex B (informative): Implementation tips for PAD insertion

To reduce the data rate needed for PAD, the following optimizations on encoder side should be made:

- If no PAD is inserted (neither F-PAD, nor X-PAD information is available), then no data_stream_element() encapsulating the PAD data should be added to the access unit. A value of 0 for the F-PAD field is equivalent to "no F-PAD information is available".

For low audio bit rates, it is important to take into account that additional (X-)PAD data will reduce the bit rate for the audio and thus the audio quality. For low audio bit rates, even the bit rate for dynamic labels might become significant. It is not just the average (X-)PAD data rate that should be taken into account, but especially for low audio bit rates, also the "burstiness" of (X-)PAD data insertion should be taken into consideration.

If the exact timing for X-PAD insertion is not important (it is not important for most multimedia information such as dynamic labels) and low audio bit rates are used, then the following should be considered:

- The bit rate used for PAD depends on the total number of used bytes (including encapsulation and F-PAD) per audio super frame. It might sometimes be more efficient to use one single big X-PAD field in one audio super frame than to use multiple smaller X-PAD fields in the same audio super frame;

- Some multimedia data is broadcast in bursts (e.g., if one dynamic label with title/artist information is broadcast every 15 seconds). The impact on the audio quality is reduced if such a burst is spread over multiple audio super frames instead of just one (e.g., a label of 32 bytes could be broadcast within one audio super frame or spread over 4 audio super frames with only about one fourth of the X-PAD capacity used per audio super frame).

Annex C (informative): Synchronizing to the audio super frame structure

Before the receiver can decode an audio super frame, it must first concatenate five consecutive DAB frames to build the audio super frame. The receiver must therefore be able to determine which five consecutive DAB frames within a sub channel constitute an audio super frame.

Synchronization is needed when the receiver switches to an audio sub-channel, or when the receiver detects a loss of synchronization (i.e. too many consecutive audio super frames with uncorrectable errors in the audio super frame header).

This synchronization process must be fast and reliable. Reliability also implies that the receiver is able to synchronize to the audio super frame structure even in the presence of reception errors.

The synchronization comprises multiple steps:

1) At least 5 DAB frames are stored one after the other into the audio super frame buffer. This step takes d × 24 ms, where d is the number of DAB frames that are added to the audio super frame buffer. d is at least 5, so this step takes at least 120 ms.

2) A first test is used to check if the first DAB frame in the audio super frame buffer might be the beginning of an audio super frame. If that is not the case, go to step 4.

3) It is checked whether the first five DAB frames are a valid audio super frame (i.e., the first DAB frame contains a valid audio super frame header). The receiver does the RS decoding to correct all reception errors (if there were any) and then checks whether the header checksum is correct; for this test, the Fire code is used for error detection only. If that is the case, then the first valid audio super frame was received. Synchronization is completed.

4) If the first DAB frame of the audio super frame buffer is not a valid beginning of an audio super frame, then the first DAB frame is removed from the audio super frame buffer and the next received DAB frame is added to the end of the audio super frame buffer. DAB frames are received every 24 ms. We then restart at step 2.

Fast synchronization implies that the check for a valid audio super frame (step 2 and 3) can be made every 24 ms (i.e. for every received DAB frame it is possible to check whether it could be the beginning of an audio super frame). This is needed to ensure that the synchronization process is able to synchronize to the first occurrence of an audio super frame.

Depending on the receiver implementation, step 3 may be time consuming. If the time to check for a valid audio super frame (including the RS decoding) is longer than 24 ms (the time when the next DAB frame is received), then step 2 is needed to make a fast prefiltering. The time consuming step 3 is then only needed when the first DAB frame in the audio super frame buffer is very likely the beginning of an audio super frame.

Step 2 could use the Fire code in error detection mode in order to check for a valid audio super frame header even in the presence of errors in the super frame header. If the Fire code detects too many errors (i.e. more errors that it could correct), then the first DAB frame probably does not contain the beginning of an audio super frame.

NOTE 1: This check will detect all audio super frame headers with no errors or an error burst of up to 6 bits, however, it might sometimes wrongly assume to have found a valid super frame header. But still the use of the Fire code reduces the number of frames that the time consuming step 3 needs to test.

NOTE 2: This check will not detect an audio super frame header with more errors than the Fire code could correct. In such a case, the synchronization is delayed by one audio super frame (120 ms). This happens even if RS decoding (step 3) is able to correct the errors in the super frame header.

Step 2 could do the RS decoding progressively. It would use d = 5 for step 1. It would first decode just the RS codewords that protect (among other bytes) the first 11 bytes of the audio super frame, these 11 bytes are protected by the super frame header Fire code (Note: for bitrates below 96 kbps this already implies that all RS codewords are decoded, so this optimization step only makes sense for higher bit rates). In the next step the Fire code is used to check whether a valid audio super frame header is present at the beginning of the potential audio super frame. For this check, the Fire code is used for error detection only.

If a receiver has hardware RS decoding and step 3 is very fast (i.e. fast enough to do the RS decoding and checking for a valid audio super frame in less than 24 ms), then the receiver should use d = 5 for step 1 and check for a valid audio super frame every 24 ms (i.e. every time a DAB frame is received). Step 2 would not be used. This assures a fast and very reliable synchronization.

Annex D (informative): Processing a super frame

Once synchronization to the audio super frame structure is achieved (see annex C), the decoder will process super frame after super frame.

The following steps will be performed:

1) First the RS decoding of the super frame is performed. Usually it will be possible to correct all reception errors (if any).

2) Then the Fire code is used (error detection only) to check if there are errors in the super frame header. If there are errors, a single burst error of up to 6 bits can be corrected by the Fire code (error correction mode).

3) If the RS decoding indicates no errors or a correctable number of errors and the Fire code (error detection mode) detects no error then it is safe to extract the audio parameters from the current audio super frame. These audio parameters are then stored.

4) If either RS or Fire code (error detection mode) indicate uncorrectable errors, then the receiver should not extract the audio parameters from the current audio super frame, but use the audio parameters extracted from an audio super frame where the RS decoding indicated no error or a correctable number of errors and the Fire code (error detection mode) indicated no error (i.e. the receiver assumes that the audio parameters are unchanged).

5) The audio parameters are evaluated to determine the number of AUs per audio super frame and the configuration of the audio decoder and the DAC.

6) If the Fire code (error detection mode) detected an error in the super frame header, then the error correction capability of the Fire code may be used to try correcting an error burst of up to 6 bits in the super frame header. At this stage, additional information from the RS decoding process may be evaluated (e.g. if one RS codeword contained uncorrectable errors, but all other RS codewords could be corrected, then this information may be used by the Fire code decoder).

The decoder then iterates over the au_start parameters. The first AU starts immediately after the super frame header; the last AU ends at the super frame end.

For all au_start values, the decoder first does a sanity check. All au_start values must be between au_start[0] and audio_super_frame_size.

To extract au[n], both au_start[n] and au_start[n + 1] must pass the sanity check; au_start[n] must also be smaller than au_start[n + 1].

NOTE: A corrupted au_start value will cause the loss of two adjacent AUs.

From au_start[n] and au_start[n + 1] the decoder derives offset and length of the AU within the audio super frame. Then the au_crc is checked. If the CRC check detects an error, then the AU is concealed. If the CRC is correct, then the AU is decoded, see annex A.

If too many consecutive AUs must be concealed, then the decoder will assume a synchronization loss and restart the synchronization process, see annex C.

Annex E (informative): Bit-rate available for audio

The capacity available for audio (including PAD) is tabulated in table E.1 for reference.

Table E.1: Bit-rate available for audio (including PAD)

Subchannel index, s	Sub-channel size (kbps)	Bit-rate available for audio (bps)			
		AAC core sampling rate			
		16 kHz	24 kHz	32 kHz	48 kHz
1	8	6 733	6 533	6 267	5 800
2	16	14 067	13 867	13 600	13 133
3	24	21 400	21 200	20 933	20 467
4	32	28 733	28 533	28 267	27 800
5	40	36 067	35 867	35 600	35 133
6	48	43 400	43 200	42 933	42 467
7	56	50 733	50 533	50 267	49 800
8	64	58 067	57 867	57 600	57 133
9	72	65 400	65 200	64 933	64 467
10	80	72 733	72 533	72 267	71 800
11	88	80 067	79 867	79 600	79 133
12	96	87 400	87 200	86 933	86 467
13	104	94 733	94 533	94 267	93 800
14	112	102 067	101 867	101 600	101 133
15	120	109 400	109 200	108 933	108 467
16	128	116 733	116 533	116 267	115 800
17	136	124 067	123 867	123 600	123 133
18	144	131 400	131 200	130 933	130 467
19	152	138 733	138 533	138 267	137 800
20	160	146 067	145 867	145 600	145 133
21	168	153 400	153 200	152 933	152 467
22	176	160 733	160 533	160 267	159 800
23	184	168 067	167 867	167 600	167 133
24	192	175 400	175 200	174 933	174 467

Annex F (informative): Bibliography

ETSI TR 101 496-2: "Digital Audio Broadcasting (DAB); Guidelines and rules for implementation and operation; Part 2: System features".

ISO/IEC 23003-1: "Information technology - MPEG audio technologies - Part 1: MPEG Surround".

History

Document history		
V1.1.1	February 2007	Publication