ETSI ES 201 980 V1.2.2 (2003-04)

ETSI Standard

Digital Radio Mondiale (DRM);
System Specification

European Broadcasting Union

Union Européenne de Radio-Télévision

EBU·UER

Reference

RES/JTC-DRM-03

Keywords

broadcasting, digital, DRM, radio

Contents

Intellectual Property Rights

Foreword

This ETSI Standard (ES) 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: 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 present document is the revision of TS 101 980.

Introduction

The frequency bands used for broadcasting below 30 MHz are:

- Low Frequency (LF) band - from 148,5 kHz to 283,5 kHz, in ITU Region 1 [1] only;
- Medium Frequency (MF) band - from 526,5 kHz to 1 606,5 kHz, in ITU Regions 1 [1] and 3 [1] and from 525 kHz to 1 705 kHz in ITU Region 2 [1];
- High Frequency (HF) band - a set of individual broadcasting bands in the frequency range 2,3 MHz to 27 MHz, generally available on a Worldwide basis.

These bands offer unique propagation capabilities that permit the achievement of:

- large coverage areas, whose size and location may be dependent upon the time of day, season of the year or period in the (approximately) 11 year sunspot cycle;
- portable and mobile reception with relatively little impairment caused by the environment surrounding the receiver.

There is thus a desire to continue broadcasting in these bands, perhaps especially in the case of international broadcasting where the HF bands offer the only reception possibilities which do not also involve the use of local repeater stations.

However, broadcasting services in these bands:

- use analogue techniques;
- are subject to limited quality;
- are subject to considerable interference as a result of the long-distance propagation mechanisms which prevail in this part of the frequency spectrum and the large number of users.

As a direct result of the above considerations, there is a desire to effect a transfer to digital transmission and reception techniques in order to provide the increase in quality which is needed to retain listeners who, increasingly, have a wide variety of other programme reception media possibilities, usually already offering higher quality and reliability.

In order to meet the need for a digital transmission system suitable for use in all of the bands below 30 MHz, the Digital Radio Mondiale (DRM) consortium was formed in early 1998. The DRM consortium is a non-profit making body which seeks to develop and promote the use of the DRM system worldwide. Its members include broadcasters, network providers, receiver and transmitter manufacturers and research institutes. More information is available from their website (http://www.drm.org/).

1 Scope

The present document gives the specification for the Digital Radio Mondiale (DRM) system for digital transmissions in the broadcasting bands below 30 MHz.

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.

[1]	ITU-R Radio Regulations.
[2]	ISO/IEC 14496-3: "Information technology - Coding of audio-visual objects - Part 3: Audio".
[3]	ETSI EN 300 401: "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers".
[4]	EN 62106: "Specification of the radio data system (RDS) for VHF/FM sound broadcasting in the frequency range from 87,5 to 108,0 MHz".
[5]	ISO/IEC 10646-1: "Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane".
[6]	ISO 639-2: "Codes for the representation of names of languages - Part 2: Alpha-3 code".
[7]	ISO 3166 (all parts): "Codes for the representation of names of countries and their subdivisions".
[8]	ISO 8859-1: "Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1".
[9]	ITU-R Recommendation BS.559-2: "Objective measurement of radio-frequency protection ratios in LF, MF and HF broadcasting".
[10]	ITU-R Recommendation SM.328-10: "Spectra and bandwidth of emissions".
[11]	ETSI TS 101 968: "Digital Radio Mondiale (DRM); Data applications directory".

3 Definitions, symbols, abbreviations and convention

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

cell: sine wave portion of duration T_s, transmitted with a given amplitude and phase and corresponding to a carrier position

NOTE: Each OFDM symbol is the sum of K such sine wave portions equally spaced in frequency.

energy dispersal: operation involving deterministic selective complementing of bits in the logical frame, intended to reduce the possibility that systematic patterns result in unwanted regularity in the transmitted signal

Fast Access Channel (FAC): channel of the multiplex data stream which contains the information that is necessary to find services and begin to decode the multiplex

Main Service Channel (MSC): channel of the multiplex data stream which occupies the major part of the transmission frame and which carries all the digital audio services, together with possible supporting and additional data services

mod: the modulo operator

NOTE: (x mod y) = z, where y > 0, such that x = qy + z, q is an integer, and $0 \le z < y$.

OFDM symbol: transmitted signal for that portion of time when the modulating amplitude and phase state is held constant on each of the equally-spaced carriers in the signal

reserved for future addition (rfa): bits with this designation shall be set to zero

NOTE: Receivers shall ignore these bits.

reserved for future use (rfu): bits with this designation shall be set to zero

NOTE: Receivers shall check that these bits are zero in order to determine the valid status of the other fields in the same scope.

Service Description Channel (SDC): channel of the multiplex data stream which gives information to decode the services included in the multiplex

NOTE: The SDC also provides additional information to enable a receiver to find alternative sources of the same data.

Single Frequency Network (SFN): network of transmitters sharing the same radio frequency to achieve a large area coverage

transmission frame: a number of consecutive OFDM symbols (duration of 400 ms), whereby the first OFDM symbol contains the time reference cells

transmission super frame: three consecutive transmission frames (duration of 1 200 ms), whereby the first OFDM symbols contain the SDC block

logical frame: contains data of one stream during 400 ms

multiplex frame: logical frames from all streams form a multiplex frame (duration of 400 ms)

NOTE: It is the relevant basis for coding and interleaving.

3.2 Symbols

For the purposes of the present document, the following symbols apply:

$E[\ ]$	expectation value of the expression in brackets
f_c	reference frequency of the emitted signal
K	number of active carriers in the OFDM symbol
$K_{\max}$	carrier index of the upper active carrier in the OFDM signal
$K_{\min}$	carrier index of the lower active carrier in the OFDM signal
L_{MUX}	number of input bits per multiplex frame for the multilevel encoding
N_{MUX}	number of MSC cells (QAM symbols) per multiplex frame
T	elementary time period, equal to $83^{1/3}$ μs (1/12 kHz)
T_f	duration of the transmission frame, equal to 400 ms
T_g	duration of the guard interval
T_s	duration of an OFDM symbol
T_{sf}	duration of the transmission super-frame built from three transmission frames

T_u	duration of the useful (orthogonal) part of an OFDM symbol, excluding the guard interval
X^*	complex conjugate of value X
$\lceil \ \rceil$	round towards plus infinity
$\lfloor \ \rfloor$	round towards minus infinity

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AAC	Advanced Audio Coding
AFS	Alternative Frequency Switching
BER	Bit Error Rate
CELP	Code Excited Linear Prediction
CRC	Cyclic Redundancy Check
DFT	Discrete Fourier Transform
EEP	Equal Error Protection
FAC	Fast Access Channel
HF	High Frequency
HVXC	Harmonic Vector eXcitation Coding
IFFT	Inverse Fast Fourier Transform
ISO	International Organization for Standardization
LF	Low Frequency
LPC	Linear Predictive Coding
LSb	Least Significant bit
LSP	Line Spectral Pairs
MF	Medium Frequency
MPEG	Moving Picture Experts Group
MSb	Most Significant bit
MSC	Main Service Channel
OFDM	Orthogonal Frequency Division Multiplexing
PRBS	Pseudo-Random Binary Sequence
QAM	Quadrature Amplitude Modulation
RF	Radio Frequency
rfa	reserved for future addition
rfu	reserved for future use
SBR	Spectral Band Replication
SDC	Service Description Channel
SFN	Single Frequency Network
SM	Standard Mapping
SPP	Standard Protected Part
UEP	Unequal Error Protection
uimsbf	unsigned integer most significant bit first
VSPP	Very Strongly Protected Part

3.4 Convention

Unless otherwise stated, the following convention, regarding the order of bits within each step of processing is used:

- in figures, the bit shown in the left hand position is considered to be first;
- in tables, the bit shown in the left hand position is considered to be first;
- in numerical fields, the Most Significant bit (MSb) is considered to be first and denoted by the higher number. For example, the MSb of a single byte is denoted "b7" and the Least Significant bit (LSb) is denoted "b0";
- in vectors (mathematical expressions), the bit with the lowest index is considered to be first.

4 General characteristics

4.1 System overview

The DRM system is designed to be used at any frequency below 30 MHz, i.e. within the long wave, medium wave and short wave broadcasting bands, with variable channelization constraints and propagation conditions throughout these bands. In order to satisfy these operating constraints, different transmission modes are available. A transmission mode is defined by transmission parameters classified in two types:

- signal bandwidth related parameters;
- transmission efficiency related parameters.

The first type of parameters defines the total amount of frequency bandwidth for one transmission. Efficiency related parameters allow a trade-off between capacity (useful bit rate) and ruggedness to noise, multipath and Doppler.

4.2 System architecture

This clause gives a general presentation of the system architecture, based on the synoptic diagram of figure 1, which gives reference to the clauses defining the individual parts of the system.

Figure 1 describes the general flow of different classes of information (audio, data, etc.) and does not differentiate between different services that may be conveyed within one or more classes of information. A detailed description on the distribution of services onto those classes can be found in clause 6.

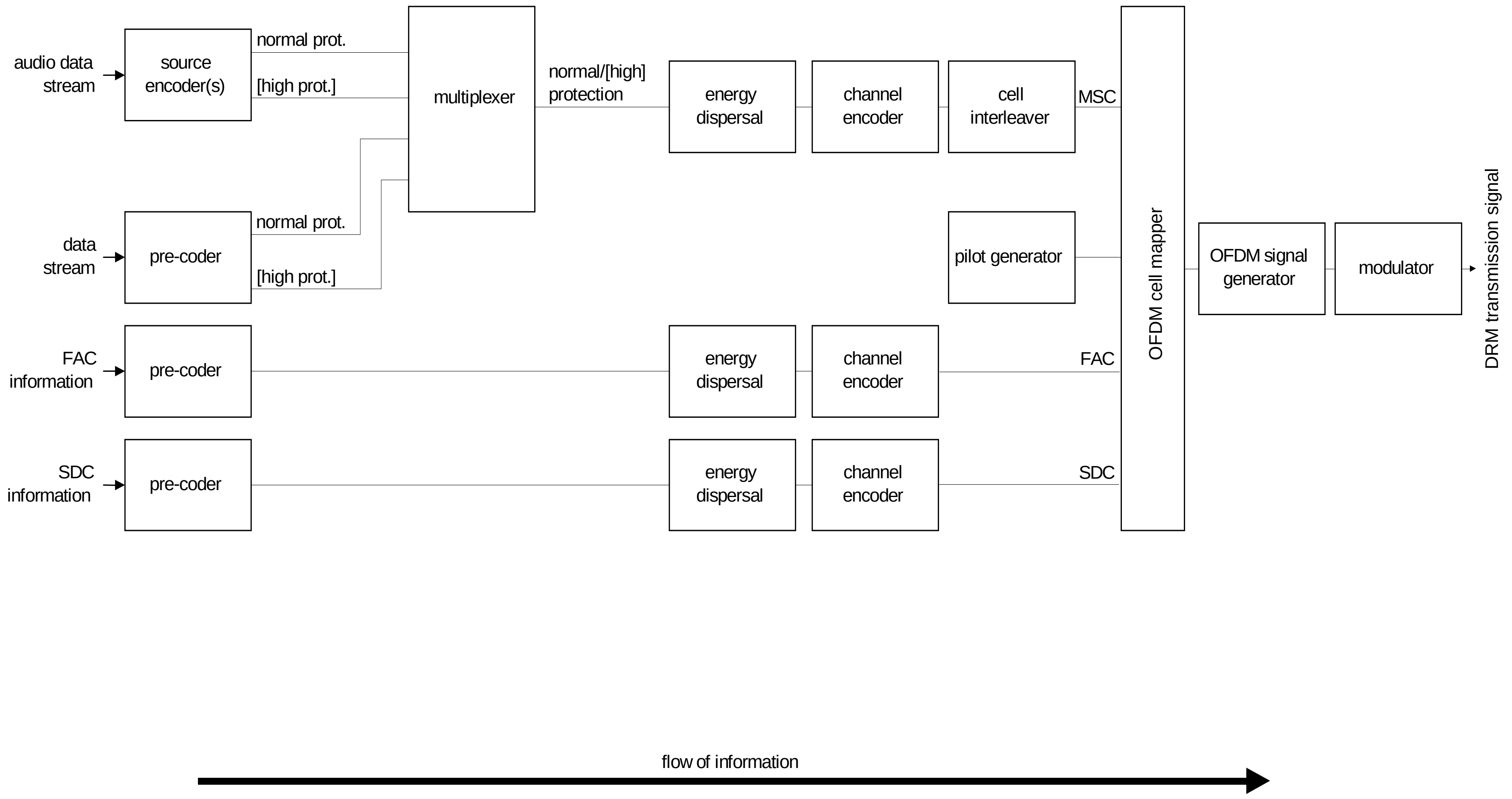


Figure 1: Conceptual DRM transmission block diagram

The source encoder and pre-coders ensure the adaptation of the input streams onto an appropriate digital transmission format. For the case of audio source encoding, this functionality includes audio compression techniques as described in clauses 4.3 and 5. The output of the source encoder(s) and the data stream pre-coder may comprise two parts requiring different levels of protection within the subsequent channel encoder. All services have to use the same two levels of protection.

The multiplexer combines the protection levels of all data and audio services as described in clause 6.

The energy dispersal provides a deterministic selective complementing of bits in order to reduce the possibility that systematic patterns result in unwanted regularity in the transmitted signal.

The channel encoder adds redundant information as a means for quasi error-free transmission and defines the mapping of the digital encoded information onto QAM cells as described in clause 7.

Cell interleaving spreads consecutive QAM cells onto a sequence of cells quasi-randomly separated in time and frequency, in order to provide robust transmission in time-frequency dispersive channels. The pilot generator provides means to derive channel state information in the receiver, allowing for a coherent demodulation of the signal.

The OFDM cell mapper collects the different classes of cells and places them on the time-frequency grid as specified in clause 7.

The OFDM signal generator transforms each ensemble of cells with same time index to a time domain representation of the signal. Consecutively, the OFDM symbol is obtained from this time domain representation by inserting a guard interval as a cyclic repetition of a portion of the signal, as specified in clause 7.

The modulator converts the digital representation of the OFDM signal into the analogue signal in the air. This operation involves digital-to-analogue conversion and filtering that have to comply with spectrum requirements as described in annex E.

4.3 Source coding

Within the constraints of broadcasting regulations in broadcasting channels below 30 MHz and the parameters of the coding and modulation scheme applied, the bit rate available for source coding is in the range from 8 kbit/s (half channels) to ≈20 kbit/s (standard channels) to up to ≈72 kbit/s (double channels).

In order to offer optimum quality at a given bit rate, the system offers different source coding schemes:

- a subset of MPEG-4 AAC (Advanced Audio Coding) including error robustness tools for generic mono and stereo audio broadcasting;
- a subset of MPEG-4 CELP speech coder for error robust speech broadcasting in mono, for cases when only a low bit rate is available or especially high error robustness is required;
- a subset of MPEG-4 HVXC speech coding for very low bit rate and error robust speech broadcasting in mono, especially well suited also for speech data base applications;
- Spectral Band Replication (SBR), an audio coding enhancement tool that allows to achieve full audio bandwidth at low bit rates. It can be applied to AAC and CELP.

The bit-stream transport format of the source coding schemes has been modified to meet the requirements of the DRM system (audio superframing). Unequal Error Protection (UEP) can be applied to improve the system behaviour in error prone channels.

Provision is made for further enhancement of the audio system by linking two DRM signals together.

4.4 Transmission modes

4.4.1 Signal bandwidth related parameters

The current channel widths for radio broadcasting below 30 MHz are 9 kHz and 10 kHz. The DRM system is designed to be used:

- within these nominal bandwidths, in order to satisfy the current planning situation;
- within half these bandwidths (4,5 kHz or 5 kHz) in order to allow for simulcast with analogue AM signals;
- within twice these bandwidths (18 kHz or 20 kHz) to provide for larger transmission capacity where and when the planning constraints allow for such facility.

These signal bandwidth related parameters are specified in clause 8.

4.4.2 Transmission efficiency related parameters

For any value of the signal bandwidth parameter, transmission efficiency related parameters are defined to allow a trade off between capacity (useful bit rate) and ruggedness to noise, multipath and Doppler. These parameters are of two types:

- coding rate and constellation parameters, defining which code rate and constellations are used to convey data;
- OFDM symbol parameters, defining the structure of the OFDM symbols to be used as a function of the propagation conditions.

4.4.2.1 Coding rates and constellations

As a function of the desired protection associated within each service or part of a service, the system provides a range of options to achieve one or two levels of protection at a time. Depending on service requirements, these levels of protection may be determined by either the code rate of the channel encoder (e.g. 0,6 …), by the constellation order (e.g. 4-QAM, 16-QAM, 64-QAM), or by hierarchical modulation. Detailed definition of these options is given in clause 7.

4.4.2.2 OFDM parameter set

The OFDM parameter set is presented in this clause. The specification of the signal waveform is given in clause 8. These values are defined for different propagation-related transmission conditions to provide various robustness modes for the signal. In a given bandwidth, the different robustness modes provide different available data rates. Table 1 illustrates typical uses of the robustness modes.

Table 1: Robustness mode uses

Robustness mode	Typical propagation conditions
A	Gaussian channels, with minor fading
B	Time and frequency selective channels, with longer delay spread
C	As robustness mode B, but with higher Doppler spread
D	As robustness mode B, but with severe delay and Doppler spread

The transmitted signal comprises a succession of OFDM symbols, each symbol being made of a guard interval followed by the so-called useful part of the symbol. Each symbol is the sum of K sine wave portions equally spaced in frequency. Each sine wave portion, called a "cell", is transmitted with a given amplitude and phase and corresponds to a carrier position. Each carrier is referenced by the index *k, k* belonging to the interval $[k_{\min}, k_{\max}]$ ($k = 0$ corresponds to the reference frequency of the transmitted signal).

The time-related OFDM symbol parameters are expressed in multiples of the elementary time period T , which is equal to $83^{1/3}$ µs. These parameters are:

- T_g : duration of the guard interval;
- T_s : duration of an OFDM symbol;
- T_u : duration of the useful (orthogonal) part of an OFDM symbol (i.e. excluding the guard interval).

The OFDM symbols are grouped to form transmission frames of duration T_f .

As specified in clause 8, a certain number of cells in each OFDM symbol are transmitted with a predetermined amplitude and phase, in order to be used as references in the demodulation process. They are called "reference pilots" and represent a certain proportion of the total number of cells.

Table 2: OFDM symbol parameters

Parameters list	Robustness mode			
	A	B	C	D
T (µs)	$83^{1/3}$	$83^{1/3}$	$83^{1/3}$	$83^{1/3}$
T_u (ms)	24 (288 × T)	$21^{1/3}$ (256 × T)	$14^{2/3}$ (176 × T)	$9^{1/3}$ (112 × T)
T_g (ms)	$2^{2/3}$ (32 × T)	$5^{1/3}$ (64 × T)	$5^{1/3}$ (64 × T)	$7^{1/3}$ (88 × T)
T_g / T_u	1/9	1/4	4/11	11/14
$T_s = T_u + T_g$ (ms)	$26^{2/3}$	$26^{2/3}$	20	$16^{2/3}$
T_f (ms)	400	400	400	400

5 Source coding modes

5.1 Overview

The source coding options in the DRM system are shown in figure 2.

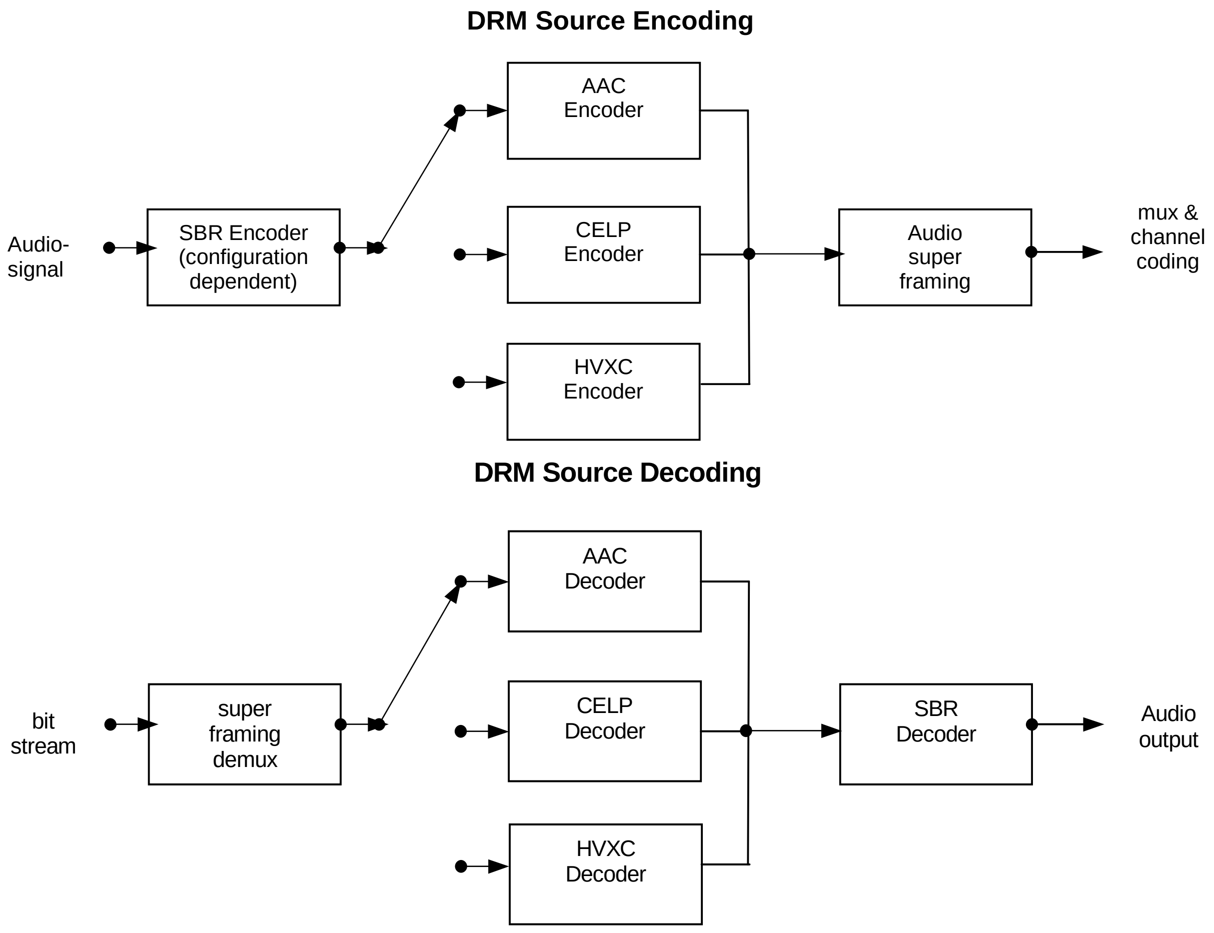


Figure 2: Source coding overview

As described in clause 4.3, the DRM system offers audio coding (AAC) and speech coding (CELP and HVXC). In general, a high frequency reconstruction method (SBR) can be used to enhance the perceptual audio quality. However, at present the use of SBR is only defined for use with AAC. Special care is taken so that the encoded audio can be composed into audio super frames of constant length. Multiplexing and UEP of audio/speech services is done by means of the multiplex and channel coding units. Audio specific configuration information is transmitted in the SDC (see clause 6.4.3.10).

5.1.1 AAC Audio Coding

For generic audio coding, a subset of the MPEG-4 Advanced Audio Coding (AAC) toolbox chosen to best suit the DRM system environment is used. For example a standard configuration for use in one short wave channel could be 20 kbit/s mono AAC.

Specific features of the AAC stream within the DRM system are:

- Bit rate: AAC can be used at any bit rate. Byte-alignment of the 400 ms audio super frame leads to a granularity of 20 bit/s for the AAC bit rate.
- Sampling rates: permitted sampling rates are 12 kHz and 24 kHz.
- Transform length: the transform length is 960 to ensure that one audio frame corresponds to 80 ms or 40 ms in time. This is required to harmonize CELP and AAC frame lengths and thus to allow the combination of an integer number of audio frames to build an audio super frame of 400 ms duration.
- Error robustness: a subset of MPEG-4 tools is used to improve the AAC bit stream error robustness in error prone channels.
- Audio super framing: 5 (12 kHz) or 10 (24 kHz) audio frames are composed into one audio super frame, which always corresponds to 400 ms in time. The audio frames in the audio super frames are encoded together such that each audio super frame is of constant length, i.e. that bit exchange between audio frames is only possible within an audio super frame. One audio super frame is always placed in one logical frame (see clause 6). In this way no additional synchronization is needed for the audio coding. Retrieval of frame boundaries and provisions for UEP are also taken care of within the audio super frame.
- UEP: better graceful degradation and better operation at higher BERs is achieved by applying UEP to the AAC bit stream. Unequal error protection is realized via the multiplex/coding units.

5.1.2 MPEG CELP coding

MPEG CELP speech coding is offered in the DRM system to allow for reasonable speech quality at bit rates significantly below the standard rate (for example "half rate" operation at 8 kbit/s). Possible scenarios for the use of the speech coder are:

- Dual/triple speech applications: instead of one audio program at 20 kbit/s to 24 kbit/s, the channel contains two or three speech signals of 8 kbit/s to 10 kbit/s each, allowing simultaneous speech transmissions.
- Speech services in addition to an audio service.
- Simulcast transmissions: in case of analogue/digital simulcast only bit rates as low as 8 kbit/s may be available.
- Very robust speech applications: due to its nature a speech coder can be expected to offer higher robustness against channel errors. Therefore 8 kbit/s speech coding can be used to do ultra robust speech coding in one channel.

Basic features of MPEG CELP coding are:

- 8 kHz or 16 kHz sampling rate;
- Bit rates between 4 kbit/s and 20 kbit/s;
- error robustness;
- composition of an integer number of CELP frames to build one audio super frame.

5.1.3 MPEG HVXC coding

MPEG-4 HVXC (Harmonic Vector eXcitation Coding) speech coding is offered in the DRM system to allow for reasonable speech quality at very low bit rates such as 2,0 kbit/s. The operating bit rates of HVXC open up new applications for DRM such as:

- Speech services in addition to an audio service.
- Multi-language application.
- Solid-state storage of multiple programs such as news, data base in a card radio (e.g. total of about 4,5 hours of radio programs can be stored in 4 MByte Flash memory).
- Time scale modification for fast playback/browsing of stored program.
- Highly error robust transmission with or without hierarchical modulation scheme.

Basic features of HVXC coding are:

- 8 kHz sampling rate;
- Bit rates of 2,0 kbit/s and 4,0 kbit/s for fixed rate coding;
- Time scale and pitch modification of arbitrary amounts;
- error robust syntax is supported, and a CRC tool can be used to improve the error resilience of the HVXC bitstream in error prone channels;
- composition of a constant integer number of HVXC frames (20) to build one audio super frame.

5.1.4 SBR coding

To maintain a reasonable perceived audio quality at low bit rates, classical audio or speech source coding algorithms need to limit the audio bandwidth and to operate at low sampling rates. It is desirable to be able to offer high audio bandwidth also in very low bit rate environments. This can be realized by the use of SBR (Spectral Band Replication).

The purpose of SBR is to recreate the missing high frequency band of the audio signal that could not be coded by the encoder. In order to do this in the best possible way, some side information needs to be transmitted in the audio bitstream, removing a small percentage of the available data rate from the audio coder. This side information is computed on the full bandwidth signal, prior to encoding and aids the reconstruction of the high frequencies after audio/speech decoding.

SBR exists in two versions. SBR-LC, a tool of low complexity, that offers intermediate sound quality, and SBR-HQ a tool that offers higher sound quality compared to SBR-LC, albeit at a somewhat higher complexity. Both versions are encoder and bitstream compatible and thus offer a "future-proof" upgrade concept. The version difference is only reflected in the decoder design.

The SBR technique is described in detail in clause 5.6.

5.2 UEP and audio super framing

Today's coding schemes are highly optimized in terms of coding efficiency and according to information theory this should lead to the fact, that the entropy of the bits is nearly equal. If this assumption is true, then the channel coding must be optimized, such that the total amount of residual errors usually referred to as bit error rate (BER) is minimized. This criterion can be fulfilled by a channel coding method called equal error protection (EEP), were all information bits are protected with the same amount of redundancy.

However, the audible effects of errors are not independent of the part of the bitstream that was hit by the error. This behaviour of unequal error sensitivity is well known for source coding schemes that are used in broadcast and communication systems, like DAB (Eureka 147) or GSM. The optimized solution to cope with this unequal error sensitivity is called unequal error protection (UEP). In such a system, higher protection is assigned to the more sensitive information, whereas lower protection is assigned to the less sensitive part of the bitstream.

To accommodate for UEP channel coding, it is necessary to have frames with a constant length and a UEP profile that is constant as well for a given bit rate. Since AAC is a coding scheme with a variable length, several coded frames are grouped together to build one audio super frame. The bit rate of the audio super frame is constant. Since the channel coding is based on audio super frames, the audio super frames themselves consist of two parts: a higher protected part and a lower protected part. Therefore, the coded audio frames itself have to be split into these two part. Further details on the audio super frame structure of AAC, CELP and HVXC are provided in the subsequent clauses. Note that HVXC is intended for use with the EEP scheme only.

Table 3: Syntax of audio_super_frame()

Syntax	No. of bits	Note
Audio_super_frame(audio_info) //audio info from the SDC		
{		
switch (audio_info.audio_coding) {		
case AAC: aac_super_frame(audio_info)		
break;		
case CELP: celp_super_frame(audio_info)		
break;		
case HVXC: hvxc_super_frame(audio_info)		
break;		
}		
}		

NOTE: The SDC describes the audio coder used, and the parameters associated with that coder. It also provides information about the sampling rate and bit rate used (see clause 6).

5.3 AAC coding

The following two clauses explain how the AAC and AAC + SBR frames fit into the audio super frame.

5.3.1 AAC

ISO/IEC 14496-3 [2] defines the MPEG-4 Audio standard. The audio coding standard MPEG-4 AAC is part of the MPEG-4 Audio standard. Two versions are defined, but only version 2 is intended for the use in error prone channels. The AAC bitstreams in the DRM system are therefore MPEG-4 version 2 bitstreams. From the possible audio object types, only the Error Robust (ER) AAC Low Complexity object type (Object Type ID = 17), which is part of the High Quality Audio Profile, is used in the DRM system.

DRM specific usage of MPEG-4 AAC: Three error robustness tools may be used within an MPEG-4 version 2 AAC bitstream: HCR (Huffman Codeword Reordering), VCB11 (Virtual Codebooks for Codebook 11) and RVLC (Reversible Variable Length Coding). In the DRM system, all AAC bitstreams shall use the HCR tool, since this tool reduces the error sensitivity of the bitstream significantly with a minimum of overhead. The VCB11 tool shall be used, since for low bit rates, the VCB11 overhead is less than 1 %. The RVLC tool is not used, since it introduces a significant bit rate overhead that is a major drawback for the low bit rates used by DRM.

The MPEG-4 AAC tool PNS (Perceptual Noise Substitution) is not used in DRM since SBR provides this functionality more appropriately.

For DRM the 960 transform shall be used.

When 12 kHz sampling is used, 5 AAC frames shall be combined into one audio super frame.

When 24 kHz sampling is used, 10 AAC frames shall be combined into one audio super frame.

The AAC sampling rate shall be 24 kHz when the stereo mode is used.

No standard extension_payload() shall be used and the only allowed extension is SBR (signalled via SDC).

The left and the right channel in one stereo audio frame are transmitted in an interleaved way to achieve a decreasing error sensitivity within the stereo frame.

The element_instance_tag for a single_channel_element(), respectively a channel_pair_element(), is not used in order to save 4 bits.

Any DRM AAC bitstream can easily be translated into an MPEG-4 V2 compliant bitstream by applying the above rules.

The MPEG-4 standard defines how the bits for one raw error robust AAC audio frame are stored. Each element of the error robust AAC bitstream is assigned an error sensitivity category. In the DRM system there are two possible error robust AAC audio frames:

mono audio frame

One mono audio frame consists of three consecutive parts, hereinafter called mono1, mono2 and mono3. Mono1 contains the Side Information (SI) bits, mono2 contains the Temporal Noise Shaping (TNS) bits and mono3 contains the spectral data bits. The error sensitivity decreases from mono1 to mono3.

stereo audio frame

One stereo audio frame consists of seven consecutive parts, hereinafter called stereo1 (common side info), stereo2 (side info left channel), stereo3 (side info right channel), stereo4 (TNS left channel), stereo5 (TNS right channel), stereo6 (spectral data left channel), stereo7 (spectral data right channel). With this interleaving of left and right channel, the error sensitivity is decreasing from stereo1 to stereo7.

5.3.1.1 AAC audio super frame

Table 4: Syntax of aac_super_frame()

Syntax	No. of bits	Note
aac_super_frame(audio_info) //audio info from the SDC		
{		
switch (audio_info.audio_sampling_rate) {//only 12 000 and 24 000 is allowed		
case 12 000: num_frames = 5;		
break;		
case 24 000: num_frames = 10;		
break;		
}		
aac_super_frame_header(num_frames - 1)		
for (f = 0; f < num_frames; f++) {		
//higher_protected_block		
for (b = 0; b < num_higher_protected_bytes; b++)		
audio_frame[f][b]	**8**	
aac_crc_bits[f]	**8**	see annex D
}		
//lower_protected_part		
for (f = 0; f < num_frames; f++) {		
num_lower_protected_bytes = frame_length[f] - num_higher_protected_bytes;		
for (b = 0; b < num_lower_protected_bytes; b++)		
audio_frame[f][num_higher_protected_bytes + b]	**8**	
}		
}		
NOTE 1: num_higher_protected_bytes is derived from the UEP profile used (see clause 6).		
NOTE 2: audio_frame is either an AAC or an AAC + SBR frame.		

Table 5: Syntax of aac_super_frame_header()

Syntax	No. of bits	Note
aac_super_frame_header(num_borders)		
{		
previous_border = 0;		
for (n = 0; n < num_borders; n++) {		
frame_length[n] = **frame_border** - previous_border; //frame border in bytes	**12**	
previous_border = frame_border;		
}		
frame_length[num_borders] = audio_payload_length - previous_border;		
if (num_borders == 9)		
reserved //byte-alignment	**4**	
}		
NOTE: The audio_payload_length is derived from the length of the audio super frame (data_length_of_part_A + data_length_of_part_B) subtracting the audio super frame overhead (bytes used for the audio super frame header() and for the aac_crc_bits).		

higher protected part

The higher protected part contains one header followed by num_frames higher protected blocks. num_frames is the number of audio frames in the audio super frame.

header

The header contains information to recover the frame lengths of the num_frames AAC frames stored in the audio super frame.

All the frame lengths are derived from the absolute positions of the frame borders. These frame borders are stored consecutively in the header. Each frame border occupies 12 bits (unsigned integer, most significant bit first). The frame border is measured in bytes from the start of the AAC bitstream sequence. 4 padding bits are added in case num_frames==10. num_frames-1 frame borders are stored in the header.

higher protected block

One higher protected block contains a certain amount of bytes from the start of each AAC frame, dependent upon the UEP profile. One 8-bit CRC check derived from the CRC-bits of the corresponding AAC frame follows (see annex D for CRC calculation). For a mono signal, the CRC-bits cover (mono1, mono2). For a stereo signal, the CRC-bits cover (stereo1, stereo2, stereo3, stereo4, stereo5).

lower protected part

The lower protected bytes (the remaining bytes not stored in the higher protected part) of the AAC frames are stored consecutively in the lower protected part.

Figure 3 illustrates an example audio super frame for a 24 kHz sampled signal.

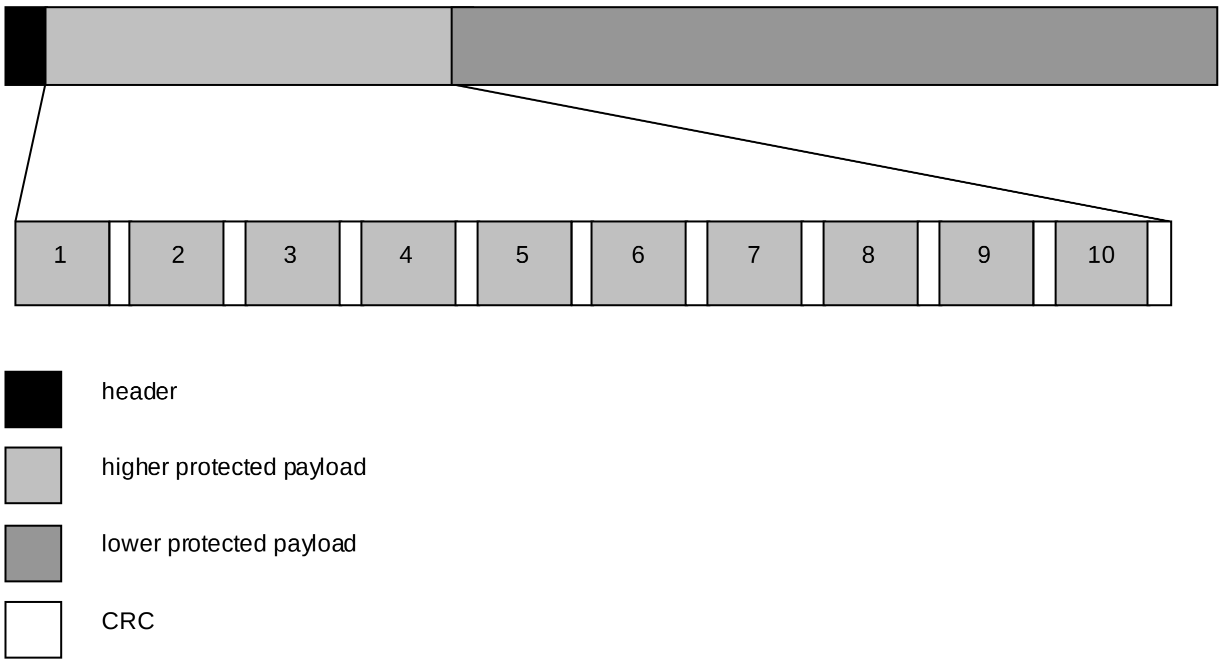


Figure 3: Example AAC audio super frame (24 kHz)

5.3.2 AAC + SBR

The SBR sampling rate shall be 48 kHz and the AAC sampling rate shall be 24 kHz. One raw AAC + SBR frame contains an AAC part and a SBR part. The SBR part of the data is located at the end of the frame. The first bit in the SBR-bitstream is the last bit in the frame, and the SBR bits are thus written/read in reversed order. In this way, the starting points of respective part of the frame data are always easily found.

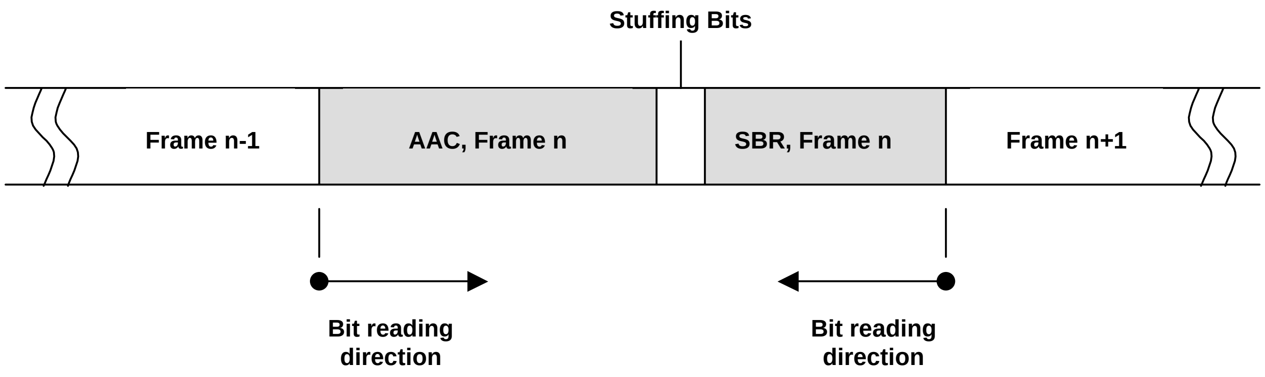


Figure 4: AAC + SBR frame

Both AAC and SBR data-sizes vary from frame to frame. The total size of the individual frames, now including the SBR data, can be derived from the aac_super_frame_header() as described in clause 5.3.1. Thus no extra signalling due to the varying SBR bit rate is needed.

The AAC + SBR frames are inserted into the audio super frame in the same manner as when SBR is not used.

For source coding bit rates at 20 kbit/s or greater, SBR shall be used. For bit rates below 20 kbit/s, SBR may be used.

The details of the SBR-bitstream are described in clause 5.5.3.

5.4 MPEG CELP coding

5.4.1 MPEG CELP

ISO/IEC 14496-3 [2] defines the MPEG-4 Audio standard. The speech coding standard MPEG-4 CELP (Code Excited Linear Prediction) is part of the MPEG-4 Audio standard. Two versions are defined, but only version 2 is intended for the use in error prone channels. The CELP bitstreams in the DRM system are therefore MPEG-4 version 2 bitstreams. From the possible audio object types, only the Error Robust (ER) CELP object type (Object Type ID = 24), which is part of the High Quality Audio Profile, is used in the DRM system.

The MPEG-4 CELP covers the compression and decoding of natural speech sound at bit rates ranging between 4 kbit/s and 24 kbit/s. MPEG-4 CELP is a well-known coding algorithm with new functionality. Conventional CELP coders offer compression at a single bit rate and are optimized for specific applications. Compression is one of the functionalities provided by MPEG-4 CELP, but MPEG-4 also enables the use of one basic coder in multiple applications. It provides scalability in bit rate and bandwidth, as well as the ability to generate bitstreams at arbitrary bit rates. The MPEG-4 CELP coder supports two sampling rates, namely, 8 kHz and 16 kHz. The associated bandwidths are 100 Hz to 3 800 Hz for 8 kHz sampling and 50 Hz to 7 000 Hz for 16 kHz sampling.

A basic block diagram of the CELP decoder is given in figure 5.

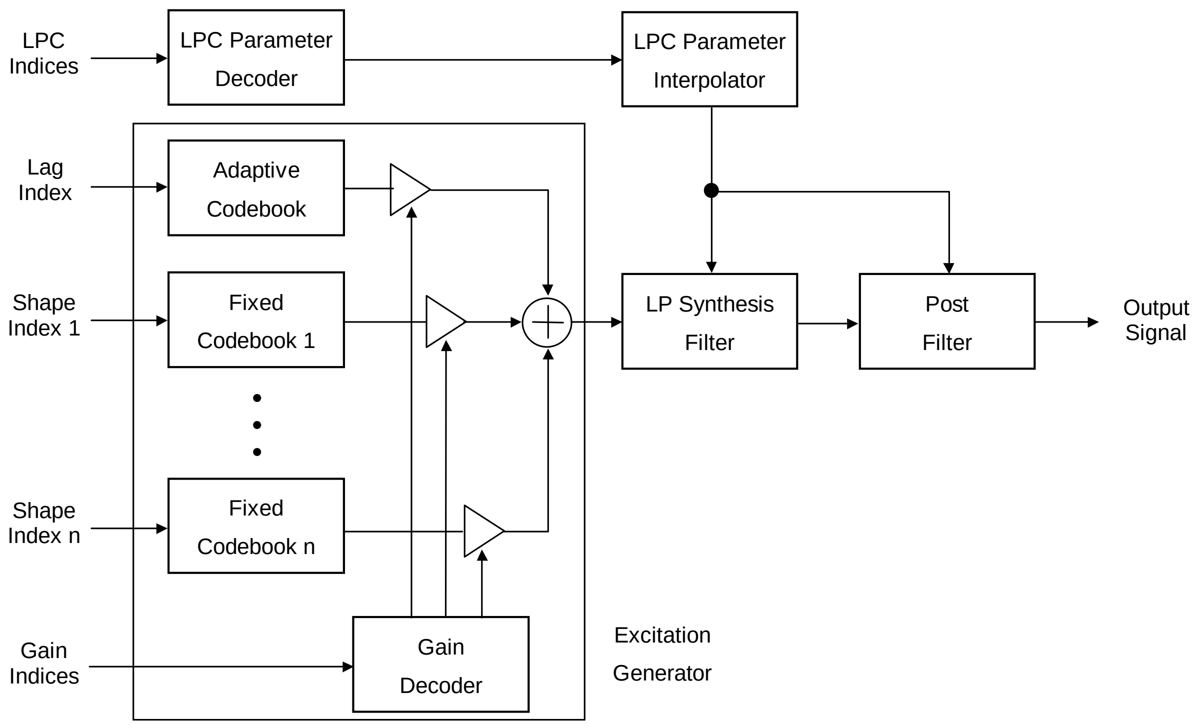


Figure 5: Block diagram of a CELP decoder

The CELP decoder primarily consists of an excitation generator and a synthesis filter. Additionally, CELP decoders often include a post-filter. The excitation generator has an adaptive codebook to model periodic components, fixed codebooks to model random components and a gain decoder to represent a speech signal level. Indices for the codebooks and gains are provided by the encoder. The codebook indices (pitch-lag index for the adaptive codebook and shape index for the fixed codebook) and gain indices (adaptive and fixed codebook gains) are used to generate the excitation signal. The excitation signal is then filtered by the linear predictive synthesis filter (LP synthesis filter). Filter coefficients are reconstructed using the LPC indices, then are interpolated with the filter coefficients of successive analysis frames. Finally, a post-filter can optionally be applied in order to enhance the speech quality.

The MPEG-4 CELP coder offers the following functionalities: Multiple bit rates, Bit rate Scalability, Bandwidth Scalability, and Fine Rate Control. DRM only uses the multiple bit rates functionality.

Multiple bit rates: the available bit rates depend on the sampling rate. The following fixed bit rates can be used:

Table 6: Fixed bit rates for the CELP coder

Bit rates for the 8 kHz sampling rate (bit/s)	Bit rates for the 16 kHz sampling rate (bit/s)
3 850, 4 250, 4 650, 5 700, 6 000, 6 300, 6 600, 6 900, 7 100, 7 300, 7 700, 8 300, 8 700, 9 100, 9 500, 9 900, 10 300, 10 500, 10 700, 11 000, 11 400, 11 800, 12 000, 12 200	10 900, 11 500, 12 100, 12 700, 13 300, 13 900, 14 300, 14 700, 15 900, 17 100, 17 900, 18 700, 19 500, 20 300, 21 100, 13 600, 14 200, 14 800, 15 400, 16 000, 16 600, 17 000, 17 400, 18 600, 19 800, 20 600, 21 400, 22 200, 23 000, 23 800

The algorithmic delay of the CELP coder comes from the frame length and an additional look ahead length. The frame length depends on the coding mode and the bit rate. The look ahead length, which is an informative parameter, also depends on the coding mode. The delays presented below are applicable to the modes used in DRM.

Table 7: Delay and frame length for the CELP coder at 8 kHz sampling rate

Bit rate (bit/s)	Delay (ms)	Frame length (ms)
3 850, 4 250, 4 650	45	40
5 700, 6 000, 6 300, 6 600, 6 900, 7 100, 7 300, 7 700, 8 300, 8 700, 9 100, 9 500, 9 900, 10 300, 10 500, 10 700	25	20
11 000, 11 400, 11 800, 12 000, 12 200	15	10

Table 8: Delay and frame length for the CELP coder at 16 kHz sampling rate

Bit rate (bit/s)	Delay (ms)	Frame length (ms)
10 900, 11 500, 12 100, 12 700, 13 300, 13 900, 14 300, 14 700, 15 900, 17 100, 17 900, 18 700, 19 500, 20 300, 21 100	25	20
13 600, 14 200, 14 800, 15 400, 16 000, 16 600, 17 000, 17 400, 18 600, 19 800, 20 600, 21 400, 22 200, 23 000, 23 800	15	10

5.4.1.1 CELP audio super frame

CELP frames have a fixed frame length. The CELP audio frames are grouped together to form audio super frames of 400 ms duration. UEP is applicable. The start of each audio frame is mapped into the higher protected part, the remaining bits are allocated to the lower protected part. The partitioning of the CELP frames is given in tables 10 and 11. The CELP bit rate index is signalled in the SDC.

Table 9: Syntax of celp_super_frame()

Syntax	No. of bits	Note
celp_super_frame(celp_table_ind) //CELP table index from the SDC		
{		
switch (audio_info.audio_sampling_rate) {//only 8 000 and 16 000 is allowed		
case 8 000:		
(num_frames, num_higher_protected_bits, num_lower_protected_bits)		
= read_table_10 (CELP_index)		
break;		
case 16 000:		
(num_frames, num_higher_protected_bits, num_lower_protected_bits)		
= read_table_11 (CELP_index)		
break;		
}		
for (f = 0; f < num_frames; f++) {		
//higher_protected_block		
for (b = 0; b < num_higher_protected_bits; b++)		
celp_frame[f][b]	**1**	
if (audio_info.CELP_CRC == 1)		
celp_crc_bits[f]	**8**	see annex D
}		
//lower_protected_part		
for (f = 0; f < num_frames; f++) {		
for (b = 0; b < num_lower_protected_bits; b++)		
celp_frame[f][num_higher_protected_bits + b]	**1**	
}		
}		

Table 10: UEP parameters for 8 kHz sampling CELP

CELP bit rate index	Bit rate (bits/s)	Audio frame length (ms)	Higher protected part (bits/audio frame)	Lower protected part (bits/audio frame)	Higher protected part (bytes/audio super frame)	Lower protected part (bytes/audio super frame)	Audio super frame length (bytes)
0	3 850	40	36	118	45	148	193 (see note)
1	4 250	40	36	134	45	168	213 (see note)
2	4 650	40	36	150	45	188	233 (see note)
6	5 700	20	24	90	60	225	285
7	6 000	20	24	96	60	240	300
8	6 300	20	24	102	60	255	315
9	6 600	20	24	108	60	270	330
10	6 900	20	24	114	60	285	345
11	7 100	20	24	118	60	295	355
12	7 300	20	24	122	60	305	365
13	7 700	20	36	118	90	295	385
14	8 300	20	36	130	90	325	415
15	8 700	20	36	138	90	345	435
16	9 100	20	36	146	90	365	455
17	9 500	20	36	154	90	385	475
18	9 900	20	36	162	90	405	495
19	10 300	20	36	170	90	425	515
20	10 500	20	36	174	90	435	525
21	10 700	20	36	178	90	445	535
22	11 000	10	24	86	120	430	550
23	11 400	10	24	90	120	450	570
24	11 800	10	24	94	120	470	590
25	12 000	10	24	96	120	480	600
26	12 200	10	24	98	120	490	610
NOTE: For these bit rates, the last four bits of the audio super frame are padded with 0s.							

Table 11: UEP parameters for 16 kHz sampling CELP

CELP bit rate index	Bit rate (bits/s)	Audio frame length (ms)	Higher protected part (bits/audio frame)	Lower protected part (bits/audio frame)	Higher protected part (bytes/audio super frame)	Lower protected part (bytes/audio super frame)	Audio super frame length (bytes)
0	10 900	20	64	154	160	385	545
1	11 500	20	64	166	160	415	575
2	12 100	20	64	178	160	445	605
3	12 700	20	64	190	160	475	635
4	13 300	20	64	202	160	505	665
5	13 900	20	64	214	160	535	695
6	14 300	20	64	222	160	555	715
8	14 700	20	92	202	230	505	735
9	15 900	20	92	226	230	565	795
10	17 100	20	92	250	230	625	855
11	17 900	20	92	266	230	665	895
12	18 700	20	92	282	230	705	935
13	19 500	20	92	298	230	745	975
14	20 300	20	92	314	230	785	1 015
15	21 100	20	92	330	230	825	1 055
16	13 600	10	50	86	250	430	680
17	14 200	10	50	92	250	460	710
18	14 800	10	50	98	250	490	740
19	15 400	10	50	104	250	520	770
20	16 000	10	50	110	250	550	800
21	16 600	10	50	116	250	580	830
22	17 000	10	50	120	250	600	850
24	17 400	10	64	110	320	550	870
25	18 600	10	64	122	320	610	930
26	19 800	10	64	134	320	670	990
27	20 600	10	64	142	320	710	1 030
28	21 400	10	64	150	320	750	1 070
29	22 200	10	64	158	320	790	1 110
30	23 000	10	64	166	320	830	1 150
31	23 800	10	64	174	320	870	1 190

5.5 HVXC

The MPEG-4 HVXC (Harmonic Vector eXcitation Coding) speech coding toolset as defined in ISO/IEC 14496-3 [2] covers the compression and decoding of natural speech sound at bit rates of 2,0 kbit/s and 4,0 kbit/s. HVXC employs harmonic coding of LPC residual signals for voiced segments and Vector eXcitation Coding (VXC) for unvoiced segments. HVXC provides communications-quality to near-toll-quality speech in the 100 Hz to 3 800 Hz band at 8 kHz sampling rate. In addition, the functionality of pitch and speed change during decoding is supported. This functionality is useful for fast speech database search or browsing. HVXC has a syntax providing error sensitivity categories that can be used with an error robustness tool. Additionally the error concealment functionality is supported for the use in error-prone channels.

DRM uses a subset of the HVXC description in ISO/IEC 14496-3 [2], which limits the syntax to the error robust syntax and the data rates to the two options of 2,0 kbit/s and 4,0 kbit/s. Further, HVXC is used with the non-scalable syntax only. For robust decoding in error-prone channels a low-complexity error concealment tool (CRC and intra-frame interleaving) is defined specifically for DRM.

The syntax of the HVXC audio super frame is identical for all possible HVXC modes, since HVXC does not support UEP functionality and the length of a HVXC audio frame is always 20 ms.

Table 12: Syntax of hvxc_super_frame()

Syntax	No. of bits	Note

hvxc_super_frame(audio_info) //audio info from the SDC
{
num_frames = 20;
}		

The number of bits contained in one audio frame is given by the audio information from the SDC (HVXC_rate, HVXC_CRC).

In case the 4,0 kbit/s fixed rate HVXC coder is used with the CRC tool, the last 4 bits of each audio super frame are padded with zeros and the receiver shall ignore these bits. The resulting bit rate therefore is 4,66 kbit/s.

Only the fixed rate modes of HVXC (2,0 kbit/s or 4,0 kbit/s) are used in audio super frames. Variable rate modes may be applicable for use with packet mode applications in the future.

5.5.1 Definitions

5.5.1.1 HVXC source coder parameters

The definition of the basic data entities of a MPEG-4 compliant HVXC speech coding system is given in ISO/IEC 14496-3 [2] and is reproduced as table N.1.

5.5.1.2 CRC bits for fixed bit rate modes

Table 13 describes the various CRC bits added for error protection in the DRM system.

Table 13: CRC bits for fixed bit rate modes

Parameter	Description	length (bits)
CRC0_2k	CRC bits for ESC0 at 2 kbit/s	6
CRC1_2k	CRC bits for ESC1 at 2 kbit/s	1
CRC2_2k	CRC bits for ESC2 at 2 kbit/s	1
CRC0_4k	CRC bits for ESC0 at 4 kbit/s	6
CRC1_4k	CRC bits for ESC1 at 4 kbit/s	5
CRC2_4k	CRC bits for ESC2 at 4 kbit/s	1
CRC3_4k	CRC bits for ESC3 at 4 kbit/s	1

5.5.2 HVXC decoder

Figure 6 shows the overall structure of the HVXC decoder. The basic decoding process is composed of four steps; de-quantization of parameters, generation of excitation signals for voiced frames by sinusoidal synthesis (harmonic synthesis) and noise component addition, generation of excitation signals for unvoiced frames by codebook look-up, and LPC synthesis. To enhance the synthesized speech quality spectral post-filtering is used.

For voiced frames, a fixed dimension harmonic spectral vector, obtained by de-quantization of the spectral magnitude, is first converted to the one having the original dimension which varies frame by frame in accordance with the pitch value. This is done by the dimension converter in which a band-limited interpolator generates a set of spectral magnitude values at harmonic frequencies without changing the shape of the spectral envelope. Using the spectral magnitude values, a time domain excitation signal is then generated by the fast harmonic synthesis algorithm using an IFFT. In order to make the synthesized speech sound natural, a noise component is additionally used. A Gaussian noise spectral component, covering 2 kHz to 3,8 kHz, is coloured in accordance with the harmonic spectral magnitudes in the frequency domain, and its IDFT is added to voiced excitation signals in the time domain. The amount and bandwidth of this additive noise is controlled by the transmitted two-bit V/UV value, which is encoded based on the normalized maximum autocorrelation of the LPC residual signal. Noise added harmonic excitation signals for voiced segments are then fed into the LPC synthesis filter followed by the postfilter.

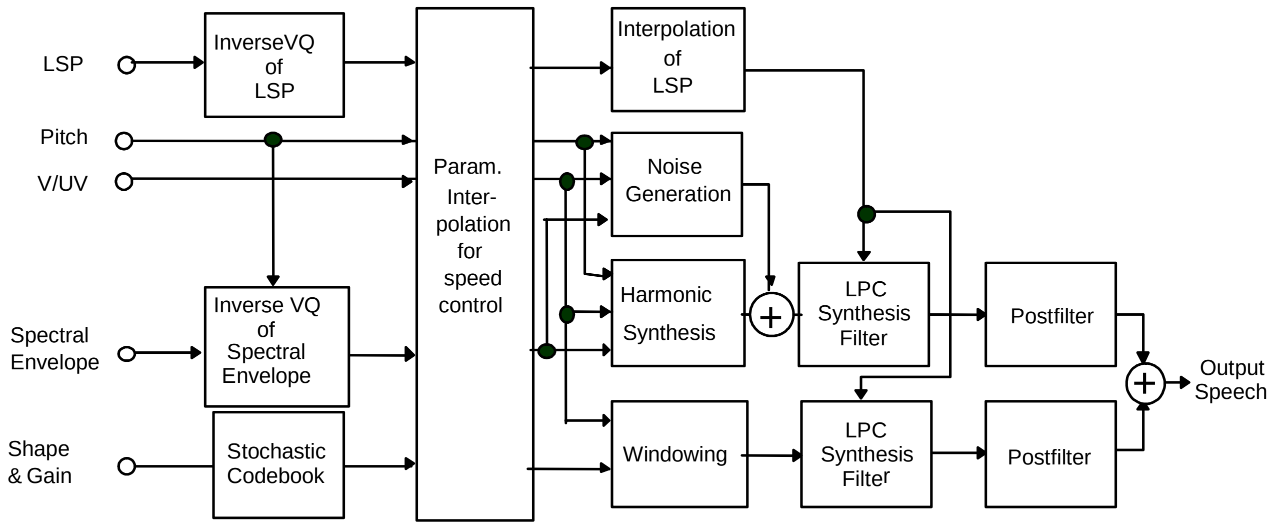


Figure 6: Block diagram of the HVXC decoder

For unvoiced segments, the usual VXC decoding algorithm is used where an excitation signal is generated by multiplying the gain value with the stochastic code vector. The result is then fed into the LPC synthesis filter followed by the postfilter. Finally, the synthesized speech components for voiced and unvoiced segments are added to form the output signal. The description of the time-scale modification standard can be found in the MPEG-4 [2].

5.5.3 HVXC encoder

Figure 7 shows the overall structure of the encoder of the MPEG-4 HVXC. Table N.2 shows the bit allocations for the 2,0 kbit/s and 4,0 kbit/s coders using fixed rate coding. The parameters followed by (enh) are used only for the 4,0 kbit/s mode. Operation of each part is described in figure 7.

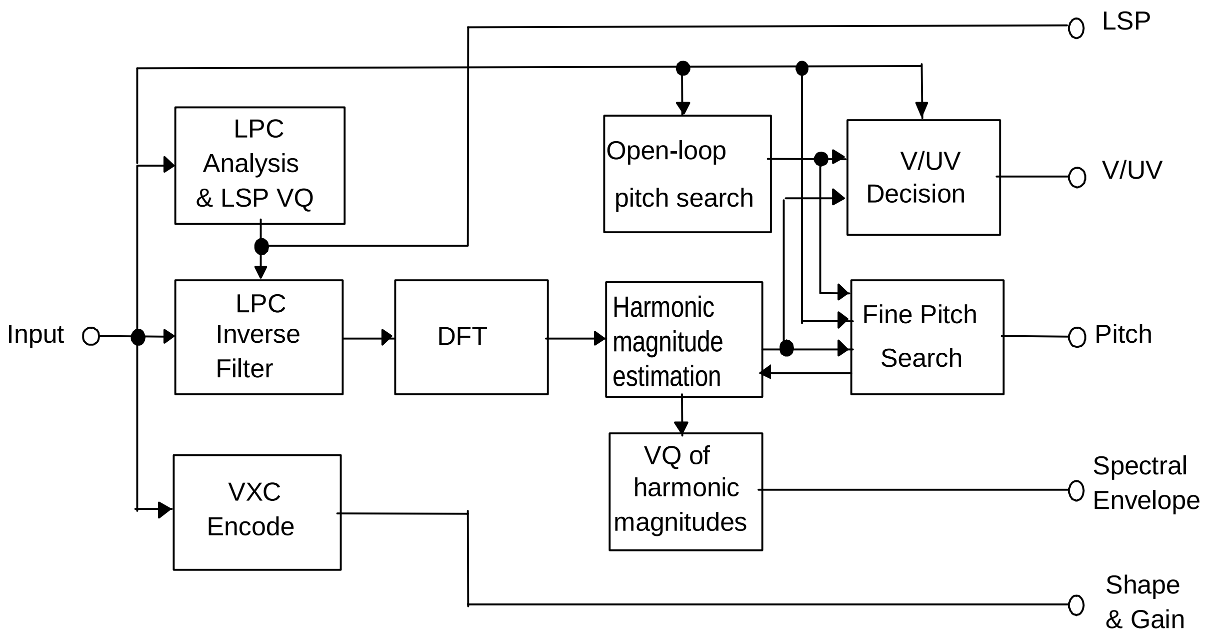


Figure 7: Block diagram of the HVXC encoder

5.5.3.1 LPC analysis and LSP quantization

The speech input at a sampling rate of 8 kHz is formed into frames with a length and interval of 256 samples and 160 samples, respectively. Tenth order LPC analysis is carried out using windowed input data over one frame. LPC parameters are converted to LSP parameters and vector quantized with a partial prediction and multi-stage vector quantization scheme. LPC residual signals are computed by inverse filtering the input data using quantized and interpolated LSP parameters.

5.5.3.2 Open loop pitch search

The open loop pitch value is estimated based on the peak values of the autocorrelation of the LPC residual signals. Using estimated past and current pitch values, pitch tracking is conducted to have a continuous pitch contour and to make the reliability of the pitch estimation higher. The voiced/unvoiced decision of the previous frame is also used to ensure the pitch tracking operation.

5.5.3.3 Harmonic magnitude and fine pitch estimation

The power spectrum of the LPC residual signal is then fed into the fine pitch and harmonic magnitude estimation block, where the harmonic spectral envelope of the LPC residual signal is estimated as follows: A basis spectrum representing one harmonic spectrum is gain scaled and arranged with the spacing of the fundamental frequency obtained by the open loop pitch search. The gain scaling for each harmonic and fundamental frequency is adjusted simultaneously so that the difference between the synthesized power spectrum and actual LPC residual spectrum is minimized. The harmonic spectral envelope for a voiced segment is then vector quantized.

5.5.3.4 Vector quantization of harmonic magnitudes

In order to vector quantize a spectral envelope composed of a variable number of harmonics, the harmonic spectral vector is first converted to a fixed-dimension vector. Band-limited interpolation by a polyphase filter bank is used for the dimensional conversion. A fixed-dimension spectral vector $\boldsymbol{x}$ is then quantized with weighted distortion measure D;

$$D = \left\| \boldsymbol{WH}(\boldsymbol{x} - g(\boldsymbol{s}_0 + \boldsymbol{s}_1)) \right\|^2$$

where $\mathbf{s}_0$ is the output of the first shape codebook, $\mathbf{s}_1$ is the output of the second shape codebook, and g is the output of the gain codebook. The diagonal components of the matrices $\mathbf{H}$ and $\mathbf{W}$ are the magnitudes of the frequency response of the LPC synthesis filter and the perceptual weighting filter, respectively. In order to reduce the memory requirements and search complexity while maintaining a high performance, two-stage vector quantization scheme is employed for the spectral shape together with a scalar quantizer for the gain under 2,0 kbit/s coding. For the 4,0 kbit/s mode, the quantized harmonic magnitude vector with fixed dimension is first converted to the dimension of original harmonics by the same band-limited interpolation mentioned above. The difference between the original harmonics and de-quantized and dimension converted harmonics are then quantized with additional vector quantizers. This multi-stage structure allows generation of scalable bit-streams.

5.5.3.5 Voiced/Unvoiced decision

The Voiced/Unvoiced decision is made based on the maximum autocorrelation of the LPC residual signals, the number of zero crossing and the harmonic structure of the power spectrum of the LPC residual signals.

5.5.3.6 VXC coding of unvoiced signals

For unvoiced segments, regular VXC coding is carried out, where only stochastic codebooks are used. A 6 bits shape codebook of dimension 80 and 4 bits gain codebook are used for the 2,0 kbit/s mode. For the 4,0 kbit/s mode, the quantization error of the 2,0 kbit/s mode is quantized using a 5 bits shape codebook of dimension 40 and a 3 bits gain codebook at the additional stage.

5.5.4 HVXC channel coding

5.5.4.1 Protected bit selection

According to the sensitivity of bits, encoded bits are classified to several Error Sensitivity Categories (ESC). The number of bits for each ESC is shown in table 14 (2,0 kbit/s, voiced sound), table 15 (2,0 kbit/s, unvoiced sound), table 16 (4,0 kbit/s, voiced sound) and table 17 (4,0 kbit/s, unvoiced sound). ESC0 is the group of the most error sensitive bits and ESC4 is the group of the least sensitive bits. Bit rate setting of total of 2,4 kbit/s using 2,0 kbit/s source coder rate, and total of 4,66 kbit/s using 4,0 kbit/s source coder rate are shown.

NOTE: The overall bit rate due to the usage of a CRC is 4,65 kbit/s, but additionally 4 padding bits have to be inserted for each audio super frame, resulting in an overall bit rate of 4,66 kbit/s, see clause 5.5.

Table 14: Number of ESC bits at 2,0 kbit/s fixed rate mode (voiced sound)

Parameters	**Voiced frame**				
	ESC0 (bits)	**ESC1 (bits)**	**ESC2 (bits)**	**ESC3 (bits)**	**total (bits)**
LSP1	5	-	-	-	5
LSP2	2	-	-	5	7
LSP3	1	-	-	4	5
LSP4	1	-	-	-	1
VUV	2	-	-	-	2
Pitch	6	-	-	1	7
SE_gain	5	-	-	-	5
SE_shape1	-	4	-	-	4
SE_shape2	-	-	4	-	4
total	**22**	**4**	**4**	**10**	**40**
CRC	6	1	1	-	8
total + CRC	**28**	**5**	**5**	**10**	**48**

Table 15: Number of ESC bits at 2,0 kbit/s fixed rate mode (unvoiced sound)

Parameters	**Unvoiced frame**				
	ESC0 (bits)	**ESC1 (bits)**	**ESC2 (bits)**	**ESC3 (bits)**	**total (bits)**
LSP1	5	-	-	-	5
LSP2	4	3	-	-	7
LSP3	2	1	2	-	5
LSP4	1	-	-	-	1
VUV	2	-	-	-	2
VX_gain1 [0]	4	-	-	-	4
VX_gain1 [1]	4	-	-	-	4
VX_shape1 [0]	-	-	2	4	6
VX_shape1 [1]	-	-	-	6	6
total	**22**	**4**	**4**	**10**	**40**
CRC	6	1	1	-	8
total + CRC	**28**	**5**	**5**	**10**	**48**

Table 16: Number of ESC bits at 4,0 kbit/s fixed rate mode (voiced sound)

Parameters	**Voiced frame**					
	ESC0 (bits)	**ESC1 (bits)**	**ESC2 (bits)**	**ESC3 (bits)**	**ESC4 (bits)**	**total (bits)**
LSP1	5	-	-	-	-	5
LSP2	4	-	-	-	3	7
LSP3	1	-	-	-	4	5
LSP4	1	-	-	-	-	1
LSP5	1	-	-	-	7	8
VUV	2	-	-	-	-	2
Pitch	6	-	-	-	1	7
SE_gain	5	-	-	-	-	5
SE_shape1	-	-	4	-	-	4
SE_shape2	-	-	-	4	-	4
SE_shape3	5	-	-	-	2	7
SE_shape4	1	9	-	-	-	10
SE_shape5	1	8	-	-	-	9
SE_shape6	1	5	-	-	-	6
Total	**33**	**22**	**4**	**4**	**17**	**80**
CRC	6	5	1	1	-	13
Total + CRC	**39**	**27**	**5**	**5**	**17**	**93**

Table 17: Number of ESC bits at 4,0 kbit/s fixed rate mode (unvoiced sound)

Parameters	**Unvoiced frame**					
	ESC0 (bits)	**ESC1 (bits)**	**ESC2 (bits)**	**ESC3 (bits)**	**ESC4 (bits)**	**total (bits)**
LSP1	5	-	-	-	-	5
LSP2	4	3	-	-	-	7
LSP3	1	4	-	-	-	5
LSP4	1	-	-	-	-	1
LSP5	1	7	-	-	-	8
VUV	2	-	-	-	-	2
VX_gain1 [0]	4	-	-	-	-	4
VX_gain1 [1]	4	-	-	-	-	4
VX_shape1 [0]	-	6	-	-	-	6
VX_shape1 [1]	-	1	4	1	-	6
VX_gain2 [0]	3	-	-	-	-	3
VX_gain2 [1]	3	-	-	-	-	3
VX_gain2 [2]	3	-	-	-	-	3
VX_gain2 [3]	2	1	-	-	-	3
VX_shape2 [0]	-	-	-	3	2	5
VX_shape2 [1]	-	-	-	-	5	5
VX_shape2 [2]	-	-	-	-	5	5
VX_shape2 [3]	-	-	-	-	5	5
total	**33**	**22**	**4**	**4**	**17**	**80**
CRC	6	5	1	1	-	13
total + CRC	**39**	**27**	**5**	**5**	**17**	**93**

5.5.4.2 Syntax of DRM HVXC error robustness (ErHVXCfixframe_CRC)

The bitstream syntax consists of several Error Sensitivity Categories (ESC). Some ESCs include source bits and CRC bits, where CRC bits are computed from source bits within the same ESC.

The HVXC_CRC field in the SDC is used to indicate whether the stream includes CRC parity bits or not (see clause 6.4.3.10). The bitstream syntax of the input of the CRC checker is given in figure 8.

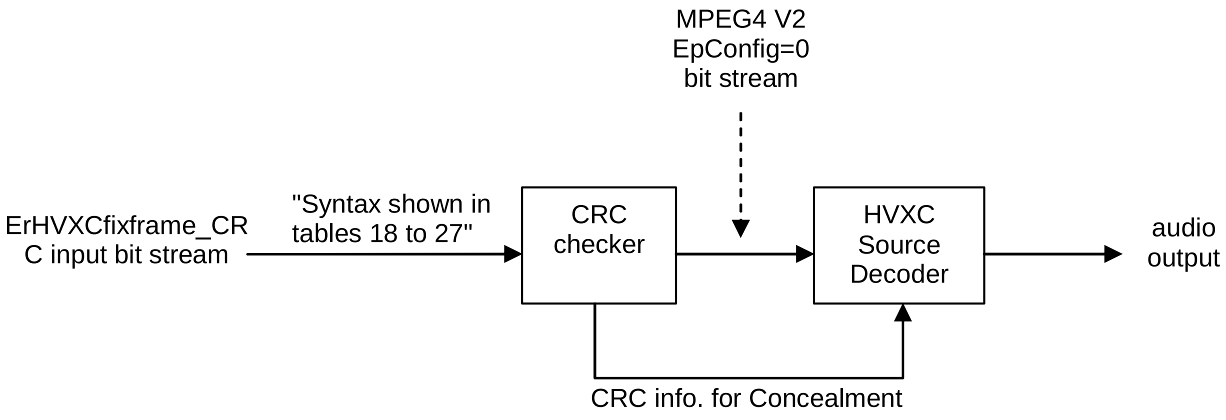

MPEG4 V2
EpConfig=0
bit stream
ErHVXCfixframe_CRC input bit stream
"Syntax shown in tables 18 to 27"
CRC checker
HVXC Source Decoder
audio output
CRC info. for Concealment

Figure 8: Block diagram of HVXC syntax

Table 18: Syntax of ErHVXCfixframe_CRC()

Syntax	No. of bits	Mnemonic
ErHVXCfixframe_CRC(rate)		
{		
if (rate == 2000) {		
2k_ESC0_CRC0();		
2k_ESC1_CRC1()		
2k_ESC2_CRC2();		
2k_ESC3_NoCRC();		
} else {		
4k_ESC0_CRC0();		
4k_ESC1_CRC1()		
4k_ESC2_CRC2();		
4k_ESC3_CRC3();		
4k_ESC4_NoCRC();		
}		
}		

Table 19: Syntax of 2k_ESC0_CRC0()

Syntax	No. of bits	Mnemonic
2k_ESC0_CRC0()		
{		
VUV, 1-0;	2	uimsbf
LSP4, 0;	1	uimsbf
if (VUV!=0) {		
SE_gain, 4-0;	5	uimsbf
LSP1, 4-0;	5	uimsbf
Pitch, 6-1;	6	uimsbf
LSP2, 6;	1	uimsbf
LSP3, 4;	1	uimsbf
LSP2, 5;	1	uimsbf
} else {		
VX_gain1[0], 3-0;	**4**	uimsbf
VX_gain1[1], 3-0;	**4**	uimsbf
LSP1, 4-0;	**5**	uimsbf
LSP2, 6-3;	**4**	uimsbf
LSP3, 4-3;	**2**	uimsbf
}		
if (HVXC_CRC==1) {		
CRC0_2k, 5-0;	6	uimsbf
}		
}		

Table 20: Syntax of 2k_ESC1_CRC1()

Syntax	No. of bits	Mnemonic
2k_ESC1_CRC1()		
{		
if (VUV!=0) {		
SE_shape1, 3-0;	4	uimsbf
} else {		
LSP2, 2-0;	**3**	uimsbf
LSP3, 2;	**1**	uimsbf
}		
if (HVXC_CRC==1) {		
CRC1_2k, 0;	1	uimsbf
}		
}		

Table 21: Syntax of 2k_ESC2_CRC2()

Syntax	No. of bits	Mnemonic
2k_ESC2_CRC2()		
{		
if (VUV!=0) {		
SE_shape2, 3-0;	4	uimsbf
}		
else {		
LSP3, 1-0;	**2**	uimsbf
VX_shape1[0], 5-4;	**2**	uimsbf
}		
if (HVXC_CRC==1) {		
CRC2_2k, 0;	1	uimsbf
}		
}		

Table 22: Syntax of 2k_ESC3_NoCRC()

Syntax	No. of bits	Mnemonic
2k_ESC3_NoCRC()		
{		
if (VUV!=0) {		
LSP2, 4-0;	5	uimsbf
LSP3, 3-0;	4	uimsbf
Pitch, 0;	1	uimsbf
}		
else {		
VX_shape1[0], 3-0;	**4**	uimsbf
VX_shape1[1], 5-0;	**6**	uimsbf
}		
}		

Table 23: Syntax of 4k_ESC0_CRC0()

Syntax	No. of bits	Mnemonic
4k_ESC0_CRC0()		
{		
VUV, 1-0;	**2**	uimsbf
LSP4, 0;	**1**	uimsbf
if (VUV!=0) {		
SE_gain, 4-0;	**5**	uimsbf
LSP1, 4-0;	**5**	uimsbf
Pitch, 6-1;	**6**	uimsbf
LSP2, 6-3;	**4**	uimsbf
SE_shape3, 6-2;	**5**	uimsbf
LSP3, 4;	**1**	uimsbf
LSP5, 7;	**1**	uimsbf
SE_shape4, 9;	**1**	uimsbf
SE_shape5, 8;	**1**	uimsbf
SE_shape6, 5;	**1**	uimsbf
} else {		
VX_gain1[0], 3-0;	**4**	uimsbf
VX_gain1[1], 3-0;	**4**	uimsbf
LSP1, 4-0;	**5**	uimsbf
LSP2, 6-3;	**4**	uimsbf
LSP3, 4;	**1**	uimsbf
LSP5, 7;	**1**	uimsbf
VX_gain2[0], 2-0;	**3**	uimsbf
VX_gain2[1], 2-0;	**3**	uimsbf
VX_gain2[2], 2-0;	**3**	uimsbf
VX_gain2[3], 2-1;	**2**	uimsbf
}		
if (HVXC_CRC==1) {		
CRC0_4k, 5-0;	6	uimsbf
}		
}		

Table 24: Syntax of 4k_ESC1_CRC1()

Syntax	No. of bits	Mnemonic
4k_ESC1_CRC1()		
{		
if (VUV!=0) {		
SE_shape4, 8-0;	**9**	uimsbf
SE_shape5, 7-0;	**8**	uimsbf
SE_shape6, 4-0;	**5**	uimsbf
} else {		
VX_gain2[3], 0;	**1**	uimsbf
LSP2, 2-0;	**3**	uimsbf
LSP3, 3-0;	**4**	uimsbf
LSP5, 6-0;	**7**	uimsbf
VX_shape1[0], 5-0;	**6**	uimsbf
VX_shape1[1], 5;	**1**	uimsbf
}		
if (HVXC_CRC==1) {		
CRC1_4k, 4-0;	5	uimsbf
}		
}		

Table 25: Syntax of 4k_ESC2_CRC2()

Syntax	No. of bits	Mnemonic
4k_ESC2_CRC2()		
{		
if (VUV!=0) {		
SE_shape1, 3-0;	**4**	uimsbf
} else {		
VX_shape1[1], 4-1;	**4**	uimsbf
}		
if (HVXC_CRC==1) {		
CRC2_4k, 0;	1	uimsbf
}		
}		

Table 26: Syntax of 4k_ESC3_CRC3()

Syntax	No. of bits	Mnemonic
4k_ESC3_CRC3()		
{		
if (VUV!=0) {		
SE_shape2, 3-0;	**4**	uimsbf
} else {		
VX_shape1[1], 0;	**1**	uimsbf
VX_shape2[0], 4-2;	**3**	uimsbf
}		
if (HVXC_CRC==1) {		
CRC3_4k, 0;	1	uimsbf
}		
}		

Table 27: Syntax of 4k_ESC4_NoCRC()

Syntax	No. of bits	Mnemonic
4k_ESC4_NoCRC()		
{		
if (VUV!=0) {		
LSP2, 2-0;	**3**	uimsbf
LSP3, 3-0;	**4**	uimsbf
LSP5, 6-0;	**7**	uimsbf
Pitch, 0;	**1**	uimsbf
SE_shape3, 1-0;	**2**	uimsbf
} else {		
VX_shape2[0], 1-0;	**2**	uimsbf
VX_shape2[1], 4-0;	**5**	uimsbf
VX_shape2[2], 4-0;	**5**	uimsbf
VX_shape2[3], 4-0;	**5**	uimsbf
}		
}		

5.5.5 Category interleaving

In order to improve the robustness of the bitstream to channel errors, bit interleaving is carried out where data bits are reordered inside one frame. Bit interleaving is carried out after the bitstream is ordered and the CRC bits are added as shown in tables 18 to 27. The interleaving is done in two steps. First, the HVXC bitstream is divided into two sequences as:

- $x[0..N_1-1]$: sequence consists of bits of ESC0 where N_1 is the number of bits included in ESC0.
- $y[0..N_2-1]$: sequence consists of bits other than ESC0.

The sequence $y[]$ is composed of ESC1, ESC2 ..., in this order. N_1 is the number of bits of ESC0, and N_2 is the number of bits other than ESC0. The total number of bits is $N = N_1 + N_2$. r is defined as:

$$r = \frac{N}{N_1}$$

Next, the sequences $x[]$ and $y[]$ are interleaved into one sequence $z[0..N-1]$. Every single bit from $x[]$ is inserted at the beginning of every $\lfloor r \rfloor - 1$ or $\lfloor r \rfloor$ bits from $y[]$, where $\lfloor r \rfloor$ is the largest integer less than or equal to r. The flowchart is shown in figure 9.

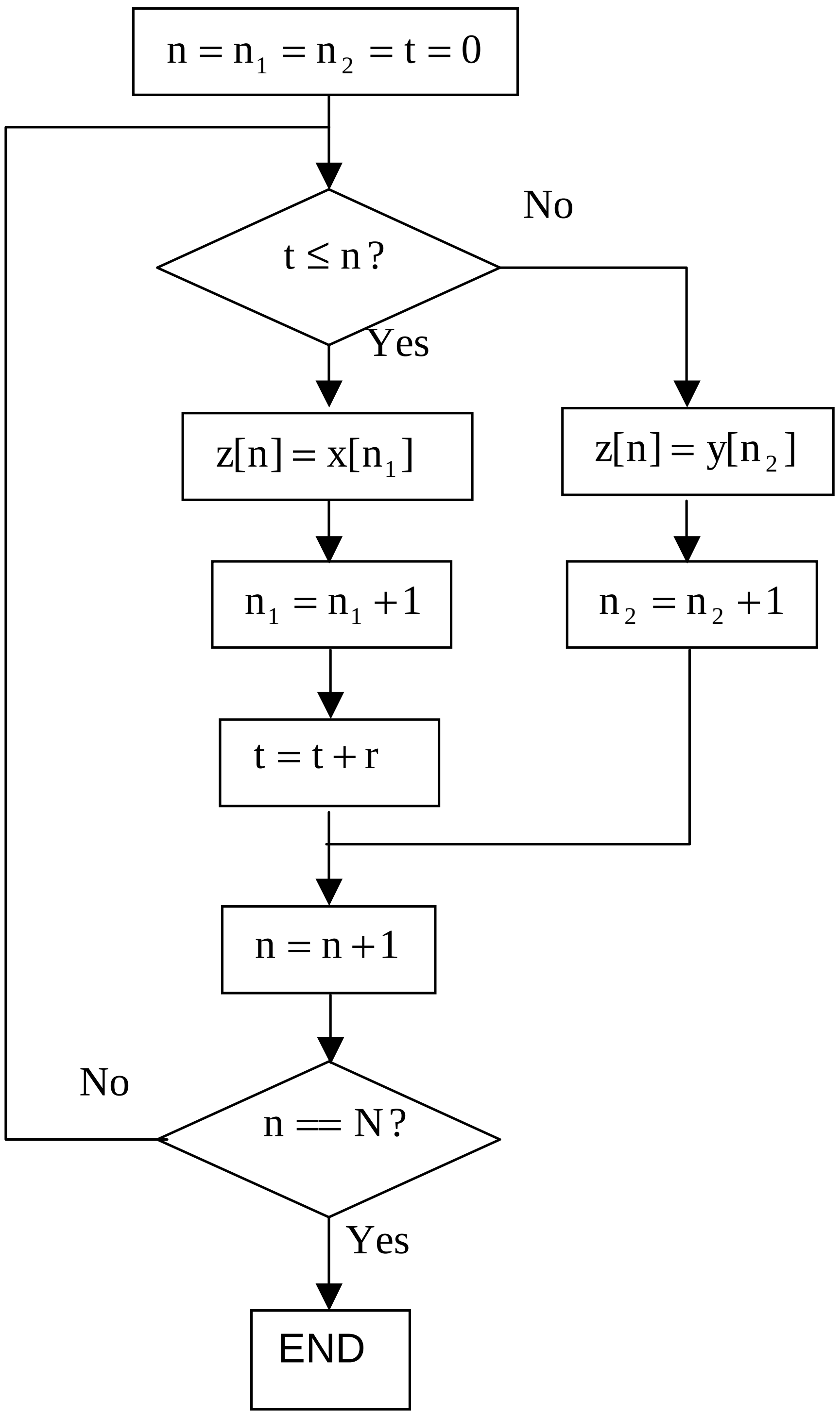


Figure 9: Flowchart of category interleaving method

5.5.6 HVXC error detection and concealment

5.5.6.1 Cyclic Redundancy Check (CRC)

The CRC parity bits are computed from the source bits in the same ESC. A schematic diagram of the CRC checker, the polynomials and initialization procedure is given in annex D.

5.5.6.2 Error concealment

In case a CRC error is detected, error concealment processing (bad frame masking) is carried out in the HVXC decoder. The state transition diagram of the concealment processing is shown in figure 10. A frame masking state of the current frame is updated based on the decoded CRC result of ESC0. If a CRC error is detected in ESC0, the frame is declared to be a "bad" frame. The initial state of the state transition diagram is state = 0. The arrow with a letter "1" denotes the transition for a bad frame, and that with a letter "0" the one for a good frame. At the 2,0 kbit/s rate ESC1 and ESC2 are protected by CRC bits; at the 4,0 kbit/s ESC1 to ESC3 are protected by CRC bits. The results of the CRC checks against these ESCs are used to maintain toll quality as described in detail below.

5.5.6.2.1 Parameter replacement

According to the state value, the following parameter replacement is done. In error free condition, the state value becomes 0, and received source coder bits are used without any concealment processing.

LSP parameters

At state = 1...6, LSP parameters are replaced with those of previous states. At state = 7, if LSP4 = 0 (LSP quantization mode without inter-frame prediction), then LSP parameters are calculated from all LSP indices received in the current frame. If LSP4 = 1 (LSP quantization mode with inter-frame coding), then LSP parameters are calculated with the following method, where LSP parameters belonging to the LSP1 index are interpolated with the previous LSPs.

$$LSP_{base}(n) = p \times LSP_{prev}(n) + (1 - p) \times LSP_{1st}(n) \quad \text{for n = 1...10} \qquad (1)$$

$LSP_{base}(n)$ is LSP parameters of the base layer, $LSP_{prev}(n)$ is the decoded LSPs of the previous frame, $LSP_{1st}(n)$ is the decoded LSPs from the current LSP1 index, and p is the interpolation factor. p is changed according to the number of previous bad frames as shown in table 28. LSP indices LSP2, LSP3 and LSP5 are not used, and $LSP_{base}(n)$ computed according to the Equation (1) is used as current LSP parameters.

Table 28: p factor

frame	p
0	0,7
1	0,6
2	0,5
3	0,4
4	0,3
5	0,2
6	0,1
7	0,0

Mute variable

According to the "state" value, a variable "mute" is set to control the output level of the reproduced speech. The "mute" values in table 29 are used. At state = 7, the average of 1,0 and "mute" value of the previous frame (= 0,5 (1,0 + previous "mute value")) is used. However, when this value is more than 0,8, "mute" value is replaced with 0,8.

Table 29: Mute value

state	mute value
0	1,0
1	0,8
2	0,7
3	0,5
4	0,25
5	0,125
6	0,0
7	average/0,8

Replacement and gain control of "voiced" parameters

At state = 1...6, spectrum parameter SE_shape1, SE_shape2, spectrum gain parameter SE_gain, spectrum parameter for 4,0 kbit/s mode SE_shape3 … SE_shape6 are replaced with the corresponding parameters of the previous frame. Also, to control the volume of the output speech and the harmonic magnitude parameters of LPC residual signal, " $Am[0 \ldots 127]$ " is gain controlled as shown in equation (2). In the equation, $Am_{(org)}[i]$ is computed from the received spectrum parameters of the latest error free frame.

$$Am[i] = mute \times Am_{(org)}[i] \text{ for i = 0...127} \tag{2}$$

If the previous frame is unvoiced and the current state is state = 7, equation (2) is replaced with equation (3);

$$Am[i] = 0{,}6 \times mute \times Am_{(org)}[i] \text{ for i = 0...127} \tag{3}$$

As described before, SE_shape1 and SE_shape2 are individually protected by 1 bit CRCs. ESC1 applies to SE_shape1 and ESC2 applies to SE_shape2 at 2,0 kbit/s. In the same way, ESC2 applies to SE_shape1 and ESC3 applies to SE_shape2 at 4,0 kbit/s. At state = 0 or 7, if both of the CRCs of SE_shape1 and SE_shape2 are in error at the same time, the quantized harmonic magnitudes with fixed dimension $Am_{qnt}[1...44]$ are gain suppressed as:

$$Am_{qnt}[i] = s[i] \times Am_{qnt(org)}[i] \text{ for i = 1...44} \tag{4}$$

$s[i]$ is the factor for the gain suppression. $Am_{qnt(org)}[i]$ is the fixed dimension harmonic magnitudes generated using the SE_shape1 and SE_shape2 which contains bit errors. $Am_{qnt}[i]$ is then dimension converted to obtain $Am[i]$.

Table 30: Factor for gain suppression, s[0...44]

i	1	2	3	4	5	6	7...44
$s[i]$	0,10	0,25	0,40	0,55	0,70	0,85	1,00

At 4,0 kbit/s, SE_shape4, SE_shape5, and SE_shape6 are subject to CRC checking as ESC1. When a CRC error is detected, the spectrum parameter of the enhancement layer is not used.

Replacement and gain control of "unvoiced" parameters

At state = 1...6, the stochastic codebook gain parameters VX_gain1[0] and VX_gain1[1] are replaced with the previous frame's VX_gain1[1]. Also the stochastic codebook gain parameters for the 4,0 kbit/s mode VX_gain2[0]...VX_gain2[3] are replaced with the previous frame's VX_gain2[3].

The stochastic codebook shape parameters VX_shape1[0], VX_shape1[1] and the stochastic codebook shape parameters for the 4,0 kbit/s mode VX_shape2[0], VX_shape2[1], VX_shape2[2] and VX_shape2[3] are generated from randomly generated index values.

Finally, in order to control the volume of the output speech, the LPC residual signal $res[0 \ldots 159]$ is gain controlled as shown in equation (5). $res_{(org)}[i]$ is the excitation signal obtained by using the stochastic codebook gain and the shape parameters generated as explained above.

$$res[i] = mute \times res_{(org)}[i] \quad (0 \le i \le 159) \tag{5}$$

Frame masking state transitions

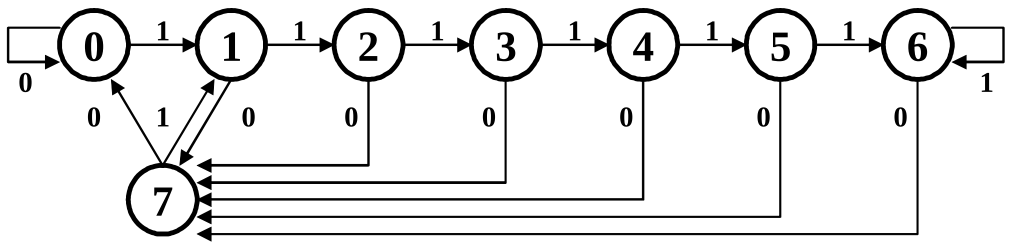


Figure 10: Frame masking state transitions

5.6 SBR

5.6.1 Conceptual overview

Spectral Band Replication (SBR) is an audio coding enhancement method for coding the high frequencies in audio and speech codecs. SBR can increase the bandwidth of conventional codecs to equal or exceed the analogue FM audio bandwidth (15 kHz), see figure 11. This can be achieved within the low bit rates available from DRM. SBR can also improve the performance of narrow-band speech codecs, offering the broadcaster 12 kHz commentary audio bandwidth, used for example for multilingual broadcasting. As most speech codecs are narrowband, SBR is important not only for improving speech quality, but also for improving speech intelligibility and speech comprehension. SBR is mainly a post-process, although some pre-processing is performed in the encoder in order to guide the decoding process.

Annex I contains essential information for implementing SBR.

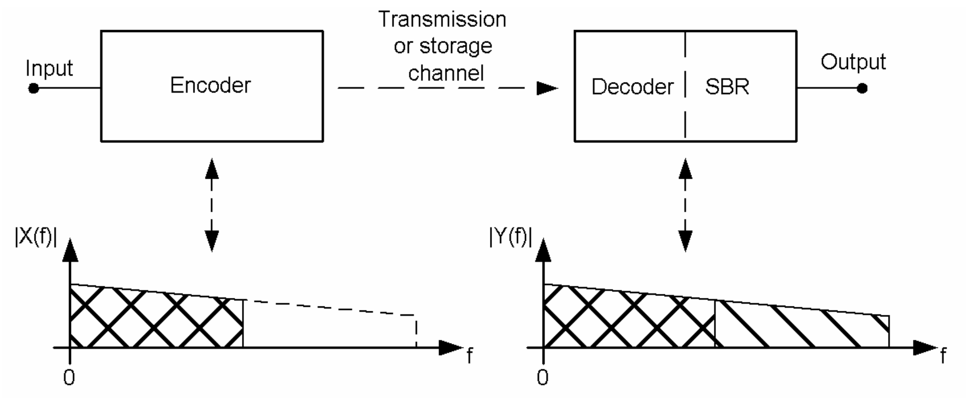


Figure 11: The SBR source coding system

The human voice and most musical instruments generate quasi-stationary excitation signals that emerge from oscillating systems. The wide-band excitation spectrum is created by for example vocal cords, strings or reeds etc and its frequency components form a harmonic series. The harmonic series is filtered by resonators such as the vocal tract, violin body etc., giving the voice or musical instrument its characteristic tone colour or timbre. A bandwidth limitation of such a signal is equivalent to a truncation of the harmonic series, see figure 12. Such a truncation alters the perceived timbre and the audio signal sounds "muffled" or "dull", and the intelligibility may be reduced.

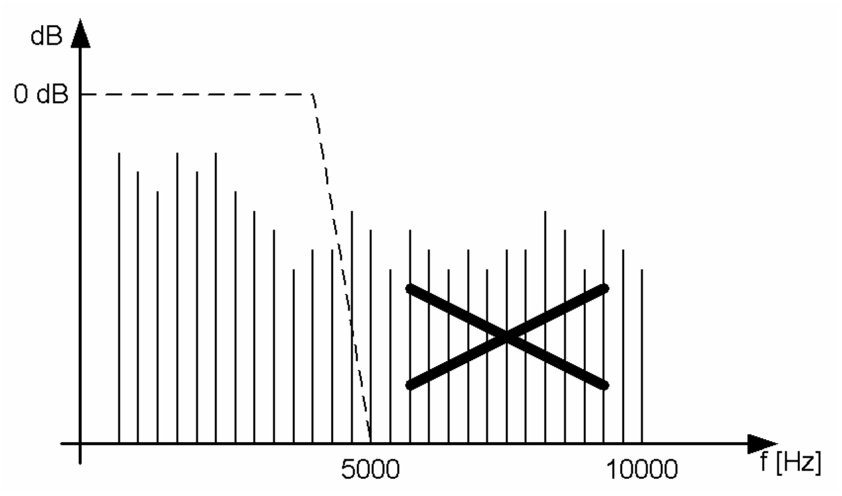


Figure 12: Example of band-limiting of a typical signal

The SBR concept postulates that a truncated harmonic series can be extended based on the relation between lowband and highband spectral components. The spectral envelope, i.e. the coarse spectral distribution, of the replicated highband, must resemble that of the original signal. This is assured by transmitting spectral envelope guidance information from the encoder to decoder at a very low bit rate (approximately 2 kbit/s/channel).

It is important to maintain the ratio between harmonic and noise-like components in the replicated highband. This makes it necessary to selectively add noise components to the SBR replicated high-band.

Two different bit stream protocols are used, one for use with AAC, and one for use with CELP. They are described in the clauses below.

5.6.2 AAC + SBR decoding process

The input to the SBR decoder is the SBR bitstream data and the decoded time domain signal from the AAC decoder, which is sampled at half the rate of the output signal. The decoded low-band time domain signal is fed to an analysis QMF bank, as shown in figure 13. The subband signals obtained from the filter bank are re-routed by a High Frequency generator (HF generator) to reconstruct the high-band. Using the envelope information derived from the transmitted SBR bitstream a gain adjustment within the filter bank scalefactor bands is performed on the high-band subband signals. The adjusted high-band and the delay compensated low-band are then fed to the common synthesis QMF bank which generates the wide-band time domain output signal.

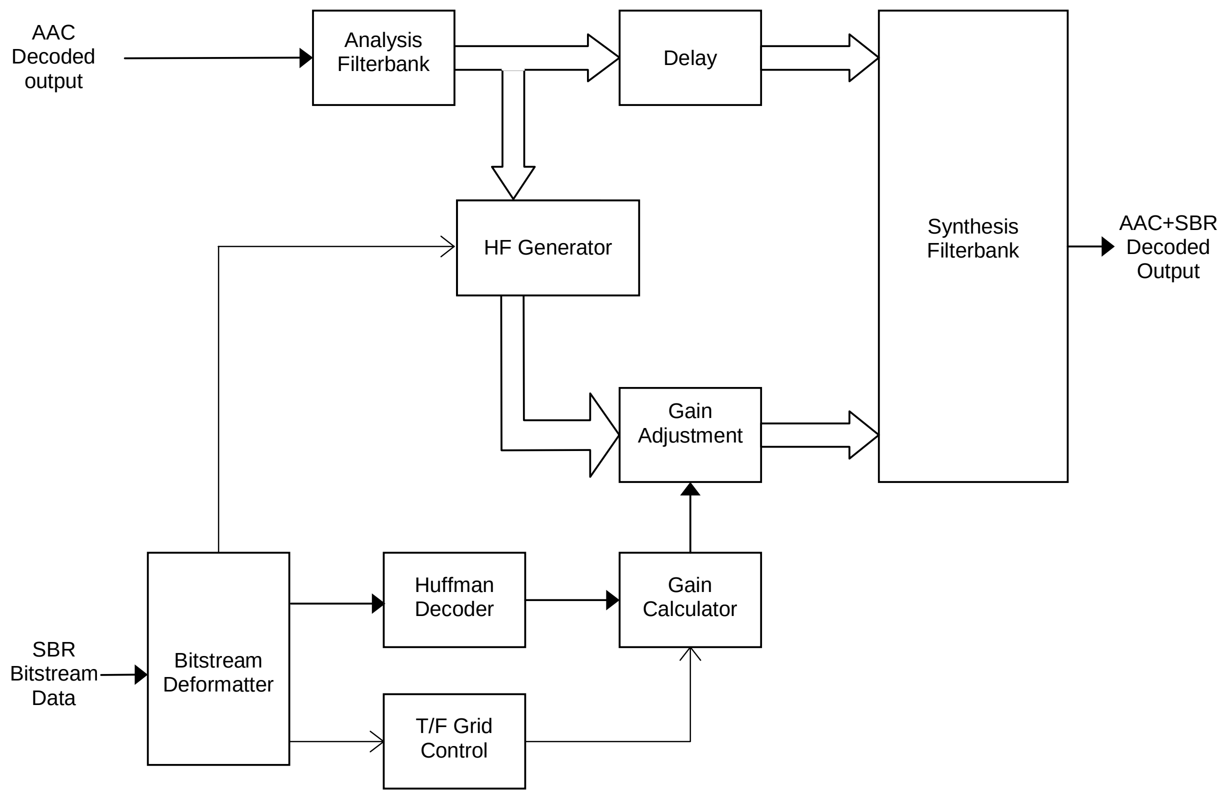


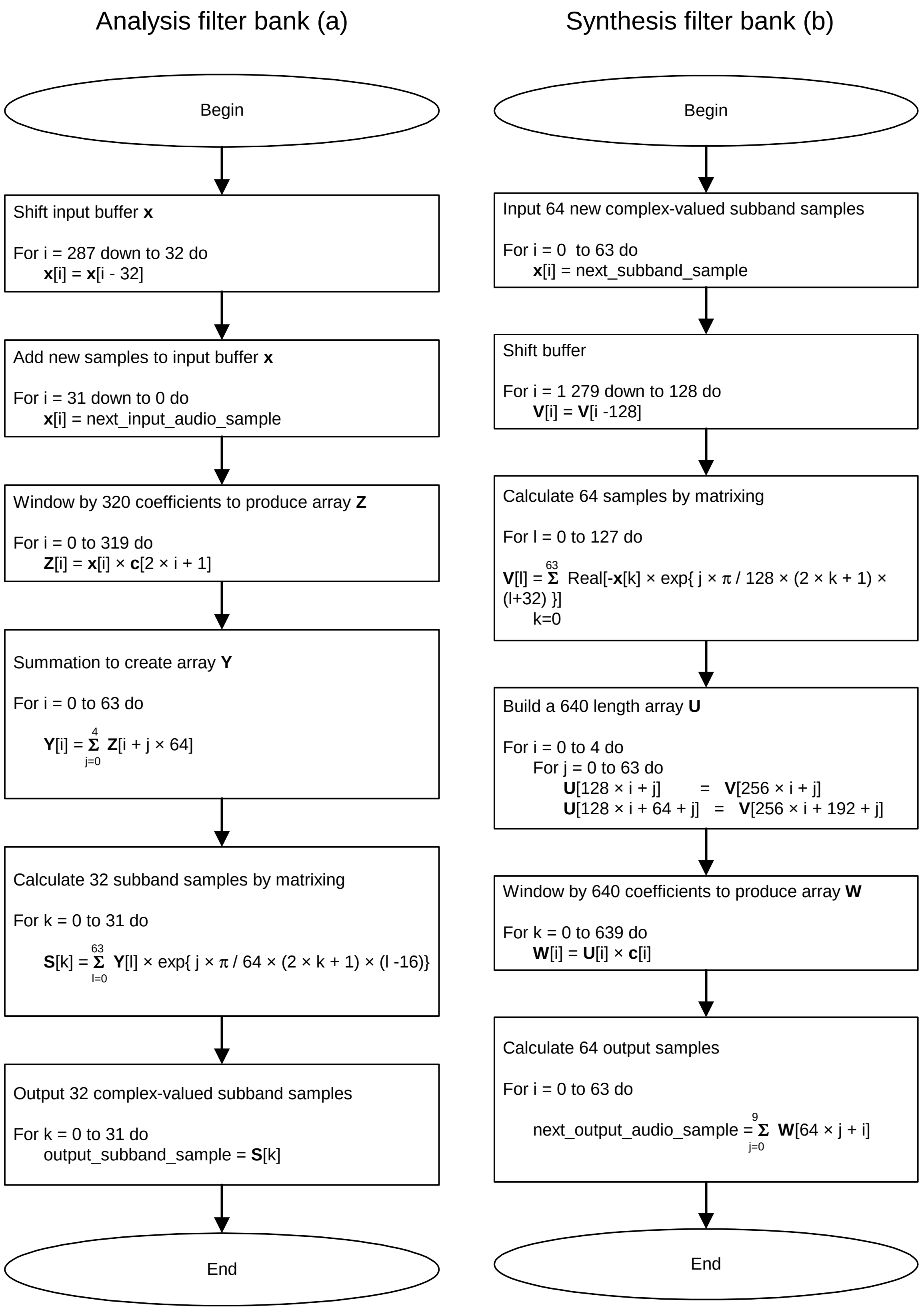
Figure 13: Flow Diagram of the AAC-SBR decoding process

5.6.2.1 Analysis filterbank

Subband filtering of the AAC decoded time domain signal is achieved using a 32-channel QMF bank. The flowchart of figure 14(a) shows the operation. The window coefficients are given in table I.1. For every loop in the flowchart, the output is 32 subband samples, each representing the output from one filterbank channel. For every frame the filterbank produces 30 subband samples for every channel, corresponding to a time domain signal of length 960 samples.

5.6.2.2 Synthesis filterbank

Synthesis filtering of the delayed subband samples and the gain-adjusted subband samples is accomplished using a 64-channel complex QMF bank. The operation is shown in the flowchart of figure 14(b), and the window coefficients are the same as for the analysis bank and tabulated in table I.1. The delayed subband samples are fed to the lowest 32 channels. The real-valued gain-adjusted subband samples are fed to channels corresponding to higher frequencies. The output for every frame is 1 920 time domain samples.


Analysis filter bank (a)

```
Begin

Shift input buffer x
For i = 287 down to 32 do
    x[i] = x[i - 32]

Add new samples to input buffer x
For i = 31 down to 0 do
    x[i] = next_input_audio_sample

Window by 320 coefficients to produce array Z
For i = 0 to 319 do
    Z[i] = x[i] × c[2 × i + 1]

Summation to create array Y
For i = 0 to 63 do
    $Y[i] = \sum_{j=0}^{4} Z[i + j \times 64]$

Calculate 32 subband samples by matrixing
For k = 0 to 31 do
    $S[k] = \sum_{l=0}^{63} Y[l] \times \exp\{ j \times \pi / 64 \times (2 \times k + 1) \times (l - 16)\}$

Output 32 complex-valued subband samples
For k = 0 to 31 do
    output_subband_sample = S[k]

End
```

Synthesis filter bank (b)

```
Begin

Input 64 new complex-valued subband samples
For i = 0  to 63 do
    x[i] = next_subband_sample

Shift buffer
For i = 1 279 down to 128 do
    V[i] = V[i -128]

Calculate 64 samples by matrixing
For l = 0 to 127 do
$V[l] = \sum_{k=0}^{63} \mathrm{Real}[-x[k] \times \exp\{ j \times \pi / 128 \times (2 \times k + 1) \times (l+32) \}]$

Build a 640 length array U
For i = 0 to 4 do
    For j = 0 to 63 do
        U[128 × i + j]        =  V[256 × i + j]
        U[128 × i + 64 + j]   =  V[256 × i + 192 + j]

Window by 640 coefficients to produce array W
For k = 0 to 639 do
    W[i] = U[i] × c[i]

Calculate 64 output samples
For i = 0 to 63 do
    $\mathrm{next\_output\_audio\_sample} = \sum_{j=0}^{9} W[64 \times j + i]$

End
```

Figure 14: Flow chart of the analysis (a) and synthesis (b) filter bank

5.6.2.3 Frequency band tables

The grouping of QMF subband samples in frequency is described by frequency band tables. The tables are defined by functions, most arguments of which are transmitted in the SBR header. For each envelope, there are two frequency band tables; a high frequency resolution table, $\mathbf{f}_{TableHigh}$, and a low frequency resolution table, $\mathbf{f}_{TableLow}$. The noise floor and the limiter also have corresponding frequency band tables, $\mathbf{f}_{TableNoise}$ and $\mathbf{f}_{TableLim}$. All aforementioned tables are derived from one master frequency band table, $\mathbf{f}_{Master}$. The frequency band tables contain the frequency borders for each frequency band, represented as QMF channels. This clause describes how to calculate $\mathbf{f}_{Master}$, $\mathbf{f}_{TableHigh}$, $\mathbf{f}_{TableLow}$ and $\mathbf{f}_{TableNoise}$. The calculation of $\mathbf{f}_{TableLim}$ is described in clause 5.6.2.6.

5.6.2.3.1 Master frequency band table

First the start and stop QMF channels for the master frequency band table are calculated. The start channel, *k0*, is defined by:

$$k0 = startMin + \mathbf{offset}\left(start\_freq\right),$$

where $\mathbf{offset}$ and $startMin$ are given by:

$$\mathbf{offset} = \left[0,1,2,3,4,5,6,7,9,11,13,16,20,24,28,33\right]$$

$$startMin = \begin{cases} NINT\left(3000 \cdot \dfrac{128}{Fs}\right) & , Fs < 32000 \\ NINT\left(4000 \cdot \dfrac{128}{Fs}\right) & , 32000 \le Fs < 64000 \\ NINT\left(5000 \cdot \dfrac{128}{Fs}\right) & , 64000 \le Fs \end{cases}$$

The stop channel, $k2$, is defined by:

$$k2 = \begin{cases} \mathbf{stopVector}\left(stop\_freq\right) & , 0 \le stop\_freq < 14 \\ 2 \cdot k0 & , stop\_freq = 14 \\ 3 \cdot k0 & , stop\_freq = 15 \end{cases}$$

where $stopMin$ and $\mathbf{stopVector}$ are given by:

$$StopMin = \begin{cases} NINT\left(6000 \cdot \dfrac{128}{Fs}\right) & , Fs < 32000 \\ NINT\left(8000 \cdot \dfrac{128}{Fs}\right) & , 32000 \le Fs < 64000 \\ NINT\left(10000 \cdot \dfrac{128}{Fs}\right) & , 64000 \le Fs \end{cases}$$

$$\mathbf{stopVector}\left(k\right) = StopMin + \sum_{i=0}^{k-1} \mathbf{stopDkSort}\left(i\right)$$

and $\mathbf{stopDkSort}$ is given by:

$$\mathbf{stopDkSort} = sort\left(\mathbf{stopDk}\right)$$

$$\mathbf{stopDk}(p) = NINT\left(StopMin \cdot \left(\frac{64}{StopMin} \right)^{\frac{p+1}{13}} \right) - NINT\left(StopMin \cdot \left(\frac{64}{StopMin} \right)^{\frac{p}{13}} \right), 0 \le p \le 12$$

The master frequency band table, $\mathbf{f}_{Master}$, is calculated from *k0, k2*, *freq_scale* and *alter_scale*, which is described in figures 15 and 16 (of course $\mathbf{f}_{Master}$ is only defined for *k2 > k0*.)

5.6.2.3.2 Derived frequency band tables

The envelope high frequency resolution frequency band table, $\mathbf{f}_{TableHigh}$, is obtained by extracting a subset of the borders in $\mathbf{f}_{Master}$ according to:

$$N_{High} = N_{Master} - xover\_band$$

$$N_{Low} = INT\left(\frac{N_{High}}{2} \right) + \left(N_{High} - 2 \cdot INT\left(\frac{N_{High}}{2} \right) \right)$$

$$\mathbf{n} = [N_{Low}, N_{High}]$$

$$\mathbf{f}_{TableHigh}(k) = \mathbf{f}_{Master}(k + xover\_band) \quad , 0 \le k \le N_{High}$$

$$M = \mathbf{f}_{TableHigh}(N_{high}) - \mathbf{f}_{TableHigh}(0)$$

$$lsb = \mathbf{f}_{TableHigh}(0)$$

The envelope low frequency resolution frequency band table, $\mathbf{f}_{TableLow}$, is extracted from $\mathbf{f}_{TableHigh}$ according to:

$$\mathbf{f}_{TableLow}(k) = \mathbf{f}_{TableHigh}(i(k)) \quad , 0 \le k \le N_{Low}$$

where $i(k)$ is defined by:

$$i(k) = \begin{cases} 0 & if\ k = 0 \\ 2 \cdot k - \frac{1-(-1)^{N_{High}}}{2} & if\ k \ne 0 \end{cases}$$

The noise floor frequency band table, $\mathbf{f}_{TableNoise}$, is extracted from $\mathbf{f}_{TableLow}$ according to:

$$\mathbf{f}_{TableNoise}(k) = \mathbf{f}_{TableLow}(i(k)) \quad , 0 \le k \le N_Q$$

where $i(k)$ is defined by:

$$i(k) = \begin{cases} 0 & if\ k = 0 \\ i(k-1) + INT\left(\frac{N_{Low} - i(k-1)}{N_Q + 1 - k} \right) & if\ k \ne 0 \end{cases}$$

and where N_Q is defined below.

$$N_Q = \max\left(1, NINT\left(noise\_bands \cdot \frac{\log\left(\frac{k2}{lsb} \right)}{\log(2)} \right) \right) \quad , 0 \le noise\_bands \le 3$$

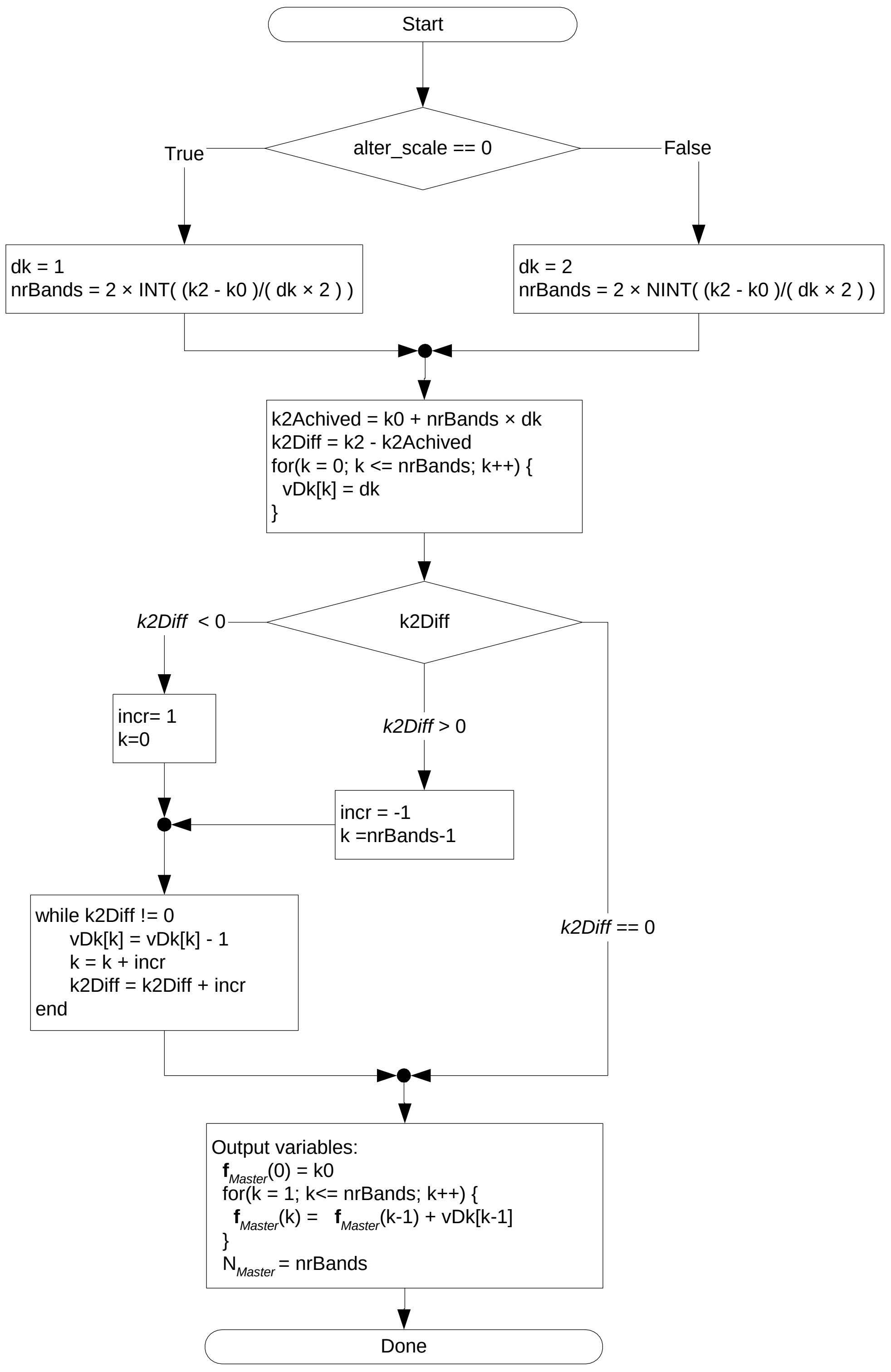

Start
alter_scale == 0
True
False
dk = 1
nrBands = 2 × INT((k2 - k0)/(dk × 2))
dk = 2
nrBands = 2 × NINT((k2 - k0)/(dk × 2))
k2Achived = k0 + nrBands × dk
k2Diff = k2 - k2Achived
for(k = 0; k <= nrBands; k++) {
vDk[k] = dk
}
k2Diff
k2Diff < 0
k2Diff > 0
k2Diff == 0
incr= 1
k=0
incr = -1
k =nrBands-1
while k2Diff != 0
vDk[k] = vDk[k] - 1
k = k + incr
k2Diff = k2Diff + incr
end
Output variables:
$\mathbf{f}_{Master}(0) = k0$
for(k = 1; k<= nrBands; k++) {
$\mathbf{f}_{Master}(k) = \mathbf{f}_{Master}(k-1) + vDk[k-1]$
}
N_{Master} = nrBands
Done

Figure 15: Flowchart calculation of f_{Master} when freq_scale = 0

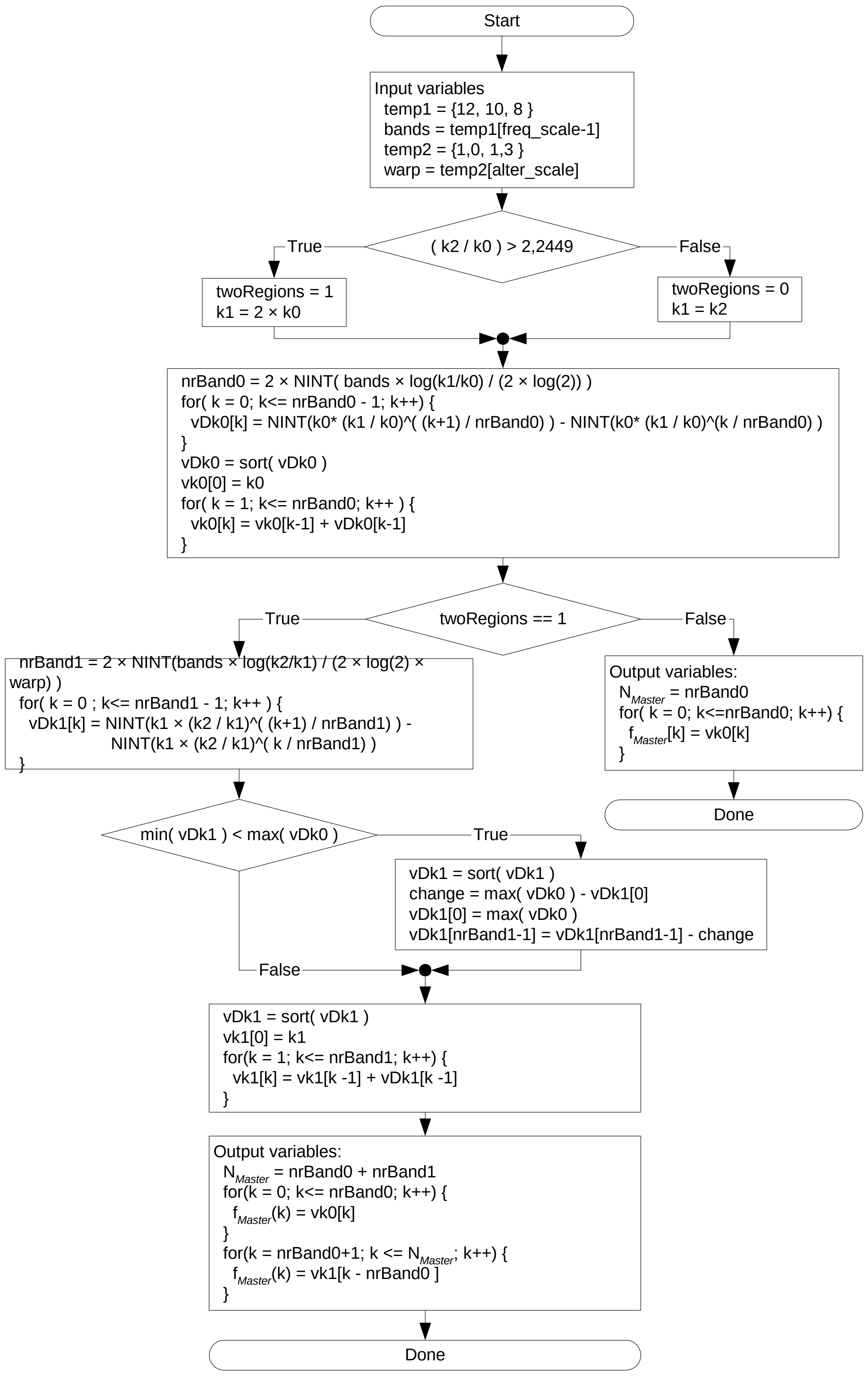

Start
Input variables
temp1 = {12, 10, 8 }
bands = temp1[freq_scale-1]
temp2 = {1,0, 1,3 }
warp = temp2[alter_scale]
(k2 / k0) > 2,2449
True
False
twoRegions = 1
k1 = 2 × k0
twoRegions = 0
k1 = k2
nrBand0 = 2 × NINT(bands × log(k1/k0) / (2 × log(2)))
for(k = 0; k<= nrBand0 - 1; k++) {
vDk0[k] = NINT(k0* (k1 / k0)^((k+1) / nrBand0)) - NINT(k0* (k1 / k0)^(k / nrBand0))
}
vDk0 = sort(vDk0)
vk0[0] = k0
for(k = 1; k<= nrBand0; k++) {
vk0[k] = vk0[k-1] + vDk0[k-1]
}
twoRegions == 1
True
False
nrBand1 = 2 × NINT(bands × log(k2/k1) / (2 × log(2) × warp))
for(k = 0 ; k<= nrBand1 - 1; k++) {
vDk1[k] = NINT(k1 × (k2 / k1)^((k+1) / nrBand1)) -
NINT(k1 × (k2 / k1)^(k / nrBand1))
}
Output variables:
N_Master = nrBand0
for(k = 0; k<=nrBand0; k++) {
f_Master[k] = vk0[k]
}
Done
min(vDk1) < max(vDk0)
True
vDk1 = sort(vDk1)
change = max(vDk0) - vDk1[0]
vDk1[0] = max(vDk0)
vDk1[nrBand1-1] = vDk1[nrBand1-1] - change
False
vDk1 = sort(vDk1)
vk1[0] = k1
for(k = 1; k<= nrBand1; k++) {
vk1[k] = vk1[k -1] + vDk1[k -1]
}
Output variables:
N_Master = nrBand0 + nrBand1
for(k = 0; k<= nrBand0; k++) {
f_Master(k) = vk0[k]
}
for(k = nrBand0+1; k <= N_Master; k++) {
f_Master(k) = vk1[k - nrBand0]
}
Done

Figure 16: Flowchart calculation of f_{Master} when freq_scale > 0

5.6.2.4 T/F grid control

The time/frequency grid part of the bitstream, decoded by *sbr_grid*(), describes the number of envelopes and noise floors as well as the time segment associated with each envelope and noise floor. Furthermore, it describes what frequency band table to use for each envelope. Four different frame classes, FIXFIX, FIXVAR, VARFIX and VARVAR, are used, each of which has different capabilities with respect to time/frequency grid selection. The names refer to whether the locations of the leading and trailing frame borders (i.e. the frame boundaries) are variable or not. The envelope and noise floor time segments are described by the vectors, $\mathbf{t}_E(l)$ and $\mathbf{t}_Q(l)$ respectively, which contain the borders for each time segment expressed in time slots. The calculation of $\mathbf{t}_E(l)$ is described below.

First the leading frame border, *absBordLead*, and the trailing frame border, *absBorderTrail*, are obtained from the bitstream data according to:

$$absBordLead = \begin{cases} 0 & , frame\_class = FIXFIX \ or \ FIXVAR \\ abs\_bord & , frame\_class = VARFIX \\ abs\_bord\_0 & , frame\_class = VARVAR \end{cases}$$

$$absBordTrail = \begin{cases} NO\_TIME\_SLOTS & , frame\_class = FIXFIX \ or \ VARFIX \\ abs\_bord & , frame\_class = FIXVAR \\ abs\_bord\_1 & , frame\_class = VARVAR \end{cases}$$

In order to decode the time borders of all envelopes within the frame, the number of relative borders associated with the leading and trailing time borders respectively are calculated according to:

$$n_{RelLead} = \begin{cases} L_E - 1 & , frame\_class = FIXFIX \\ 0 & , frame\_class = FIXVAR \\ num\_rel & , frame\_class = VARFIX \\ num\_rel\_0 & , frame\_class = VARVAR \end{cases}$$

$$n_{RelTrail} = \begin{cases} 0 & , frame\_class = FIXFIX \ or \ VARFIX \\ num\_rel & , frame\_class = FIXVAR \\ num\_rel\_1 & , frame\_class = VARVAR \end{cases}$$

where

$L_E = \text{num\_env}$

The envelope time border vector, $\mathbf{t}_E(l)$, of the current SBR-frame is then calculated as shown below.

$$\mathbf{t}_E(l) = \begin{cases} absBordLead & if \ l = 0 \\ absBordTrail & if \ l = L_E \\ absBordLead + \sum_{i=0}^{l-1} \mathbf{relBordLead}(i) & if \ 1 \le l \le n_{RelLead} \\ absBordTrail - \sum_{i=0}^{L_E - l - 1} \mathbf{relBordTrail}(i) & if \ n_{RelTrail} < l < L_E \end{cases}$$

where $0 \le l \le L_E$ and $\mathbf{relBordLead}(l)$ and $\mathbf{relBordTrail}(l)$ are vectors containing the relative borders associated with the leading and trailing borders respectively. Both vectors are (if applicable) defined below.

$$\mathbf{relBordLead}(l) = \begin{cases} \dfrac{NO\_TIME\_SLOTS}{L_E} & , frame\_class = FIXFIX \\ NA & , frame\_class = FIXVAR \\ \mathbf{rel\_bord}(l) & , frame\_class = VARFIX \\ \mathbf{rel\_bord\_0}(l) & , frame\_class = VARVAR \end{cases}$$

where $0 \le l < n_{RelLead}$.

$$\mathbf{relBordTrail}(l) = \begin{cases} NA & , frame\_class = FIXFIX \ or \ VARFIX \\ \mathbf{rel\_bord}(l) & , frame\_class = FIXVAR \\ \mathbf{rel\_bord\_1}(l) & , frame\_class = VARVAR \end{cases}$$

where $0 \le l < n_{RelTrail}$.

Within one frame there can either be one or two noise floors. The noise floor time borders are derived from the envelope time border vector according to:

$L_Q = \text{num\_noise}$

$$\mathbf{t}_Q = \begin{cases} \left[\mathbf{t}_E(0), \mathbf{t}_E(1)\right] & , L_E = 1 \\ \left[\mathbf{t}_E(0), \mathbf{t}_E(middleBorder), \mathbf{t}_E(L_E)\right] & , L_E > 1 \end{cases}$$

where *middleBorder* = *func*(*frame_class, pointer,* L_E) is calculated according to table 31.

Table 31: *middleBorder* function

frame_class pointer	$FIXFIX$	$VARFIX$	$FIXVAR, VARVAR$
$=0$	$\frac{L_E}{2}$	1	$L_E - 1$
=1	$\frac{L_E}{2}$	$L_E - 1$	$L_E - 1$
>1	$\frac{L_E}{2}$	$pointer - 1$	$L_E + 1 - pointer$

Each envelope can be of either high or low frequency resolution. This is described by an envelope frequency resolution vector, **f**(*l*), which is calculated according to:

$$\mathbf{f}(l) = \mathbf{freq\_res}(l) \quad , 0 \le l < L_E$$

where

$\mathbf{f}(l) = 0$ signals the usage of $\mathbf{f}_{TableHigh}$ for envelope *l*

$\mathbf{f}(l) = 1$ signals the usage of $\mathbf{f}_{TableLow}$ for envelope *l*

5.6.2.5 Huffman decoder

The envelope and noise samples are quantized and delta coded, either in the frequency or time direction, and the direction chosen is signalled to the decoder. The delta values are Huffman coded, and the codewords are transmitted to the decoder. Obviously, the task of the Huffman decoder module is to reconvert the codewords into the quantized envelope and noise samples generated at the encoder.

5.6.2.5.1 Envelope and noise floor decoding

Delta coding is done in either time or frequency direction for each envelope. When delta coding in the time direction across frame boundaries is applied, the first envelope in the current frame is delta coded with respect to the last envelope of the previous frame.

How to extract the envelope data $\mathbf{E}$ is shown below.

$$\mathbf{E}(k,l)=\begin{cases}\sum_{i=0}^{k}\mathbf{E}_{Delta}(i,l) & ,\begin{cases}0\le l<L_E\\ 0\le k<\mathbf{n}(\mathbf{f}(l))\\ \mathbf{dt\_env}(l)=0\end{cases}\\ g_E(k,l)+\mathbf{E}_{Delta}(k,l) & ,\begin{cases}0\le l<L_E\\ 0\le k<\mathbf{n}(\mathbf{f}(l))\\ \mathbf{dt\_env}(l)=1\\ \mathbf{f}(l)=g(l)\end{cases}\\ g_E(i(k),l)+\mathbf{E}_{Delta}(k,l) & ,\begin{cases}0\le l<L_E\\ 0\le k<\mathbf{n}(\mathbf{f}(l))\\ \mathbf{dt\_env}(l)=1\\ \mathbf{f}(l)=0\\ g(l)=1\\ i(k)\text{ is defined by}\\ \mathbf{f}_{TableHigh}(i(k))=\mathbf{f}_{TableLow}(k)\end{cases}\\ g_E(i(k),l)+\mathbf{E}_{Delta}(k,l) & ,\begin{cases}0\le l<L_E\\ 0\le k<\mathbf{n}(\mathbf{f}(l))\\ \mathbf{dt\_env}(l)=1\\ \mathbf{f}(l)=1\\ g(l)=0\\ i(k)\text{ is defined by}\\ \mathbf{f}_{TableLow}(i(k))\le\mathbf{f}_{TableHigh}(k)<\mathbf{f}_{TableLow}(i(k)+1)\end{cases}\end{cases}$$

Where $g_E(k,l)$ and $g(l)$ is defined below and $\mathbf{E}_{Delta}(k,l)$ is read from the bitstream element *data_env* as shown below. As $\mathbf{E}$ represents the envelope scalefactors for the current frame, the envelope scalefactors from the previous frame is denoted $\mathbf{E}'$. Envelope scalefactors from the previous frame, $\mathbf{E}'$ is needed when delta coding in the time direction over frame boundaries. The number of envelopes of the previous frame is denoted L'_E, and is also needed in that case, as well as the frequency resolution vector of the previous frame, denoted $\mathbf{f}'$.

$$g_E(k,l)=\begin{cases}\mathbf{E}(k,l-1) & ,\begin{cases}1\le l<L_E\\ 0\le k<\mathbf{n}(\mathbf{f}(l))\end{cases}\\ \mathbf{E}'(k,L'_E-1) & ,\begin{cases}l=0\\ 0\le k<\mathbf{n}(\mathbf{f}(l))\end{cases}\end{cases}\text{ and } g(l)=\begin{cases}\mathbf{f}(l-1) & ,1\le l<L_E\\ \mathbf{f}'(L'_E-1) & ,l=0\end{cases}$$

$$\mathbf{E}_{Delta}(k,l)=data\_env[ch][l][k]\quad ,\begin{cases}ch\text{ is the current channel}\\ 0\le l<L_E\\ 0\le k<\mathbf{n}(\mathbf{f}(l))\end{cases}$$

How to extract the noise floor scalefactors from the bitstream is shown below.

$$\mathbf{Q}(k,l)=\begin{cases}\sum_{i=0}^{k}\mathbf{Q}_{Delta}(i,l) & ,\begin{cases}0\le l<L_Q\\0\le k<N_Q\\\mathbf{dt\_noise}(l)=0\end{cases}\\\mathbf{Q}(k,l-1)+\mathbf{Q}_{Delta}(k,l) & ,\begin{cases}1\le l<L_Q\\0\le k<N_Q\\\mathbf{dt\_noise}(l)=1\end{cases}\\\mathbf{Q}'(k,L'_Q-1)+\mathbf{Q}_{Delta}(k,0) & ,\begin{cases}l=0\\0\le k<N_Q\\\mathbf{dt\_noise}(0)=1\end{cases}\end{cases}$$

Where $\mathbf{Q}'$ is the noise floor scalefactors from the previous frame and L'_Q is the number of noise floors from the previous frame. $\mathbf{Q}_{Delta}(k,l)$ is read from the bitstream element *data_noise* as shown below.

$$\mathbf{Q}_{Delta}(k,l)=data\_noise[ch][l][k]\quad,\begin{cases}ch\text{ is the current channel}\\0\le l<L_Q\\0\le k<N_Q\end{cases}$$

5.6.2.5.2 Dequantization and stereo decoding

For the quantization of the envelope scalefactors, there are two quantization steps available. *amp_res* = 0 corresponds to a quantization step of 1,5 dB and *amp_res* = 1 corresponds to a quantization step of 3,0 dB.

For a single channel element, the envelope scalefactors should be decoded according to below.

$$\mathbf{E}_{Orig}(k,l)=64\cdot2^{\mathbf{E}(k,l)\cdot amp}\quad,\begin{cases}0\le l<L_E\\0\le k<\mathbf{n}(\mathbf{f}(l))\end{cases}$$

where $amp=\begin{cases}0.5 & ,amp\_res=0\\1 & ,amp\_res=1\end{cases}$

$$\mathbf{Q}_{Orig}(k,l)=2^{NOISE\_FLOOR\_OFFSET-\mathbf{Q}(k,l)}\quad,\begin{cases}0\le l<L_Q\\0\le k<N_Q\end{cases}$$

For a channel pair element where coupling mode is not used (i.e. *coupling* = 0), the individual channels are treated as the single channel element case above. If coupling mode is used (i.e. *coupling* = 1), the time-grids $\mathbf{t}_E$ and $\mathbf{t}_Q$ are the same for both channels.

Let $\mathbf{E}_0$, $\mathbf{E}_1$, $\mathbf{Q}_0$ and $\mathbf{Q}_1$ represent the decoded envelope scalefactors and noise floor scalefactors, in accordance with the decoding process outlined above. Subscript zero represents the first decoded channel (the energy average and the noise-floor average of the original left and right channel) and subscript 1 represents the secondly decoded channel (the energy ratio and the noise-floor ratio of the original left and right channel).

Below it is shown how to dequantize the envelope and noise floor scalefactors in coupling mode

(*coupling* = 1).

$$\mathbf{E}_{LeftOrig}(k,l)=2^{\mathbf{E}_1(k,l)\cdot amp}\cdot UN\_MAP\cdot\frac{2\cdot64\cdot2^{\mathbf{E}_0(k,l)\cdot amp}}{1+UN\_MAP\cdot2^{\mathbf{E}_1(k,l)\cdot amp}}(k,l),\begin{cases}0\le l<L_E\\0\le k<\mathbf{n}(\mathbf{f}(l))\end{cases}$$

$$\mathbf{E}_{RightOrig}(k,l)=\frac{2\cdot64\cdot2^{\mathbf{E}_0(k,l)\cdot amp}}{1+UN\_MAP\cdot2^{\mathbf{E}_1(k,l)\cdot amp}},\begin{cases}0\le l<L_E\\0\le k<\mathbf{n}(\mathbf{f}(l))\end{cases}$$

$$\mathbf{Q}_{RightOrig}(k,l) = \frac{1+2^{\mathbf{Q}_1(k,l)-\mathbf{panOffset}(0)}}{2^{\mathbf{Q}_0(k,l)-NOISE\_FLOOR\_OFFSET+1}}, \begin{cases} 0 \le l < L_Q \\ 0 \le k < N_Q \end{cases}$$

$$\mathbf{Q}_{LeftOrig}(k,l) = \frac{1+2^{\mathbf{Q}_1(k,l)-\mathbf{panOffset}(0)}}{2^{\mathbf{Q}_0(k,l)-NOISE\_FLOOR\_OFFSET+1+\mathbf{Q}_1(k,l)-\mathbf{panOffset}(0)}}, \begin{cases} 0 \le l < L_Q \\ 0 \le k < N_Q \end{cases}$$

5.6.2.6 HF generator

The objective of the HF generator is to patch a number of subband signals obtained from the analysis filterbank from consecutive channels of matrix $\mathbf{X}_{Low}$ to consecutive channels of matrix $\mathbf{X}_{High}$. The subband signals $\mathbf{X}_{High}$ are subsequently inverse filtered according to the inverse filtering level signalled from the encoder. The HF generator module is also responsible for the construction of the limiter frequency tables.

The analysis filter bank splits the AAC-decoded signal $x(n)$ into 32 subband signals. Assume that a decoded signal, with sampling rate F_{sAAC}, has a bandwidth up to frequency F_c. The subband signals $\mathbf{X}_{Low}(k,n)$, k = 0 to 31, are complex-valued, each having a sampling rate $F_{sAAC}/32$.

The SBR start channel, denoted *startBand*, is in a general sense determined by:

$$startBand = INT\left(64 \cdot \frac{F_c}{F_{sAAC}}\right)$$

However, in the decoder, this value is resolved from bitstream signals. The number of patched channels is denoted *patchNoSubbands* and the highband subband signals are denoted $\mathbf{X}_{High}(k,n)$, k = 0 to 63. HF generation is defined as the process of patching, or copying, subband signals as:

$$X_{High}(startBand + k,\ n) = X_{Low}(startBand - patchNoSubbands - P + k,\ n)$$

where $0 \le k < patchNoSubbands$, $(-1)^{patchNoSubbands + P} = 1$, i.e. *patchNoSubbands +P* is an even number and *P* is an integer offset within the interval [0, *startBand – patchNoSubbands*]. This operation is repeated with different values of *startBand*, *patchNoSubbands* and *P* until the intended amount of bandwidth extension is attained.

The inverse filtering is done in two steps. Linear prediction is first performed on the subband signals of $\mathbf{X}_{Low}$. Then the actual inverse filtering is done independently for each of the subband signals patched to $\mathbf{X}_{High}$ by the HF generator. The subband signals are complex valued, which results in complex filter coefficients for the linear prediction as well as for the inverse filtering. The prediction filter coefficients are obtained from the covariance method. The covariance matrix elements calculated are:

$$\phi_k(i,j) = \sum_{n=0}^{31} \mathbf{X}_{Low}(k, n-i+t_{HfGen}) \cdot \mathbf{X}_{Low}{}^{*}(k, n-j+t_{HfGen}) \quad , \begin{cases} 0 \le i < 3 \\ 1 \le j < 3 \\ 0 \le k < k0 \end{cases}$$

The coefficients $\alpha_0(k)$ and $\alpha_1(k)$ used to filter the subband signal are calculated as:

$$\mathrm{d}(k) = \phi_k(2,2) \cdot \phi_k(1,1) - \frac{1}{1+\varepsilon_{Inv}} \left|\phi_k(1,2)\right|^2$$

$$\alpha_1(k) = \begin{cases} \dfrac{\phi_k(0,1)\cdot\phi_k(1,2) - \phi_k(0,2)\cdot\phi_k(1,1)}{d(k)} & , d(k) \ne 0 \\ 0 & , d(k) = 0 \end{cases}$$

and

$$\alpha_0(k) = \begin{cases} -\dfrac{\phi_k(0,1) + \alpha_1(k) \cdot \phi_k^*(1,2)}{\phi_k(1,1)} & , \phi_k(1,1) \neq 0 \\ 0 & , \phi_k(1,1) = 0 \end{cases}$$

In the first formula above ε is the relaxation parameter (ε_{Inv} = 1E-6). Moreover, if either of the magnitudes of $\alpha_0(k)$ and $\alpha_1(k)$ is greater than or equal to 4, both coefficients are set to zero.

The calculation of the chirp factors, **bwFac,** is shown below. Each chirp factor is used within a specific frequency range defined by the noise floor table, $\mathbf{f}_{TableNoise}$.

$$\mathbf{bwFac} = \begin{cases} 0 & if\ \mathbf{tempBw} < 0.015625 \\ \mathbf{tempBw} & if\ \mathbf{tempBw} \geq 0.015625 \end{cases}, \quad 0 \leq i < N_Q$$

where **tempBw** is calculated as:

$$\mathbf{tempBw} = \begin{cases} 0.75000 \cdot newBw + 0.25000 \cdot \mathbf{bwFac}' & if\ newBw < \mathbf{bwFac}' \\ 0.90625 \cdot newBw + 0.09375 \cdot \mathbf{bwFac}' & if\ newBw \geq \mathbf{bwFac}' \end{cases}, 0 \leq i < N_Q$$

bwFac' is the **bwFac** values calculated in the previous frame, and are assumed to be zero for the first frame. *newBw* is a function of **invf_mode**(i) and **invf_mode´** (i), where **invf_mode´** are the **invf_mode** values from the previous frame.

The patch is built in accordance to the flowchart of figure 17, where the output variable *noPatches* is an integer value specifying the number of patches. **patchStartSubband** and **patchNoSubbands** are vectors holding the data output from the patch decision algorithm.

The HF generation is obtained according to:

$$\begin{aligned} \mathbf{X}_{High}\left(k, l + t_{HfGen}\right) = \; & \mathbf{X}_{Low}\left(p, l + t_{HfGen}\right) + \mathbf{bwFac} \cdot \alpha_0(p) \cdot \mathbf{X}_{Low}\left(p, l - 1 + t_{HfGen}\right) + \\ & + [\mathbf{bwFac}]^2 \cdot \alpha_1(p) \cdot \mathbf{X}_{Low}\left(p, l - 2 + t_{HfGen}\right) \end{aligned}$$

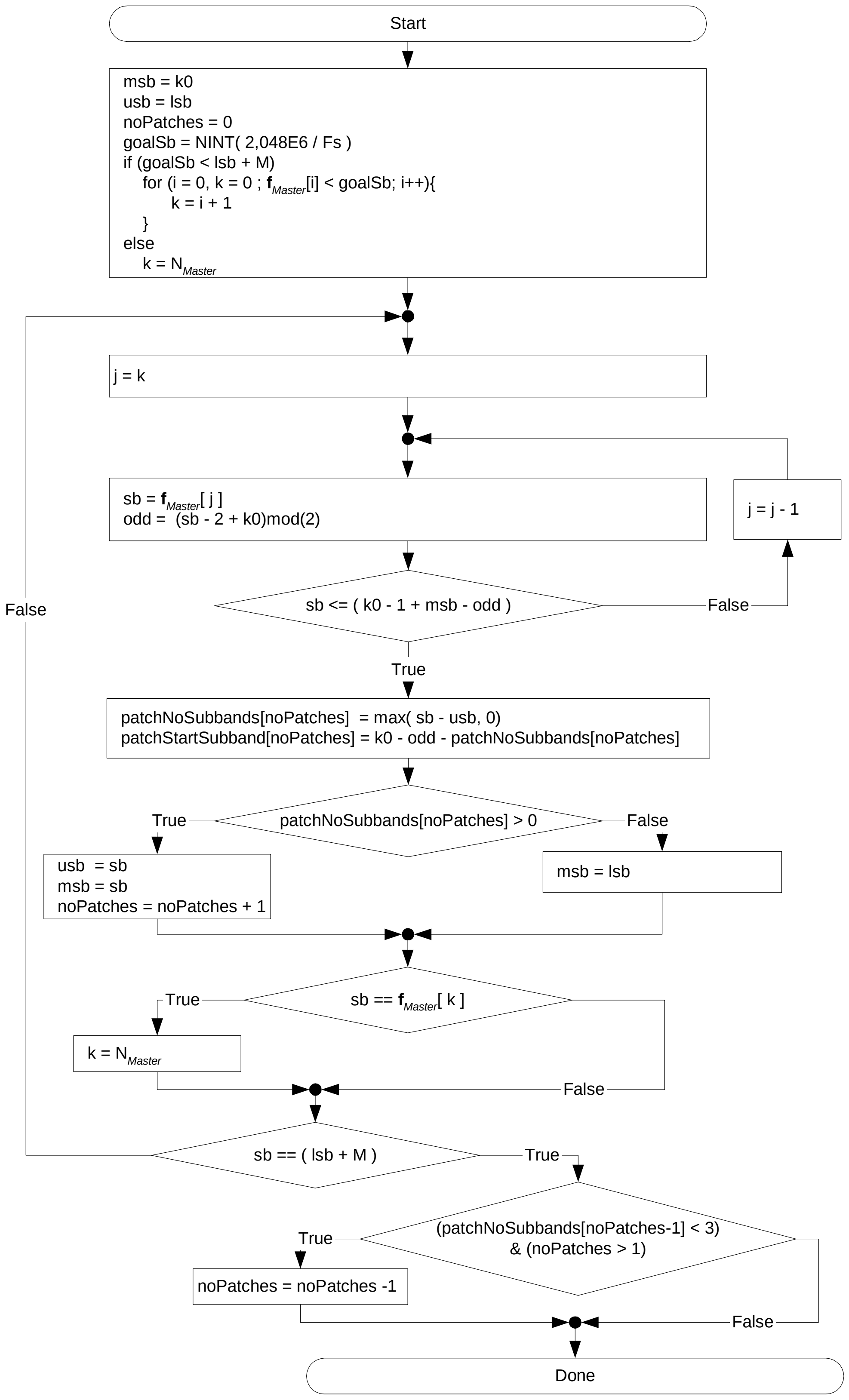

Start
msb = k0
usb = lsb
noPatches = 0
goalSb = NINT(2,048E6 / Fs)
if (goalSb < lsb + M)
for (i = 0, k = 0 ; fMaster[i] < goalSb; i++){
k = i + 1
}
else
k = NMaster
j = k
sb = fMaster[j]
odd = (sb - 2 + k0)mod(2)
j = j - 1
sb <= (k0 - 1 + msb - odd)
False
True
False
patchNoSubbands[noPatches] = max(sb - usb, 0)
patchStartSubband[noPatches] = k0 - odd - patchNoSubbands[noPatches]
patchNoSubbands[noPatches] > 0
True
False
usb = sb
msb = sb
noPatches = noPatches + 1
msb = lsb
sb == fMaster[k]
True
k = NMaster
False
sb == (lsb + M)
True
(patchNoSubbands[noPatches-1] < 3)
& (noPatches > 1)
True
noPatches = noPatches -1
False
Done

Figure 17: Flowchart of patch construction

5.6.2.6.1 Limiter frequency band table

The limiter frequency band table, $\mathbf{f}_{TableLim}$ is constructed to have either exactly one limiter band over the entire SBR range, or approximately 1,2, 2 or 3 bands per octave, decided by **limiter_bands** from the bitstream. The table holds indices of the synthesis filterbank channels, where the number of elements equals the number of bands plus one. The first element is always *lsb*. $\mathbf{f}_{TableLim}$ is a subset of the union of $\mathbf{f}_{TableLow}$ and the patch borders.

If **limiter_bands** is zero only one limiter band is used and $\mathbf{f}_{TableLim}$ is created as:

$$\mathbf{f}_{TableLim} = \left[\mathbf{f}_{TableLow}(0), \mathbf{f}_{TableLow}(N_{Low})\right]$$
$$N_L = 1$$

If **limiter_bands** > 0 the limiter frequency band table is created according to the flowchart of figure 18.

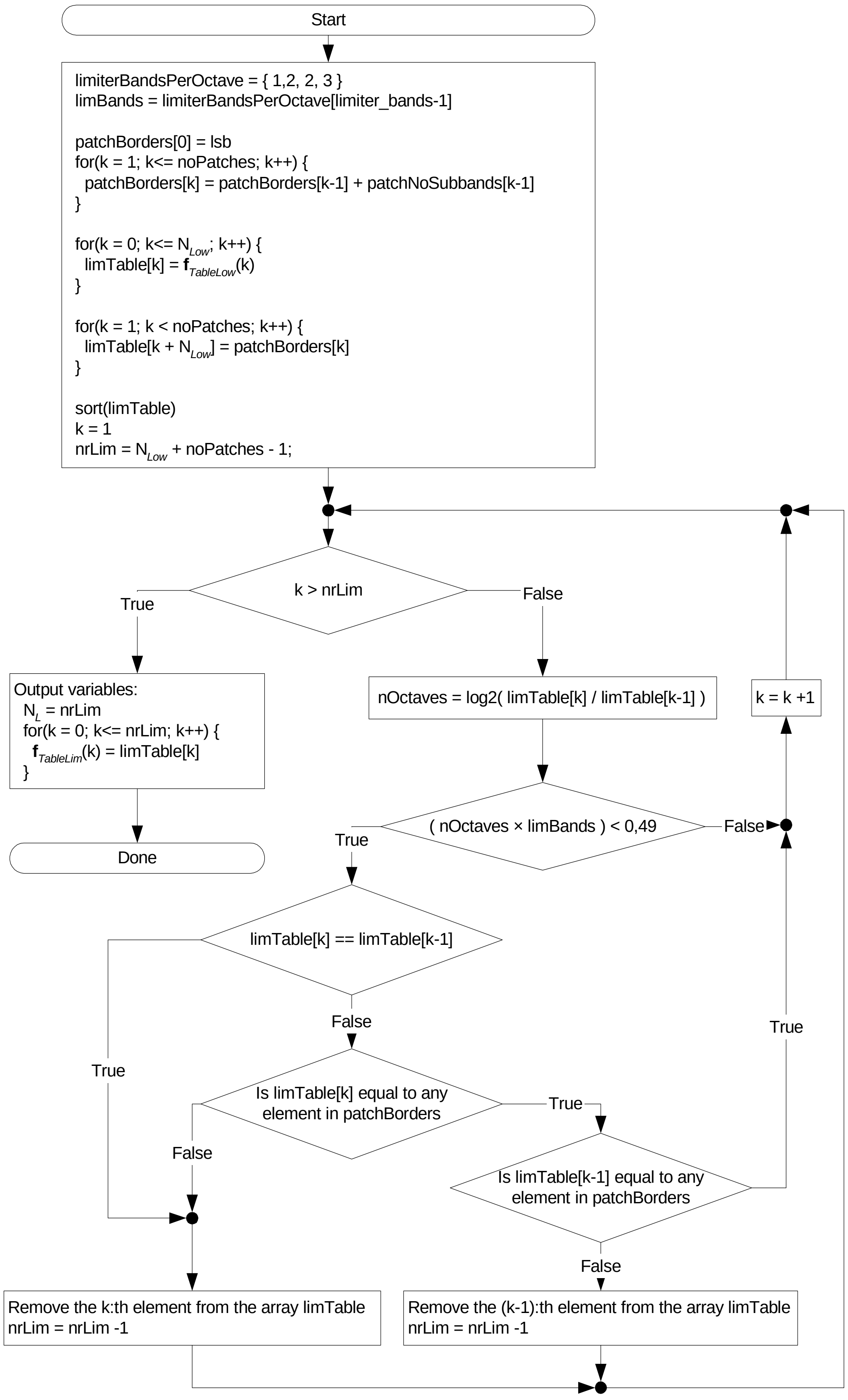

Start
limiterBandsPerOctave = { 1,2, 2, 3 }
limBands = limiterBandsPerOctave[limiter_bands-1]
patchBorders[0] = lsb
for(k = 1; k<= noPatches; k++) {
patchBorders[k] = patchBorders[k-1] + patchNoSubbands[k-1]
}
for(k = 0; k<= N_{Low}; k++) {
limTable[k] = $\mathbf{f}_{TableLow}$(k)
}
for(k = 1; k < noPatches; k++) {
limTable[k + N_{Low}] = patchBorders[k]
}
sort(limTable)
k = 1
nrLim = N_{Low} + noPatches - 1;
k > nrLim
True
False
Output variables:
N_L = nrLim
for(k = 0; k<= nrLim; k++) {
$\mathbf{f}_{TableLim}$(k) = limTable[k]
}
Done
nOctaves = log2(limTable[k] / limTable[k-1])
k = k +1
(nOctaves × limBands) < 0,49
True
False
limTable[k] == limTable[k-1]
True
False
Is limTable[k] equal to any element in patchBorders
True
False
Is limTable[k-1] equal to any element in patchBorders
True
False
Remove the k:th element from the array limTable
nrLim = nrLim -1
Remove the (k-1):th element from the array limTable
nrLim = nrLim -1

Figure 18: Calculation of $\mathbf{f}_{TableLim}$ if limiter_bands > 0

5.6.2.7 High frequency adjustment

The envelope adjuster takes the input QMF-matrix $\mathbf{X}_{High}$ and produces the output QMF matrix $\mathbf{Y}$. The envelope adjustment is done upon the entire SBR range covering *M* QMF-channels, starting on channel *lsb*, for the time-frame spanned by the current SBR frame (indicated by the vector $\mathbf{t}_E$). Throughout the description below several temporary vectors and matrices are introduced in order to make the explanation stringent. All temporary matrices and vectors are indexed from zero, removing the lsb offset. The below description of the envelope adjustment is channel independent, and outlined for one channel only, and for one SBR-frame only. Variables used below that originate from the processing of the previous frame, are assumed to be zero for the first frame.

5.6.2.7.1 Mapping

Some of the data extracted from the bitstream are vectors (or matrices) containing data elements representing a frequency range of several QMF channels. In order to simplify the explanation below, and sometimes out of necessity, this grouped data is mapped to the highest available frequency resolution for the envelope adjustment, i.e. the number of QMF channels within the SBR range. This means that several adjacent channels in the mapped vectors (or matrices) will have the same value. Furthermore, the same holds true for the time resolution of some of the data extracted from the bitstream. Hence, data elements representing a time-span of several QMF subsamples, are mapped to the highest time-resolution available for the envelope adjustment, i.e. the number of QMF-slots within the current frame.

The mapping of the envelope scalefactors and the noise floor scalefactors is outlined below. The envelope is mapped to the resolution of the QMF bank, albeit with preserved time resolution. The noise floor scalefactors are also mapped to the frequency resolution of the filterbank, but with the time resolution of the envelope scalefactors.

$$\mathbf{E}_{OrigMapped}\left(m-lsb,l\right)=\mathbf{E}_{Orig}\left(i,l\right)\quad,\mathbf{F}\left(i,\mathbf{f}\left(l\right)\right)\le m<\mathbf{F}\left(i+1,\mathbf{f}\left(l\right)\right),0\le i<\mathbf{n}\left(\mathbf{f}\left(l\right)\right),0\le l<L_E$$
$$\mathbf{Q}_{Mapped}\left(m-lsb,l\right)=\mathbf{Q}_{Orig}\left(i,k\left(l\right)\right)\quad,\mathbf{f}_{TableNoise}(i)\le m<\mathbf{f}_{TableNoise}(i+1),0\le i<N_Q,0\le l<L_E$$

where $k\left(l\right)$ is defined by $\mathbf{t}_E\left(l\right)\ge\mathbf{t}_Q\left(k\left(l\right)\right),\mathbf{t}_E\left(l+1\right)\le\mathbf{t}_Q\left(k\left(l\right)+1\right)$.

5.6.2.7.2 Estimation of current envelope

In order to envelope adjust the present frame, the envelope of the current SBR signal needs to be assessed. This is done according to below, dependent on the bitstream element *inter_freq*. The envelope is estimated by averaging the squared complex subband samples, over different time and frequency regions, given by the time/frequency grid represented by $\mathbf{t}_E$ and $\mathbf{f}$.

If interpolation (*inter_freq* = 1) is used:

$$\mathbf{E}_{Curr}\left(m,l\right)=\frac{\sum_{i=\mathbf{t}_E(l)+t_{HfAdj}}^{\mathbf{t}_E(l+1)-1+t_{HfAdj}}\left|\mathbf{X}_{High}\left(m+lsb,i\right)\right|^2}{\mathbf{t}_E(l+1)-\mathbf{t}_E(l)}\quad,\quad 0\le m<M,0\le l<L_E$$

else, no interpolation (*inter_freq* = 0):

$$\mathbf{E}_{Curr}\left(k-lsb,l\right)=\frac{\sum_{i=\mathbf{t}_E(l)+t_{HfAdj}}^{\mathbf{t}_E(l+1)-1+t_{HfAdj}}\sum_{j=k_l}^{k_h}\left|\mathbf{X}_{High}\left(j,i\right)\right|^2}{\left(\mathbf{t}_E(l+1)-\mathbf{t}_E(l)\right)\cdot\left(k_h-k_l\right)},$$

$$k_l\le k\le k_h,\begin{cases}k_l=\mathbf{F}\left(p,\mathbf{f}(l)\right)\\k_h=\mathbf{F}\left(p+1,\mathbf{f}(l)\right)-1\end{cases},0\le p<\mathbf{n}(\mathbf{f}(l)),0\le l<L_E$$

If interpolation is used the energies are averaged over every QMF filterbank channel, else the energies are averaged over every frequency band. In either case, the energies are stored with the frequency resolution of the QMF filterbank. Hence the $\mathbf{E}_{Curr}$ matrix has L_E columns (one for every envelope) and *M* rows (the number of QMF-channels covered by the SBR range).

5.6.2.7.3 Calculation of noise levels

The noise floor scalefactor is the ratio between the energy of the noise to be added to the envelope adjusted HF generated signal $\mathbf{X}_{High}$ and the energy of the same. Hence, in order to add the correct amount of noise, the noise floor scalefactor needs to be converted to a proper amplitude value, according to the following:

$$\mathbf{Q}_M(m,l) = \sqrt{\mathbf{E}_{OrigMapped}(m,l) \cdot \frac{\mathbf{Q}_{Mapped}(m,l)}{1+\mathbf{Q}_{Mapped}(m,l)}} \ , \quad 0 \le m < M \ ,0 \le l < L_E$$

5.6.2.7.4 Calculation of gain

The gain to be applied for the subband samples in order to retain the correct envelope is calculated according to below. The level of the additional added noise is taken into account.

$$\mathbf{G}(m,l) = \begin{cases} \sqrt{\dfrac{\mathbf{E}_{OrigMapped}(m,l)}{(\varepsilon+\mathbf{E}_{Curr}(m,l))\cdot(1+\delta(l)\cdot\mathbf{Q}_{Mapped}(m,l))}} & if \quad \mathbf{S}_M(m,l)=0 \\ \sqrt{\dfrac{\mathbf{E}_{OrigMapped}(m,l)}{(\varepsilon+\mathbf{E}_{Curr}(m,l))}\cdot\mathbf{Q}_{Mapped}(m,l)} & if \quad \mathbf{S}_M(m,l)\neq 0 \end{cases} \ ,0 \le m < M \ ,0 \le l < L_E$$

where:

$\delta(l) = \begin{cases} 0 & if \ l = l_A \\ 1 & otherwise \end{cases}$, and where $l_A = \begin{cases} middleBorder & if \ pointer \neq 0 \\ -1 & if \ pointer = 0 \end{cases}$

In order to avoid unwanted noise substitution the gain values are limited according to the following:

$$\mathbf{G}_{Max_{Temp}}(k,l) = \sqrt{\frac{\varepsilon_0 + \sum_{i=\mathbf{f}_{TableLim}(k)}^{\mathbf{f}_{TableLim}(k+1)-1} \mathbf{E}_{OrigMapped}(i,l)}{\varepsilon_0 + \sum_{i=\mathbf{f}_{TableLim}(k)}^{\mathbf{f}_{TableLim}(k+1)-1} \mathbf{E}_{Curr}(i,l)}} \cdot \mathbf{limGain}(limiter\_gains) \ , \quad 0 \le k < N_L \ ,0 \le l < L_E$$

$$\mathbf{G}_{Max}(m,l) = \min\left(\mathbf{G}_{Max_{Temp}}\left(\mathbf{f}_{TableLim}(k(m)),l\right),10^5\right), \quad 0 \le m < M \ ,0 \le l < L_E$$

where $k(m)$ is defined by $\mathbf{f}_{TableLim}(k(m)) \le m < \mathbf{f}_{TableLim}(k(m)+1)$,

and where $\mathbf{limGain} = \left[0.70795,\ 1.0,\ 1.41254,\ 10^{10}\right]$, and where $\varepsilon_0 \ = \ 10^{-12}$.

First limit the additional noise energy level. The additional noise added to the HF generated signal is also limited in proportion to the lost energy due to the limitation of the gain values according to the following:

$$\mathbf{Q}_{M_{Lim}}(m,l) = \min\left(\mathbf{Q}_M(m,l), \mathbf{Q}_M(m,l)\cdot\frac{\mathbf{G}_{Max}(m,l)}{\mathbf{G}(m,l)}\right) \ , \quad 0 \le m < M \ ,0 \le l < L_E$$

Then apply the limiter to the gain:

$$\mathbf{G}_{Lim}(m,l) = \min\left(\mathbf{G}(m,l), \mathbf{G}_{Max}(m,l)\right), \quad 0 \le m < M \ ,0 \le l < L_E$$

5.6.2.7.5 Assembling HF signals

The gain values to be applied to the subband samples are smoothed using the filter:

$$\mathbf{h}_{Smooth} = \begin{bmatrix} 0.33333333333333 \\ 0.30150283239582 \\ 0.21816949906249 \\ 0.11516383427084 \\ 0.03183050093751 \end{bmatrix}$$

The variable of length h_{SL} is used to control whether smoothing is applied or not, according to:

$$h_{SL} = \begin{cases} 4 & , smoothing\_mode = 0 \\ 0 & , smoothing\_mode = 1 \end{cases}.$$

For transients smoothing is not applied.

$$\mathbf{G}_{Temp}\left(m, i + h_{SL}\right) = \mathbf{G}_{LimBoost}\left(m, l\right), \mathbf{t}_E(l) \le i < \mathbf{t}_E(l+1), 0 \le l < L_E, 0 \le m < M$$

$$\mathbf{G}_{Filt}\left(m, i\right) = \begin{cases} \sum_{j=0}^{h_{SL}} \mathbf{G}_{Temp}\left(m, i - j + h_{SL}\right) \cdot \mathbf{h}_{Smooth}\left(j\right) & if\ l \ne l_A \\ \mathbf{G}_{Temp}\left(m, i + h_{SL}\right) & if\ l = l_A \end{cases},$$

for $\mathbf{t}(l) \le i < \mathbf{t}(l+1), 0 \le m < M, 0 \le l < L_E$, and where

$$l_A = \begin{cases} middleBorder & if\ pointer \ne 0 \\ -1 & if\ pointer = 0 \end{cases}.$$

The first h_{SL} columns of the $\mathbf{G}_{Temp}$ matrix are the last h_{SL} columns of the $\mathbf{G}_{Temp}$ matrix of the previous frame, unless the reset-flag is set ($reset = 1$) for which case the first h_{SL} columns of the $\mathbf{G}_{Temp}$ matrix are equal to $\mathbf{G}_{LimBoost}\left(m, 0\right)$ for all QMF-channels within the SBR-range.

The smoothed gain-values are applied to the input subband matrix $\mathbf{X}_{High}$, for all envelopes of the current frame, according to:

$$\mathbf{W}_1\left(m, i\right) = \mathbf{G}_{Filt}\left(m, i\right) \cdot \mathbf{X}_{High}\left(m + lsb, i\right), \quad \mathbf{t}_E(0) \le i < \mathbf{t}_E(L_E + 1), 0 \le m < M$$

The noise is added to the output signal, from the noise table $\mathbf{V}$. The level of the noise is smoothed similar to the smoothing of the gain values using the filter $\mathbf{h}_{Smooth}\left(n\right)$ of length h_{SL}:

$$\mathbf{Q}_{Temp}\left(m, i + h_{SL}\right) = \mathbf{Q}_{M_{Lim}Boost}\left(m, l\right), \mathbf{t}_E(l) \le i < \mathbf{t}_E(l+1), 0 \le l < L_E, 0 \le m < M$$

$$\mathbf{Q}_{Filt}\left(m, i\right) = \begin{cases} \sum_{j=0}^{h_{SL}} \mathbf{Q}_{Temp}\left(m, i - j + h_{SL}\right) \cdot \mathbf{h}_{Smooth}\left(j\right) & if\ l \ne l_A\ AND\ \mathbf{S}_{MBoost}\left(m, l\right) = 0 \\ 0 & if\ l = l_A\ OR\ \mathbf{S}_{MBoost}\left(m, l\right) \ne 0 \end{cases}$$

for $\mathbf{t}(l) \le i < \mathbf{t}(l+1), 0 \le m < M, 0 \le l < L_E$, and where:

$$l_A = \begin{cases} middle\_border & if\ pointer \ne 0 \\ -1 & if\ pointer = 0 \end{cases}.$$

The first h_{SL} columns of the $\mathbf{Q}_{Temp}$ matrix are the last h_{SL} columns of the $\mathbf{Q}_{Temp}$ matrix of the previous frame, unless the reset-flag is set ($Reset = 1$) for which case the first h_{SL} columns of the $\mathbf{Q}_{Temp}$ matrix are equal to $\mathbf{Q}_{M_{LimBoost}}(m,0)$ for all QMF-channels within the SBR-range.

The noise is added to the output according to:

$$\begin{cases} Re\{\mathbf{W}_2(m,i)\} = Re\{\mathbf{W}_1(m,i)\} + \mathbf{Q}_{Filt}(m,i)\cdot\mathbf{V}(0, f_{IndexNoise}(i)) \\ Im\{\mathbf{W}_2(m,i)\} = Im\{\mathbf{W}_1(m,i)\} + \mathbf{Q}_{Filt}(m,i)\cdot\mathbf{V}(1, f_{IndexNoise}(i)) \end{cases}, \quad \begin{cases} \mathbf{t}_E(l) \le i < \mathbf{t}_E(l+1), 0 \le l < L_E \\ 0 \le m < M \end{cases}$$

where:

$$f_{IndexNoise}(i) = (index_{Noise} + i) \bmod (512),$$

and $index_{Noise}$ is the last $f_{IndexNoise}$ from the previous frame, unless the reset-flag is set ($Reset = 1$) for which case $f_{IndexNoise} = 0 \times$ $\mathbf{V}$ is defined in the appendix.

5.6.2.8 Low Complexity stereo

In order to restore the stereo impression for stereo programme material downmixed to mono, a stereo synthesis tool is included. Hereby the tool ensures that the signal is left unprocessed when original mono programme material appears. Data describing the stereo image is signalled by lc_stereo_mode. Low Complexity stereo (LC-stereo) is enabled if and only if audio_mode is set to LC_STEREO mode.

5.6.2.8.1 Process

Let $Y_{Left}(z)$ and $Y_{Right}(z)$ be the output signal, in the Z-domain, for left and right channel respectively. Furthermore, let $X_{mono}(z)$ be the input signal, in the Z-domain, for the AAC and SBR decoded mono signal.

Low complexity stereo is then defined by the process:

$$Y_{Left}(z) = X_{mono}(z)\left(H_{LP}(z) + g_{i,dir}\cdot H_{HP}(z) + g_{i,del}\cdot z^{-d}\cdot H_{HP}(z)\cdot H_{Slope}(z)\right),$$

$$Y_{Right}(z) = X_{mono}(z)\left(H_{LP}(z) + g_{i,dir}\cdot H_{HP}(z) - g_{i,del}\cdot z^{-d}\cdot H_{HP}(z)\cdot H_{Slope}(z)\right),$$

where:

$$d = INT(0.007\cdot Fs)$$

and the filter transfer functions $H_{LP}(z)$, $H_{HP}(z)$ and $H_{Slope}(z)$ are short IIR filters using the filters structure:

$$H(z) = \frac{B(z)}{1 + A(z)},$$

where the coefficients in A and B are defined in annex I.

The gain values $g_{i,dir}$ and $g_{i,del}$ are calculated from the bitstream element *lc_stereo_mode*. The symbols $g_{i,dir}$ and $g_{i,del}$ represent the gain values after smoothing and interpolation.

Let $gain_{del}$ be the level of LC stereo for the current frame according to:

$$gain_{del} = \begin{cases} 0 & , \quad \textbf{lc\_stereo\_mode} = 0 \\ 0 & , \quad \textbf{lc\_stereo\_mode} = 1 \\ \frac{1}{\sqrt{5}} & , \quad \textbf{lc\_stereo\_mode} = 2 \\ \frac{1}{\sqrt{2}} & , \quad \textbf{lc\_stereo\_mode} = 3 \end{cases}$$

For frame *n*, smoothing using different attack and decay times is applied according to:

$$g_{del}(n) = \begin{cases} 0 \quad , \quad \textbf{lc\_stereo\_mode} = 0 \\ b\left(gain_{del}\left(z\right) + gain_{del}\left(n-1\right)\right) - ag_{del}(n-1) \quad , \quad g_{del}\left(n-1\right) \geq gain_{del}\left(n\right), \quad \textbf{lc\_stereo\_mode} > 0 \\ g_{del}\left(n-1\right) + c_a\left(gain_{ps}\left(n\right) - g_{del}\left(n-1\right)\right) \quad , \quad g_{del}\left(n-1\right) < gain_{del}\left(n\right), \quad \textbf{lc\_stereo\_mode} > 0 \end{cases}$$

and

$$g_{dir}\left(n\right) = \sqrt{1 - g_{del}^2\left(n\right)}$$

where $c_a = LC\_ATTACK\_COEFF$ and *a* and *b* are the filter coefficients of the LC_SMOOTH filter defined in annex I.

$$LC\_ATTACK\_COEFF = 1 - \exp\left(-\frac{1920}{LC\_ATTACK\_TIME\_CONST \cdot Fs}\right), \text{ where:}$$

$$LC\_ATTACK\_TIME\_CONST = 0.07.$$

The gain factors g_{dir} and g_{del} are interpolated linearly within the frame, according to below.

$$k_{g,del}\left(n\right) = \frac{g_{del}\left(n\right) - g_{del}\left(n-1\right)}{1920}$$

$$k_{g,dir}\left(n\right) = \frac{g_{dir}\left(n\right) - g_{dir}\left(n-1\right)}{1920}$$

$$g_{i,del}\left(n,i\right) = g_{del}\left(n-1\right) + i \cdot k_{g,del}\left(n\right) \quad , \quad 0 \leq i < 1920$$

$$g_{i,dir}\left(n,i\right) = g_{dir}\left(n-1\right) + i \cdot k_{g,dir}\left(n\right) \quad , \quad 0 \leq i < 1920$$

5.6.3 AAC + SBR protocol

The SBR bitstream is described by means of pseudo code, using a syntax similar to the one used in the MPEG standard documents. The location of the SBR bitstream within the AAC audio super frames was described in clause 5.3.2. This clause only describes how to extract the elements from the bitstream. The application thereof was briefly described in clause 4.

5.6.3.1 AAC + SBR syntax

Table 32: Syntax of sbr_aac_frame()

Syntax	No. of bits	Note
sbr_aac_frame(audio_mode) //audio_mode is located in the SDC		
{		
sbr_crc_bits	**8**	see annex D
if (**header_flag**)	**1**	
sbr_header (audio_mode)		
sbr_data (audio_mode, data extra) //data extra is located in sbr_header		
}		

Table 33: Syntax of sbr_header()

Syntax	No. of bits	Note
sbr_header (audio_mode)		
{		
protocol_version	**2**	
amp_res	**1**	
start_freq	**4**	**See note 1**
stop_freq	**4**	**See note 1**
xover_band	**3**	**See note 2**
reserved	**2**	
data_extra //enable optional data part	**1**	
header_extra_1 //enable optional header part 1	**1**	
header_extra_2 //enable optional header part 2	**1**	
if (audio_mode == LC_STEREO)		
lc_stereo_mode	**2**	
//optional parts		
if (header_extra_1) {		
freq_scale	**2**	
alter_scale	**1**	
noise_bands	**2**	
}		
if (header_extra_2) {		
limiter_bands	**2**	
limiter_gains	**2**	
interpol_freq	**1**	
smoothing_mode	**1**	
reserved	**1**	
}		
}		
NOTE 1: start_freq and stop_freq must define a frequency band that does not exceed 48 QMF channels. NOTE 2: This is the index into the master frequency band table at which the envelope data starts.		

Table 34: Syntax of sbr_data()

Syntax	No. of bits	Note
sbr_data (audio_mode, data_extra)		
{		
if (audio_mode != STEREO) {		
sbr_single_channel_element (data_extra)		
}		
else {		
sbr_channel_pair_element (data_extra)		
}		
}		

Table 35: Syntax of sbr_single_channel_element ()

Syntax	No. of bits	Note
sbr_single_channel_element (data_extra)		
{		
//side info		
sbr_grid (0)		
sbr_dtdf (0) //data delta coding direction		
invf_mode	**2**	
sbr_mode	**2**	
if (data_extra)		
reserved	**3**	
//raw data		
sbr_envelope (0,0)		
sbr_noise (0,0)		
//sbr extended data		
if (**extended_data**) {	**1**	
cnt = **extension_size** //cnt is value in bytes	**4**	
if (cnt = 15) {		
cnt += **esc_count**;	**8**	
}		
nr_bits_left= 8*cnt		
while(nr_bits_left > 7) {		
extension_id	**2**	
nr_bits_left -= 2		
sbr_extension(extension_id, 0, nr_bits_left)		**See note**
}		
}		
}		
NOTE: sbr_extension() must decrease nr_bits_left with the number of bits read.		

Table 36: Syntax of sbr_channel_pair_element ()

Syntax	No. of bits	Note
sbr_channel_pair_element (data_extra)		
{		
if (**coupling**) {	**1**	
//side info		
sbr_grid (0) //grid common to both channels		
num_env[1] = num_env[0]; num_noise[1] = num_noise[0]		
for(env = 0; env < num_env[0]; env++)		
freq_res[1][env] = freq_res[0][env]		
sbr_dtdf (0) //mono data delta coding direction		
sbr_dtdf (1) //stereo data delta coding direction		
invf_mode	**2**	
sbr_mode	**2**	
if (data_extra)		
reserved	**3**	
//raw data		
sbr_envelope (0,1) //mono data		
sbr_noise (0,1) //mono data		
sbr_envelope (1,1) //stereo data		
sbr_noise (1,1) //stereo data		
//sbr extended data, to be used for both channels		
if (**extended_data**) {	**1**	
cnt = **extension_size** //cnt is value in bytes	**4**	
if (cnt = 15) {		
cnt += **esc_count**;	**8**	
}		
nr_bits_left= 8*cnt		
while(nr_bits_left > 7) {		
extension_id	**2**	
nr_bits_left -= 2		
sbr_extension(extension_id, 0, nr_bits_left)		**See note**
}		

Syntax	No. of bits	Note
}		
}		
else {		
//side info		
sbr_grid (0) //left grid		
sbr_grid (1) //right grid		
sbr_dtdf (0) //left data delta coding direction		
sbr_dtdf (1) //right data delta coding direction		
invf_mode //left mode	**2**	
invf_mode //right mode	**2**	
sbr_mode //left SBR mode bits	**2**	
sbr_mode //right SBR mode bits	**2**	
if (data_extra)		
reserved	**6**	
//raw data		
sbr_envelope (0,0) //left data		
sbr_envelope (1,0) //right data		
sbr_noise (0,0) //left data		
sbr_noise (1,0) //right data		
//sbr extended data, left channel		
if (**extended_data**) {	**1**	
cnt = **extension_size** //cnt is value in bytes	**4**	
if (cnt = 15) {		
cnt += **esc_count**;	**8**	
}		
nr_bits_left= 8*cnt		
while(nr_bits_left > 7) {		
extension_id	**2**	
nr_bits_left -= 2		
sbr_extension(extension_id, 0, nr_bits_left)		**See note**
}		
}		
//sbr extended data, right channel		
if (**extended_data**) {	**1**	
cnt = **extension_size** //cnt is value in bytes	**4**	
if (cnt = 15) {		
cnt += **esc_count**;	**8**	
}		
nr_bits_left= 8*cnt		
while(nr_bits_left > 7) {		
extension_id	**2**	
nr_bits_left -= 2		
sbr_extension(extension_id, 1, nr_bits_left)		**See note**
}		
}		
}		
}		
NOTE: sbr_extension() must decrease nr_bits_left with the number of bits read.		

Table 37: Syntax of sbr_grid()

Syntax	No. of bits	Note
sbr_grid (ch)		
{		
switch (**frame_class**) {	2	
case FIXFIX		
num_env[ch] = 2^**temp** //number of envelopes	**2**	**See note**
freq_res **//**frame global frequency resolution	**1**	
for(env = 0; env < num_env[ch]; env++)		
freq_res[ch][env] = freq_res;		
break;		
case FIXVAR		
//mandatory part:		
abs_bord[ch] = **temp** + number_time_slots //last border	**3**	
num_rel[ch] //number of relative borders	**2**	
num_env[ch] = num_rel[ch] + 1 //number of envelopes		
for(rel = 0; rel < num_rel[ch]; rel++)		
rel_bord[ch][rel] = 2***temp** + 2; //relative borders, last to first	**2**	
ptr_bits = ceil (ln (num_rel[ch] + 2)/ln (2)) //number of pointer bits		
pointer[ch]	**ptr_bits**	
for(env = 0; env < num_env[ch]; env++)		
freq_res[ch][num_env[ch] - 1 - env] = **freq_res**;//last to first	**1**	
break;		
case VARFIX		
//mandatory part:		
abs_bord[ch] //first border	**3**	
num_rel[ch] //number of relative borders	**2**	
num_env[ch] = num_rel[ch] + 1 //number of envelopes		
for(rel = 0; rel < num_rel[ch]; rel++)		
rel_bord[ch][rel] = 2***temp** + 2; //relative borders, first to last	**2**	
ptr_bits = ceil (ln (num_rel[ch] + 2)/ln (2)) //number of pointer bits		
pointer[ch]	**ptr_bits**	
for(env = 0; env < num_env[ch]; env++)		
freq_res[ch] [env] = **freq_res**; //freq resolutions, first to last	**1**	
break;		
case VARVAR		
//mandatory part:		
abs_bord_0[ch] //first border	**3**	
abs_bord_1[ch] = **temp** + number_time_slots //last border	**3**	
num_rel_0[ch] //number of relative borders, first to last	**2**	
num_rel_1[ch] //number of relative borders, last to first	**2**	
num_env[ch] = num_rel_0[ch] + num_rel_1[ch] + 1 //number of env		
for(rel = 0; rel < num_rel_0[ch]; rel++)		
rel_bord_0[ch][rel] = 2***temp** + 2; //relative borders, first to last	**2**	
for(rel = 0; rel < num_rel_1[ch]; rel++)		
rel_bord_1[ch][rel] = 2***temp** + 2; //relative borders, last to first	**2**	
ptr_bits = ceil (ln (num_rel_0[ch] + num_rel_1[ch] + 2)/ln (2))//ptr bits		
pointer[ch]	**ptr_bits**	
for(env = 0; env < num_env[ch]; env++)		
freq_res[ch] [env] = **freq_res**; //freq resolutions, first to last	**1**	
break;		
if(num_env[ch] > 1)		
num_noise[ch] = 2 //number of noise floors		
else		
num_noise[ch] = 1		
}		
}		
NOTE: In addition, the condition num_env ≤ 5 must be true.		

Table 38: Syntax of sbr_dtdf ()

Syntax	No. of bits	Note
sbr_dtdf (ch)		
{		
for(env; env < num_env[ch]; env++)		
dt_env[ch][env] = **dt_flag**;	**1**	
for(noise; noise < num_noise[ch]; noise++)		
dt_noise[ch][env] = **dt_flag**;	**1**	
}		

Table 39: Syntax of sbr_envelope ()

Syntax	No. of bits	Note
sbr_envelope (ch,coupling)		
{		
//Huffman table selection		
if(coupling) {		
if(ch) {		
if(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(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;		
}		
}		
} else {		
if(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;		
}		
}		
//raw data extraction		
for(env = 0; env < num_env[ch]; env++)		
{		
if(dt_env[ch][env])		
{		
for(band = 0; band < num_env_bands[freq_res[ch][env]]; band++)		**See note 1**
data_env[ch][env][band] = huff_dec(t_huff,**codeword**);	**1...18**	**See note 2**
}		
else		
{		
if(coupling && ch)		
data_env[ch][env][0] = **env_start_value_stereo**;	**5/6**	**amp_res = 1/ amp_res = 0**
else		
data_env[ch][env][0] = **env_start_value_mono**;	**6/7**	**amp_res = 1/ amp_res = 0**

Syntax	No. of bits	Note
for(band = 1; band < num_env_bands[freq_res[ch][env]]; band++)		**See note 1**
data_env[ch][env][band] = huff_dec(f_huff,**codeword**);	**1...18**	**See note 2**
}		
}		
}		
NOTE 1: num_env_bands[freq_res[ch][env]] is calculated in clause 5.6.2.3 and is named n.		
NOTE 2: huff_dec() is explained further in annex I: SBR Huffman Tables.		

Table 40: Syntax of sbr_noise ()

Syntax	No. of bits	Note
sbr_noise (ch,coupling)		
{		
//Huffman table selection		
if(coupling) {		
if(ch) {		
t_huff = t_huffman_noise_bal_3_0dB;		
f_huff = f_huffman_noise_bal_3_0dB;		
}		
else		
{		
t_huff = t_huffman_noise_3_0dB;		
f_huff = f_huffman_noise_3_0dB;		
}		
}		
else		
{		
t_huff = t_huffman_noise_3_0dB;		
f_huff = f_huffman_noise_3_0dB;		
}		
//raw data extraction		
for(noise = 0; noise < num_noise[ch]; noise++)		
{		
if(dt_noise[ch][noise])		
{		
for(band = 0; band < num_noise_bands[ch]; band++)		**1**
data_noise[ch][noise][band] = huff_dec(t_huff,**codeword**);	**1..18**	**2**
}		
else		
{		
if(coupling && ch)		
data_noise[ch][noise][0] = **noise_start_value_stereo**;	**5**	
else		
data_noise[ch][noise][0] = **noise_start_value_mono**;	**5**	
for(band = 1; band < num_noise_bands[ch]; band++)		
data_noise[ch][noise][band] = huff_dec(f_huff,**codeword**);	**1...18**	**2**
}		
}		
}		
NOTE 1: num_noise_bands[ch] is calculated in clause 5.6.2.3 and is named N_Q. In addition, the condition $N_Q \leq 5$, must be true.		
NOTE 2: huff_dec() is explained further in annex I: SBR Huffman Tables.		

5.6.3.2 SBR bit stream element definitions

sbr_crc_bits Cyclic redundancy checksum for the SBR bit stream part. The CRC algorithm is applied to the first crc_len number of bits after the sbr_crc_bits as defined by table 41. *fkn* is defined as count = frame_length - num_codec_bits, whereas num_codec_bits is the number of bits consumed by the core coder (AAC).

Table 41: Number of bits covered by the CRC

audio_mode	header_flag	No. of bits
MONO	1	crc_len = min(74, *fkn*(count))
MONO	0	crc_len = min(47, *fkn*(count))
LC_STEREO	1	crc_len = min(76, *fkn*(count))
LC_STEREO	0	crc_len = min(47, *fkn*(count))
STEREO	1	crc_len = min(120, *fkn*(count))
STEREO	0	crc_len = min(93, *fkn*(count))

header_flag Indicates if a SBR header is present.

protocol_version Defines the version of the SBR protocol as given by table 42.

Table 42: Definition of protocol_version

protocol_version	Version	Note
0	DRM SBR Bitstream Format V 1.0	As defined by the present document
1	reserved	
2	reserved	
3	reserved	

amp_res Defines the resolution of the envelope estimates as given by table 43.

Table 43: Definition of amp_res

amp_res	Meaning	Note
0	1,5 dB	
1	3,0 dB	

start_freq Input parameter to function that calculates start of master frequency band table.

stop_freq Input parameter to function that calculates stop of master frequency band table.

xover_band Index to master frequency band table.

Table 44: Evaluation of start frequency function for *F*s = 48 000

start_freq	Frequency (Hz)
0	4 125
1	4 500
2	4 875
3	5 250
4	5 625
5	6 000
6	6 375
7	6 750
8	7 500
9	8 250
10	9 000
11	10 125
12	11 625
13	n/a
14	n/a
15	n/a
NOTE: Not applicable when AAC core operates at *F*s = 24 000.	

Table 45: Evaluation of stop frequency function for *F*s = 48 000

stop_freq	Frequency (Hz)
0	7 875
1	8 625
2	9 375
3	10 125
4	10 875
5	12 000
6	13 125
7	14 250
8	15 375
9	16 875
10	18 375
11	20 250
12	22 125
13	24 000
14	see clause 5.6.2.3
15	see clause 5.6.2.3

Table 46: Definition of xover_band

xover_band	Description	Note
0-7	Index to master frequency band table	

data_extra Indicates whether the optional data part is enabled.

header_extra_1 Indicates whether the optional header part 1 is enabled.

header_extra_2 Indicates whether the optional header part 2 is enabled.

lc_stereo_mode Defines the low complexity stereo mode behaviour as given by table 47.

Table 47: Definition of lc_stereo_mode

lc_stereo_mode	Meaning	Note
0	Off	bypass smoothing
1	None	
2	Narrow	
3	Wide	

freq_scale Defines the frequency envelope bands grouping scale as given by table 48.

Table 48: Definition of freq_scale

freq_scale	Scale	Note
0	Linear	
1	12 bands/octave	
2	10 bands/octave	
3	8 bands/octave	

alter_scale Further defines the frequency envelope bands grouping as given by table 49.

Table 49: Definition of alter_scale

alter_scale	Action for freq_scale = 0	Action for freq_scale > 0
0	no grouping of channels	no alteration
1	groups of 2 channels	extra wide bands in highest range

noise_bands Defines the number of noise bands as given by table 50.

Table 50: Definition of noise_bands

noise_bands	Meaning	Note
0	1 band	
1	1 band/octave	
2	2 band/octave	
3	3 band/octave	

limiter_bands Defines the number of limiter bands as given by table 51.

Table 51: Definition of limiter_bands

limiter_bands	Meaning	Note
0	1 band	single band
1	1,2 bands/octave	multi-band
2	2,0 bands/octave	multi-band
3	3,0 bands/octave	multi-band

limiter_gains Defines the maximum gain of the limiters as given by table 52.

Table 52: Definition of limiter_gains

limiter_gains	Max Gain [dB]	Note
0	-3	
1	0	
2	3	
3	infinite (i.e. limiter off)	

interpol_freq Defines if the frequency interpolation shall be applied as given by table 53.

Table 53: Definition of interpol_freq

interpol_freq	Meaning	Note
0	Off	
1	On	

smoothing_mode Defines if smoothing shall be applied as given by table 54.

Table 54: Definition of smoothing_mode

smoothing_mode	Meaning	Note
0	On	
1	Off	

invf_mode Defines how inverse filtering shall be applied as given by table 55.

Table 55: Definition of invf_mode

invf_mode	Meaning	Note
0	no inverse filtering	
1	low level inverse filtering	
2	intermediate inverse filtering	
3	strong inverse filtering	

sbr_mode	Defines the SBR algorithm to be used as given by table 56.

Table 56: Definition of sbr_mode

sbr_mode	Meaning	Note
0	reserved	to be ignored in V1.0
1	reserved	to be ignored in V1.0
2	SBR V1.0	
3	reserved	to be ignored in V1.0
NOTE: This bitstream field is targeted for future enhancements. A version 1.0 decoder implementation shall simply ignore this bitstream field. A version 1.0 encoder shall set **sbr_mode** to 2. A future decoder, potentially incorporating enhanced versions of the SBR decoding algorithm (and including SBR V1.0), shall depend on the **sbr_mode,** and will use the SBR version that is signalled by this bitstream field.		

extended_data	Indicates whether an SBR extended data element is present.
extension_size	Defines the size of the SBR extended data element in bytes.
esc_count	Further defines the size of the SBR extended data element in cases where the size is bigger than 14 bytes.
extension_id	Holds an ID of the SBR extended data element.
coupling	Indicates whether the stereo information between the two channels is coupled or not.
frame_class	Indicates the framing class of the current frame.
freq_res	Indicates the frequency resolution.
num_rel	Indicates the number of relative borders within the time grid information for the current frame.
pointer	Points to a border within the time grid.
abs_bord	Indicates the location of the variable border within the time grid.
dt_flag	Indicates whether to apply delta decoding in time or frequency direction.

Table 57: Definition of dt_flag

dt_flag	Meaning	Note
0	Apply delta decoding in frequency direction	
1	Apply delta decoding in time direction	

env_start_value_stereo	Holds the first envelope value in case of a coupled stereo bit stream.
env_start_value_mono	Holds the first envelope value in case of a non-coupled stereo or a mono bit stream.
noise_start_value_stereo	Holds the first noise value in case of a coupled stereo bit stream.
noise_start_value_mono	Holds the first noise value in case of a non-coupled stereo or a mono bit stream.

6 Multiplex definition

6.1 Introduction

The DRM transmission super frame consists of three channels: the Main Service Channel (MSC), the Fast Access Channel (FAC), and the Service Description Channel (SDC). The MSC contains the data for the services. The FAC provides information on the channel width and other such parameters and also provides service selection information to allow for fast scanning. The SDC gives information on how to decode the MSC, how to find alternative sources of the same data, and gives the attributes of the services within the multiplex. It can include links to analogue simulcast services.

6.2 Main Service Channel

6.2.1 Introduction

The Main Service Channel (MSC) contains the data for all the services contained in the DRM multiplex. The multiplex may contain between one and four services, and each service may be either audio or data. The gross bit rate of the MSC is dependent upon the DRM channel bandwidth and the transmission mode.

6.2.2 Structure

The MSC contains between one and four streams. Each stream is divided into logical frames each 400 ms long. Audio streams comprise compressed audio and optionally they can carry text messages. Data streams may be composed of up to four "sub-streams" consisting of data packets. A sub-stream carries packets for one service. An audio service comprises one audio stream and optionally one data stream or one data sub-stream. A data service comprises one data stream or one data sub-stream.

Each logical frame generally consists of two parts, each with its own protection level. The lengths of the two parts are independently assigned. Unequal error protection for a stream is provided by setting different protection levels to the two parts. The logical frames from all the streams are mapped together to form multiplex frames of 400 ms duration which are passed to the channel coder. Alternatively, the first stream may be carried in logical frames mapped in to hierarchical frames.

The multiplex configuration is signalled using the SDC. The multiplex may be reconfigured at transmission super frame boundaries.

Annex M contains some examples of different MSC configurations.

6.2.3 Building the MSC

The MSC consists of a sequence of multiplex frames, and if hierarchical modulation is in use a sequence of hierarchical frames also. The multiplex frames and hierarchical frames are passed separately to the channel coder.

6.2.3.1 Multiplex frames

The multiplex frames are built by placing the logical frames from each non-hierarchical stream together. The logical frames consist, in general, of two parts each with a separate protection level. The multiplex frame is constructed by taking the data from the higher protected part of the logical frame from the lowest numbered stream (stream 0 when hierarchical modulation is not used, or stream 1 when hierarchical modulation is used) and placing it at the start of the multiplex frame. Next the data from the higher protected part of the logical frame from the next lowest numbered stream is appended and so on until all streams have been transferred. The data from the lower protected part of the logical frame from the lowest numbered stream (stream 0 when hierarchical modulation is not used, or stream 1 when hierarchical modulation is used) is then appended, followed by the data from the lower protected part of the logical frame from the next lowest numbered stream, and so on until all streams have been transferred. The higher protected part is designated part A and the lower protected part is designated part B in the multiplex description.

The multiplex frame is larger than or equal to the sum of the logical frames from which it is formed. The remainder, if any, of the multiplex frame shall be filled with 0s. These bits shall be ignored by the receiver.

NOTE: No padding bits are inserted between the end of part A and the beginning of part B. The capacity of part A of the multiplex frame is equal to the sum of the higher protected parts of the logical frames, but as a result of restrictions introduced by the channel encoding procedure applied for DRM (see clause 7.2.1.1), some of the bits nominally belonging to the lower protected part B of a multiplex frame might in fact be protected at the higher level.

6.2.3.2 Hierarchical frames

The hierarchical frames only exist when hierarchical modulation is used. They are built by taking the data from the logical frame from stream 0 and placing it at the start of the hierarchical frame.

The hierarchical frame is larger than or equal to the logical frame from which it is formed. The remainder, if any, of the hierarchical frame shall be filled with 0s. These bits shall be ignored by the receiver.

6.2.4 Reconfiguration

A reconfiguration of the multiplex occurs when the channel parameters in the FAC are changed, or when the services in the multiplex are reorganized. The new configuration is signalled ahead of time in the SDC and the timing is indicated by the reconfiguration index in the FAC. Clause 6.4.6 describes the signalling of a reconfiguration.

6.3 Fast Access Channel

6.3.1 Introduction

The FAC is used to provide service selection information for fast scanning. It contains information about the channel parameters (for example the spectrum occupancy and interleaving depth) such that a receiver is able to begin to decode the multiplex effectively. It also contains information about the services in the multiplex to allow the receiver to either decode this multiplex or change frequency and search again.

6.3.2 Structure

Each transmission frame contains an FAC block. An FAC block contains parameters that describe the channel and parameters to describe one service along with a CRC. When more than one service is carried in the multiplex, a number of FAC blocks are required to describe all the services.

6.3.3 Channel parameters

The channel parameters are as follows:

-	Base/Enhancement flag	1 bit
-	Identity	2 bits
-	Spectrum occupancy	4 bits
-	Interleaver depth flag	1 bit
-	MSC mode	2 bits
-	SDC mode	1 bit
-	Number of services	4 bits
-	Reconfiguration index	3 bits
-	Rfu	2 bits

The following definitions apply:

Base/Enhancement flag: this 1-bit flag indicates whether the transmission is the base or enhancement layer as follows:

0: Base layer - decodable by all DRM receivers

1: Enhancement layer - only decodable by receivers with enhancement layer capabilities

Identity: this 2-bit field identifies the current frame and also validates the SDC AFS index (see clause 6.4) as follows:

00: first FAC of the transmission super frame and AFS index is valid

01: second FAC of the transmission super frame

10: third FAC of the transmission super frame

11: first FAC of the transmission super frame and AFS index is invalid

Spectrum occupancy: this 4-bit field specifies the configuration of the digital signal. See clause 8.

Interleaver depth flag: this 1-bit flag indicates the depth of the time interleaving as follows:

0: 2 s (long interleaving)

1: 400 ms (short interleaving)

MSC mode: this 2-bit field indicates the modulation mode in use for the MSC as follows:

00: 64-QAM, no hierarchical

01: 64-QAM, hierarchical on I

10: 64-QAM, hierarchical on I&Q

11: 16-QAM, no hierarchical

SDC mode: this 1-bit field indicates the modulation mode in use for the SDC as follows:

0: 16-QAM

1: 4-QAM

Number of services: this 4-bit field indicates the number of audio and data services as follows:

0000: 4 audio services

0001: 1 data service

0010: 2 data services

0011: 3 data services

0100: 1 audio service

0101: 1 audio service and 1 data service

0110: 1 audio service and 2 data services

0111: 1 audio service and 3 data services

1000: 2 audio services

1001: 2 audio services and 1 data service

1010: 2 audio services and 2 data services

1011: reserved

1100: 3 audio services

1101: 3 audio services and 1 data service

1110: reserved

1111: 4 data services

Reconfiguration index: this 3-bit field indicates the status and timing of a multiplex reconfiguration. A non-zero value indicates the number of transmission super frames of the old configuration that are transmitted before the new configuration takes effect, see clause 6.4.6.

Rfu: these 2 bits are reserved for future use and shall be set to zero until they are defined.

6.3.4 Service parameters

The service parameters are as follows:

-	Service identifier	24 bits
-	Short identifier	2 bits
-	CA indication	1 bit
-	Language	4 bits
-	Audio/Data flag	1 bit
-	Service descriptor	5 bits
-	Rfa	7 bits

The following definitions apply:

Service identifier: this 24-bit field indicates the unique identifier for this service.

Short Id: this 2-bit field indicates the short identifier assigned to this service and used as a reference in the SDC. The Short Id is assigned for the duration of the service and is maintained through multiplex reconfigurations.

CA indication: this 1-bit flag indicates whether the service uses conditional access as follows:

0: No CA system is used

1: CA system is used

NOTE 1: The details are provided by the SDC data entity type 2.

Language: this 4-bit field indicates the language of the target audience as defined in table 58.

NOTE 2: Further languages are also indicated by SDC data entity type 12.

Audio/Data flag: this 1-bit flag indicates whether the service is audio or data as follows:

0: Audio service

1: Data service

Service descriptor: this 5-bit field depends upon the value of the Audio/Data flag as follows:

0: Programme type

1: Application identifier

Regardless of the value of the Audio/Data flag, the value 31 (all bits set to 1) indicates that a standard DRM receiver should skip this broadcast and continue to scan for services.

NOTE: This is to allow for engineering test transmissions to be ignored by standard receivers.

Programme type: this 5-bit field indicates the programme type of an audio service as defined in table 59.

Application identifier: this 5-bit field indicates the application identifier of a data service as defined in TS 101 968 [11]

Rfa: these 7 bits are reserved for future additions and shall be set to zero until they are defined.

Table 58: Language codes

Decimal number	Language	Decimal number	Language
0	No language specified	8	Hindi
1	Arabic	9	Japanese
2	Bengali	10	Javanese
3	Chinese (Mandarin)	11	Korean
4	Dutch	12	Portuguese
5	English	13	Russian
6	French	14	Spanish
7	German	15	Other language

Table 59: Programme Type codes

Decimal number	Programme type	Decimal number	Programme type
0	No programme type	16	Weather/meteorology
1	News	17	Finance/Business
2	Current Affairs	18	Children's programmes
3	Information	19	Social Affairs
4	Sport	20	Religion
5	Education	21	Phone In
6	Drama	22	Travel
7	Culture	23	Leisure
8	Science	24	Jazz Music
9	Varied	25	Country Music
10	Pop Music	26	National Music
11	Rock Music	27	Oldies Music
12	Easy Listening Music	28	Folk Music
13	Light Classical	29	Documentary
14	Serious Classical	30	*Not used*
15	Other Music	31	*Not used - skip indicator*

6.3.5 CRC

The 8-bit Cyclic Redundancy Check shall be calculated on the channel and service parameters. It shall use the generator polynomial $G_8(x) = x^8 + x^4 + x^3 + x^2 + 1$. See annex D.

6.3.6 FAC repetition

The FAC channel parameters shall be sent in each FAC block. The FAC service parameters for one service shall be sent in each block. When there is more than one service in the multiplex the repetition pattern is significant to the receiver scan time. When all services are of the same type (e.g. all audio or all data) then the services shall be signalled sequentially. When a mixture of audio and data services is present then the patterns shown in table 60 shall be signalled.

Table 60: Service parameter repetition patterns for mixtures of audio and data services

Number of audio services	Number of data services	Repetition pattern
1	1	A1A1A1A1D1
1	2	A1A1A1A1D1A1A1A1A1D2
1	3	A1A1A1A1D1A1A1A1A1D2A1A1A1A1D3
2	1	A1A2A1A2D1
2	2	A1A2A1A2D1A1A2A1A2D2
3	1	A1A2A3A1A2A3D1

Where A*n* designates an audio service and D*n* designates a data service.

6.4 Service Description Channel

6.4.1 Introduction

This clause describes the format and content of the SDC. The SDC gives information on how to decode the MSC, how to find alternative sources of the same data, and gives attributes to the services within the multiplex.

The data capacity of the SDC varies with the spectrum occupancy of the multiplex and other parameters. The SDC capacity can also be increased by making use of the AFS index.

Alternative frequency checking may be achieved, without loss of service, by keeping the data carried in the SDC quasi-static. Therefore, the data in the SDC frames has to be carefully managed.

6.4.2 Structure

An SDC block is the SDC data contained in one transmission super frame.

The SDC is treated as a single data channel. The total amount of data to be sent may require more than a single SDC block to send. An AFS index is therefore provided to permit a receiver to know when the next occurrence of the current SDC block will be transmitted, and so allow for alternative frequency checking and switching (AFS). A validity function is provided in the FAC to indicate whether the AFS index is valid or not, indicating to a receiver when the AFS function can operate.

The SDC block is made up as follows:

- AFS index	4 bits
- data field	*n* bytes
- CRC	16 bits

The **AFS index** is an unsigned binary number in the range 0 to 15 that indicates the number of transmission super frames which separate this SDC block from the next with identical content when the identity field in the FAC is set to 00. The AFS index shall be identical for all SDC blocks. The AFS index may be changed at reconfiguration.

The **data field** carries a variable number of data entities. It may contain an end marker and padding. The length of the data field depends upon the robustness mode, SDC mode and spectrum occupancy, and is given in table 61.

Table 61: Length of SDC data field

Robustness mode	SDC mode	Length of data field (bytes)					
		Spectrum occupancy					
		0	1	2	3	4	5
A	0	37	43	85	97	184	207
	1	17	20	41	47	91	102
B	0	28	33	66	76	143	161
	1	13	15	32	37	70	79
C	0	-	-	-	68	-	147
	1	-	-	-	32	-	72
D	0	-	-	-	33	-	78
	1	-	-	-	15	-	38

The **CRC** (Cyclic Redundancy Check) field shall contain a 16-bit CRC calculated over the AFS index coded in an 8-bit field (4 msbs are 0) and the data field. It shall use the generator polynomial $G_{16}(x) = x^{16} + x^{12} + x^5 + 1$. See annex D.

6.4.3 Data entities

The data field is filled with data entities. Every data entity has a 12-bit header and a variable length body. The header has the following format:

-	length of body	7 bits
-	version flag	1 bit
-	data entity type	4 bits

The following definitions apply:

The **length of body** gives the number of whole bytes occupied by the data entity body.

The **version flag** controls the management of data in the receiver.

The **data entity type** is a number that determines the identity of the data entity.

The version flag allows three different mechanisms to control data management in the receiver, as specified below. The actual mechanism used is specified for each data entity.

Reconfiguration: For data entities using this mechanism, the version flag indicates whether the data is for the current (= 0) or next (= 1) configuration.

List: For data entities using this mechanism, the version flag indicates the version of the list. When any of the data in the list changes, the flag is inverted and the existing data in the receiver is discarded. The version flag applies to all the data delivered using the data entity type.

Unique: For data entities using this mechanism, the version flag has no meaning and shall be set to 0. These data entities carry data that is unique and therefore do not require any change mechanism:

The body of the data entities shall be at least 4 bits long. The length of the body, excluding the initial 4 bits, shall be signalled by the header.

When there is space remaining in the data field, a data end marker shall be sent. The data end marker shall be 0x00. Any remaining space shall be filled with padding. The padding shall take the value 0x00.

6.4.3.1 Multiplex description data entity - type 0

Each SDC block should contain a multiplex description entity. This data entity uses the reconfiguration mechanism for the version flag. The current configuration can always be signalled. During a reconfiguration (i.e. when the FAC reconfiguration index is non-zero) the next configuration shall be signalled. This data entity describes the multiplex of streams within the MSC. The information is as follows:

-	protection level for part A	2 bits
-	protection level for part B	2 bits
-	stream description for stream 0	24 bits

and optionally, dependent upon the number of streams in the multiplex:

-	stream description for stream 1	24 bits
-	stream description for stream 2	24 bits
-	stream description for stream 3	24 bits

The stream description for stream 0 depends on whether the MSC mode field of the FAC indicates that the hierarchical frame is present or not.

If the hierarchical frame is not present then the stream description is as follows:

- data length for part A	12 bits
- data length for part B	12 bits

If the hierarchical frame is present then the stream description is as follows:

- protection level for hierarchical	2 bits
- rfu	10 bits
- data length for hierarchical	12 bits

The stream descriptions for streams 1, 2 and 3, when present, are as follows:

- data length for part A	12 bits
- data length for part B	12 bits

The following definitions apply:

protection level for part A: this field gives the overall coding rate for data in part A (see clause 7.5.1)

protection level for part B: this field gives the overall coding rate for data in part B (see clause 7.5.1)

data length for part A: this field gives the net length of data in bytes in part A of the MSC frame used by this stream

data length for part B: this field gives the net length of data in bytes in part B of the MSC frame used by this stream

protection level for hierarchical: this field gives the overall coding rate for data in the hierarchical frame (see clause 7.5.1)

rfu: these 10 bits shall be reserved for future use by the stream description field and shall be set to zero until they are defined

data length for hierarchical: this field gives the net length of data in bytes in the hierarchical part of the MSC frame used by this stream

When equal error protection is allocated to the multiplex frame (i.e. only one protection level is used) then the data length for the part A fields shall be set to 0 and the protection level for part A fields shall be set to 0.

When unequal error protection is allocated to the multiplex frame then part A is the higher protected part and part B is the lower protected part.

NOTE 1: If more than one service is carried in the multiplex, a service may be carried in both parts (some data in part A and some data in part B), or it may be carried only in one part (part A or part B). In this way, different services can be transported using unequal error protection, equal error protection at the higher level or equal error protection at the lower level in the same multiplex.

NOTE 2: The receiver may determine the number of streams present in the multiplex by dividing the value of the length field of the header by three.

6.4.3.2 Label data entity - type 1

Services may be labelled. The label should be sent in every SDC block to enable fast display, although for data services the repetition rate can be lowered. This data entity uses the unique mechanism for the version flag. The information is as follows:

- Short Id	2 bits
- rfu	2 bits
- label	*n* bytes

The following definitions apply:

Short Id: this field contains the short Id that relates the information to the Service Id provided by the FAC

rfu: these 2 bits are reserved for future use and shall be set to zero until they are defined

label: this is a variable length field of up to 16 bytes defining the label using UTF-8 coding [5]

NOTE: The length of the label is given by the length field of the header.

6.4.3.3 Conditional access parameters data entity - type 2

This data entity allows the conditional access parameters to be sent. This data entity uses the reconfiguration mechanism for the version flag.

-	Short Id	2 bits
-	rfu	2 bits
-	CA system identifier	8 bits
-	CA system specific information	*n* bytes

The following definitions apply:

Short Id: this field contains the short Id that relates the information to the Service Id provided by the FAC

rfu: these 2 bits are reserved for future use and shall be set to zero until they are defined

CA system identifier: this field indicates the CA system used by this service

CA system specific information: this is a variable length field containing CA system specific data

6.4.3.4 Frequency information data entity - type 3

A service may be available on several DRM frequencies. The Frequency information data entity allows these frequencies to be signalled. This data entity uses the list mechanism for the version flag.

Frequency information can also be provided for other services. These may be DRM services or carried on AM, FM, or DAB [3].

Frequency information is grouped by geographical area. The areas are common for all services signalled in a DRM multiplex, whether they are actually carried in the multiplex or not. The geographical areas can be described by using the Region definition data entity.

The frequency information may be essentially static (i.e. the same frequencies are used 24 hours per day) or may vary according to the time of day. In the latter case the Frequency schedule data entity is also used.

Service linking is provided by the Linkage data entity. This allows the connection between a service carried in the DRM multiplex and any alternative sources of the service, whether they be carried on DRM, AM, FM-RDS or DAB.

The information is as follows:

-	list type flag	1 bit
-	system id	3 bits
-	list type field	4 bits
-	total number of frequency groups, *m*	4 bits
-	*m* frequency groups	

Each frequency group is coded as follows:

-	group Id	4 bits
-	number of frequencies, *n*	4 bits
-	*n* frequencies	$n \times$ (8 or 16) bits

The following definitions apply:

List type flag: this flag indicates whether the information is static or scheduled as follows:

0: static frequency information

1: scheduled frequency information

System id: this field indicates which broadcast system the frequency information applies to as follows:

000: DRM: alternative frequencies for this entire multiplex

001: DRM: frequencies for a service carried in another multiplex (and AM simulcasts of that service)

010: FM-RDS service (Europe and North America grid)

011: FM-RDS service (Asia grid)

100: DAB service

101: reserved

110: reserved

111: reserved

List type field: this field depends upon the value of the list type flag as follows:

List type flag = 0:

when System id = 000:

rfa	3 bits
enhancement flag	1 bit

when System id = 001

ShortId	2 bits
Announce flag	1 bit
rfa	1 bit

when System id = 010, 011, 100

ShortId	2 bits
rfa	2 bits

List type flag = 1:

Freq List Id	4 bits

rfa: these 1, 2 or 3 bits are reserved for future additions by the list type field and shall be set to zero until they are defined.

enhancement flag: this flag indicates whether the frequency information which follows is related to the base layer or the enhancement layer channel as follows:

0: information refers to the base layer channel

1: information refers to the enhancement layer channel

Short Id: this field contains the short Id for the service that the frequency information which follows refers to.

Announce flag: this flag indicates whether the frequency information which follows is related to the tuned service or to the announcement service as follows:

0: information refers to the tuned service; the Short Id provides the link to the Service Id contained in the FAC

1: information refers to the announcement service; the ShortId provides the link via data entity type 6 to the ServiceId of the announcement service

Freq list Id: this field indicates the identity of the frequency list. Up to 16 separate frequency lists may be specified.

total number of frequency groups: this field, coded as an unsigned binary number, shall indicate the total number of frequency groups in this list minus 1 (- 1), even if the groups are carried in more than one data entity. Between 1 and 16 frequency groups may be specified per service or frequency list.

group Id: this field indicates an identifier for the following set of frequencies.

number of frequencies: this field, coded as an unsigned binary number, indicates the number of frequency fields minus 1 (-1) which follow.

frequency: the frequency field depends upon the value of the system id field as follows:

System id field	*identifier field*	*field length*
000	DRM/AM frequency	16 bits
001	DRM/AM frequency	16 bits
010	FM1 frequency	8 bits
011	FM2 frequency	8 bits
100	DAB frequency	8 bits

DRM/AM frequency: this consists of the following fields:

- analogue/digital flag	1 bit (0 = DRM, 1 = analogue)
- frequency in kHz	15 bits

NOTE: When System Id = 000, the analogue/digital flag shall be set to 0.

FM1 (87,5 MHz to 107,9 MHz) frequency:

code	***meaning***
0 - 204:	FM frequencies 87,5 MHz - 107,9 MHz (100 kHz step)
255	FM available, frequency not signalled

FM2 (76,0 MHz to 90,0 MHz) frequency:

code	***meaning***
0 - 140:	FM frequencies 76,0 MHz - 90,0 MHz (100 kHz step)
255	FM available, frequency not signalled

DAB [3] frequency:

code	*meaning*
0 - 11:	DAB channels 2A - 4D (Band I)
64 - 95:	DAB channels 5A - 12D (Band III)
96 - 101:	DAB channels 13A - 13F (Band III +)
128 - 140:	DAB channels (L-Band, European grid)
160 - 182:	DAB channels (L-Band, Canadian grid)
255:	DAB available, frequency not signalled

Use of frequency groups

The coverage area of the DRM multiplex may be divided into one or more geographic areas. The areas may overlap at their edges or they may be distinct. Transmissions that are receivable within each area are signalled as members of the same group.

6.4.3.5 Frequency schedule data entity - type 4

This entity allows a frequency schedule to be transmitted. This data entity uses the list mechanism for the version flag. This information is as follows:

-	Short Id	2 bits
-	enhancement flag	1 bit
-	day code	7 bits
-	start time	11 bits
-	end time	11 bits
-	frequency list Id	4 bits

The following definitions apply:

Short Id: this field indicates the short Id for the service concerned

enhancement flag: this flag indicates whether the frequency list applies to the base layer or the enhancement layer channel as follows:

0: information refers to the base layer channel

1: information refers to the enhancement layer channel

day code: this field indicates which days the frequency schedule applies to. The msb indicates Monday, the lsb Sunday

start time: this field indicates the time from when the frequency is valid. The time is expressed in hours (5 bits) and minutes (6 bits) in UTC

end time: this field indicates the time until when the frequency is valid. The time is expressed in hours (5 bits) and minutes (6 bits) in UTC

frequency list Id: this field indicates the frequency list which applies

6.4.3.6 Application information data entity - type 5

All data services (or data applications for audio services) are described by this data entity. Additional information regarding the handling of data services is given in TS 101 968 [11].

Many applications may require additional data to describe them that is specific to that application. This data entity uses the reconfiguration mechanism for the version flag. The content is described by the appropriate application specification. The general form of the entity is as follows:

-	Short Id	2 bits
-	Stream Id	2 bits
-	Packet mode indicator	1 bit
-	descriptor	7 or 15 bits
-	application data	*n* bytes

The following definitions apply:

Short Id: this field indicates the short Id for the service concerned

Stream Id: this field indicates the stream Id of the stream which carries the data service (or data application) concerned

Packet mode indicator: this field indicates whether the service is carried in packet mode or not as follows:

0: synchronous stream mode

1: packet mode

descriptor: the format of this field depends upon the value of the Packet mode indicator field as follows:

when **Packet mode indicator** = 0:

-	rfa	3 bits
-	application domain	4 bits

rfa: these 3 bits are reserved for future additions and shall be set to zero until they are defined

application domain: this field indicates the source of the data application specification. The interpretation of this field is given in TS 101 968 [11].

when **Packet mode indicator** = 1:

-	data unit indicator	1 bit
-	packet Id	2 bits
-	application domain	4 bits
-	packet length	8 bits

data unit indicator: this field indicates whether the data stream is composed of single packets or data units as follows:

0: single packets

1: data units

packet Id: this field indicates the Packet Id carried in the header of packets intended for this service

application domain: this field indicates the source of the data application specification. The interpretation of this field is given in TS 101 968 [11]

packet length: this field indicates the length in bytes of the data field of each packet specified as an unsigned binary number (the total packet length is three bytes longer as it includes the header and CRC fields)

NOTE: All packets contained in one data stream shall have the same length (see clause 6.6.4).

application data: this field of variable length is defined by the data service (or data application) specification. The interpretation of this field is given in TS 101 968 [11].

6.4.3.7 Announcement support and switching entity - type 6

This feature indicates which types of announcements are supported by the tuned multiplex or by another multiplex. This data entity uses the unique mechanism for the version flag. The following information is necessary:

-	Short Id	2 bits
-	rfu	1 bit
-	same/other multiplex	1 bit
-	Announcement support flags	8 bits
-	Announcement switching flags	8 bits
-	if "same/other multiplex" = other: Service Id	24 bits

The following definitions apply:

Short Id: this field contains the short Id that relates the information to the Service Id provided by the FAC

rfu: this bit is reserved for future use and shall be set to zero until defined

Same/other multiplex: this flag signals if the announcements are provided by a service in the multiplex being received or by a service in another multiplex, as follows:

0: multiplex being received

1: other multiplex

Announcement support flags: this 8-bit field specifies the types of announcements that are provided by this service, either directly or by vectoring to another service, as follows:

B_i (i = 0,...,7)

0: Announcement type not provided

1: Announcement type provided

The meaning of each bit is as follows:

b_0: Travel

b_1: News flash

b_2: Weather flash

b_3: Warning/Alarm

b_4 - b_7: reserved for future definition

Announcement switching flags: this 8-bit field specifies the announcement types that apply to the announcement. The individual bits indicate whether or not a particular announcement type is signalled. The flags are coded as follows:

B_i (i = 0,...,7)

0: Announcement type not valid

1: Announcement type valid

The meaning of each bit is as defined for the Announcement support flags above.

Service Id: if the announcements are provided by another multiplex, this 24-bit field signals the Service Identifier.

6.4.3.8 Region definition data entity - type 7

A frequency group can be given a geographical area. This data entity uses the list mechanism for the version flag. The area is defined in terms of multiples of 1 × 1 degree "squares". It therefore gives a resolution of (EW × NS) 111 km × 111 km (at equator) or 31 km × 111 km at 70° latitude (e.g. Scandinavia, Canada). The coding provided allows for the signalling of squares of up to about 4 000 km × 4 000 km for < 73° latitude. The area may in addition be defined in terms of CIRAF zones. Up to seven zones may be specified per group.

- group Id	4 bits
- number of zones, *n*	3 bits
- latitude	8 bits
- longitude	9 bits
- latitude extent	5 bits
- longitude extent	7 bits
- *n* CIRAF zones	*n* × 8 bits

The following definitions apply:

Group Id: this field indicates an identifier for a set of frequencies

number of zones: this field, coded as an 8 bit unsigned binary number, specifies the number of CIRAF zones to follow in the range 0 to 7

Latitude: this field specifies the southerly point of the area in degrees, as a 2's complement number

Longitude: this field specifies the westerly point of the area in degrees, as a 2's complement number

Latitude extent: this field specifies the size of the area to the north, in 1 degree steps

Longitude extent: this field specifies the size of the area to the east, in 1 degree steps

CIRAF zone: this field, coded as an 8 bit unsigned binary number in the range 1 to 85, gives one CIRAF zone

6.4.3.9 Time and date information data entity - type 8

The current time and date can be specified to allow a receiver to follow frequency schedules, etc. This data entity uses the unique mechanism for the version flag. The data entity is coded as follows:

- Modified Julian Date	17 bits
- UTC (hours and minutes)	11 bits

The following definitions apply:

Modified Julian Date: this field indicates the date in MJD format

UTC: this field specifies the current time expressed in hours (5 bits) and minutes (6 bits)

When the time and date are signalled, this data entity shall be sent in the first SDC block after the minute's edge.

6.4.3.10 Audio information data entity - type 9

Each audio service needs a detailed description of the parameters needed for audio decoding. This data entity uses the reconfiguration mechanism for the version flag.

- Short Id	2 bits
- Stream Id	2 bits
- audio coding	2 bits
- SBR flag	1 bit
- audio mode	2 bits
- audio sampling rate	3 bits
- text flag	1 bit
- enhancement flag	1 bit
- coder field	5 bits
- rfa	1 bit

The following definitions apply:

short Id: this field indicates the short Id for the service concerned

Stream Id: this field indicates the stream Id of the stream that carries the service concerned

audio coding: this field indicated the source coding system as follows:

00: AAC

01: CELP

10: HVXC

11: reserved

SBR flag: this field indicates whether SBR is used or not as follows:

0: SBR not used

1: SBR used

audio mode: this field depends upon the value of the audio coding field as follows:

audio coding field = 00:

00: mono

01: lc stereo

10: stereo

11: reserved

audio coding field = 01:

- rfa	1 bit
- CELP_CRC	1 bit

audio coding field = 10:

- HVXC_rate	1 bit
- HVXC_CRC	1 bit

CELP_CRC: this field indicates whether the CRC is used or not:

0: CRC not used

1: CRC used

HVXC_rate: this field indicates the rate of the HVXC:

0: 2 kbit/s

1: 4 kbit/s

HVXC_CRC: this field indicates whether the CRC is used or not:

0: CRC not used

1: CRC used

audio sampling rate: this field indicates the audio sampling rate as follows:

000: 8 kHz

001: 12 kHz

010: 16 kHz

011: 24 kHz

100: reserved

101: reserved

110: reserved

111: reserved

text flag: this field indicates whether a text message is present or not as follows:

0: no text message is carried

1: text message is carried (see clause 6.5)

enhancement flag: this field indicates whether audio enhancement data is available in another channel as follows:

0: no enhancement available

1: enhancement is available

coder field: this field depends upon the value of the audio coding field as follows:

audio coding field = 00 or 10:

- rfa	5 bits

audio coding field = 01:

- CELP_index	5 bits

CELP_index: this field indicates the CELP bit rate index, as defined in tables 10 and 11 (see clause 5)

rfa: these 1-bit and 5-bit fields are reserved for future additions and shall be set to zero until they are defined

6.4.3.11 FAC channel parameters data entity - type 10

This data entity permits the next configuration FAC channel parameters to be specified in advance for service following across reconfigurations. This data entity uses the reconfiguration mechanism for the version flag. The fields are as follows:

-	Base/Enhancement flag	1 bit
-	Robustness mode	2 bits
-	Spectrum occupancy	4 bits
-	Interleaver depth flag	1 bit
-	MSC mode	2 bits
-	SDC mode	1 bit
-	number of services	4 bits
-	rfa	3 bits
-	rfu	2 bits

The following definitions apply:

Base/Enhancement flag: this 1-bit flag indicates whether the transmission is the base or enhancement layer as follows:

0: Base layer - decodable by all DRM receivers

1: Enhancement layer - only decodable by receivers with enhancement layer capabilities

Robustness mode: this field indicates the robustness mode of the new configuration as follows:

00: A

01: B

10: C

11: D

Spectrum occupancy: specifies the configuration of the digital signal (see clause 8)

Interleaver depth flag: indicates the depth of the time interleaving as follows:

0: 2 s (long interleaving)

1: 400 ms (short interleaving)

MSC mode: indicates the modulation mode in use for the MSC as follows:

00: 64-QAM, no hierarchical

01: 64-QAM, hierarchical on I

10: 64-QAM, hierarchical on I&Q

11: 16-QAM, no hierarchical

SDC mode: indicates the modulation mode in use for the SDC as follows:

0: 16-QAM

1: 4-QAM

Number of services: indicates the number of audio and data services as follows:

0000: 4 audio services

0001: 1 data service

0010: 2 data services

0011: 3 data services

0100: 1 audio service

0101: 1 audio service and 1 data service

0110: 1 audio service and 2 data services

0111: 1 audio service and 3 data services

1000: 2 audio services

1001: 2 audio services and 1 data service

1010: 2 audio services and 2 data services

1011: reserved

1100: 3 audio services

1101: 3 audio services and 1 data service

1110: reserved

1111: 4 data services

rfa: these 3 bits are reserved for future additions and shall be set to zero until they are defined

rfu: these 2 bits are reserved for future use and shall be set to zero until they are defined

If the DRM transmission is being discontinued at the reconfiguration, then this data entity shall be sent with the length field of the header set to 0, and the first four bits of the body field set to 0.

6.4.3.12 Linkage data entity - type 11

The linkage data entity allows the connections between alternative sources of the same service to be signalled. This data entity uses the list mechanism for the version flag. The information is as follows:

-	Short Id	2 bits
-	number DRM Service Ids, *d*	3 bits
-	number RDS PI codes, *p*	3 bits
-	programme/data flag	1 bit
-	number of DAB SIds, *s*	3 bits
-	*d* Service Ids	*d* × 24 bits
-	*p* PI codes	*p* × 16 bits
-	*s* SId codes	*s* × 16 bits or 32 bits

The following definitions apply:

Short Id: this field indicates the short Id for the service concerned

number of DRM Service Ids: this field indicates the number of Service Ids to follow in the range 0 to 7

number of RDS PI [4] codes: this field indicates the number of PI codes to follow in the range 0 to 7

number of DAB SId codes: this field indicates the number of SId codes to follow in the range 0 to 7

Service Ids: this field indicates the list of Service Ids on which the DRM service indicated by the Short Id field is also carried

PI code: this field indicates the list of PI codes on which the DRM service indicated by the Short Id field is also carried. Note that FM services without RDS/RBDS shall be given a dummy PI code where the most significant nibble is set to 0

programme/data flag: this field indicates whether the following DAB SIds are programme or data services as follows:

0: programme service: DAB SIds are 16 bits long

1: data service: DAB SIds are 32 bits long

SId code: this field indicates the list of SId codes on which the DRM service indicated by the Short Id field is also carried

6.4.3.13 Language and country data entity - type 12

The language and country data entity allows addition language and country information to be signalled. This data entity uses the unique mechanism for the version flag. The information is as follows:

-	Short Id	2 bits
-	rfu	2 bits
-	language code	24 bits
-	country code	16 bits

The following definitions apply:

Short Id: this field indicates the short Id for the service concerned

rfu: these 2 bits are reserved for future use and shall be set to zero until they are defined

Language code: this 24-bit field identifies the language of the target audience of the service according to ISO 639-2 [6] using three characters as specified by ISO 8859-1 [8]. If the language is not specified, the field shall contain three "-" characters

Country code: this 16-bit field identifies the country of origin of the service (the site of the studio) according to ISO 3166 [7] using two lower case characters as specified by ISO 8859-1 [8]. If the country code is not specified, the field shall contain two "-" characters

6.4.3.14 Other data entities

Other data entities are reserved for future definition.

6.4.4 Summary of data entity characteristics

Table 62 summarizes the version flag mechanism, repetition rate and transmission status of each data entity. The standard repetition rate is that all information for that data entity type should be transmitted within one cycle of the entire database. Individual SDC blocks may carry changed information (e.g. time and date) by use of the FAC identity field.

Table 62: Summary of data entity characteristics

Data entity	Version flag mechanism	Repetition rate	Transmission status
0	reconfiguration	every SDC block	mandatory
1	unique	every SDC block	optional
2	reconfiguration	as required	mandatory for each service for which the FAC CA indication flag = 1
3	list	standard	optional
4	list	standard	optional
5	reconfiguration	as required	mandatory for each data service and data application
6	unique	as required	optional
7	list	standard	optional
8	unique	once per minute	optional
9	reconfiguration	every SDC block	mandatory for each audio service
10	reconfiguration	every SDC block when FAC reconfiguration index is non-zero	mandatory when FAC reconfiguration index is non-zero
11	list	standard	optional
12	unique	standard	optional

Table 62 gives the recommended repetition rate for fast access to services. However, when the SDC capacity (see clause 6.4.2) is low, lower repetition rates are permitted for every data entity.

6.4.5 Changing the content of the SDC

The content of the SDC is important for the operation of alternative frequency checking and switching (AFS). For AFS to function, the receiver must know what the content of the SDC is in advance so that a correlation may be performed. For this purpose, the AFS index is provided in the SDC and the FAC validates the index by use of the Identity field.

On transmissions with no alternative frequencies, the content of the SDC can be fully dynamic and changed at will: no AFS function is required. In this case it is recommended that the AFS index should be set to 0, and the Identity field in the FAC should then indicate the sequence 11, 01, 10 etc. to indicate that the AFS function cannot be performed.

On transmissions with alternative frequencies, the assignment of data entities to SDC blocks should be carefully designed in order that the content of the SDC can be as static as possible thereby permitting use of the AFS function. In this case it is recommended that the AFS index is chosen such that all required information can be sent in one cycle of SDC blocks. If the content is completely static then the Identity field in the FAC indicates the sequence 00, 01, 10, etc. which indicates that the AFS function can be performed at every position, provided the receiver has stored the data for all the SDC blocks in the cycle.

When the Time and date data entity or announcement support and switching data entity is included in the SDC, and alternative frequencies are signalled, then a semi-dynamic use of the SDC is recommended. In this case one or more SDC blocks in the cycle defined by the AFS index are signalled to be invalid by use of the FAC Identity field thereby allowing the content of those blocks to be changed continuously, whilst other SDC blocks are always signalled as valid by use of the FAC Identity field thereby allowing the AFS function to be performed. An example of changing the SDC content and of using the semi-dynamic scheme with the AFS index = 1 is given in annex G.

A change of the AFS index is only allowed at reconfiguration.

6.4.6 Signalling of reconfigurations

Reconfiguration of the DRM multiplex shall be signalled in advance in order to permit receivers to make the best decisions about how to handle the changes. There two types of reconfiguration: a service reconfiguration, which concerns the reallocation of the data capacity between the services of the MSC; and a channel reconfiguration, which concerns changes to the overall capacity of the MSC.

Both types of reconfiguration are signalled by setting the FAC reconfiguration index to a non-zero value. The index then counts down on each subsequent transmission super frame. The reconfiguration index shall be identical for all three transmission frames of a transmission super frame. The final transmission super frame corresponding to the current configuration shall be that in which the reconfiguration index = 1. The new configuration takes effect for the next transmission super frame and in which the reconfiguration index = 0. Ideally the reconfiguration should be signalled as far in advance as possible (i.e. the reconfiguration index should first take the value 7) in order to provide the greatest chance that the receiver gets all the information necessary for the next configuration.

All data entity types that use the reconfiguration mechanism for the version flag that are present in the current configuration, and all data entity types that use the reconfiguration mechanism for the version flag that are required in the new configuration, shall be sent during the period when the reconfiguration index is non-zero with the version flag indicating the next configuration. This shall include data entity type 10 that signals the FAC channel parameters for the new configuration.

6.4.6.1 Service reconfigurations

A service reconfiguration is one in which the data capacity of the MSC is reallocated between services. This happens when the number of services in the multiplex is changed or the size of data streams is changed. A service reconfiguration shall also be signalled if any of the content of the data entity types using the reconfiguration mechanism of the version flag changes.

When a new service is introduced, and the overall capacity of the MSC is not changed, then the receiver shall follow the currently selected service through the reconfiguration. To facilitate this, the Service Identity and Short Id of all continuing services shall remain the same. The new service shall use a Short Id that is not used in the current configuration. The one exception to this rule is if there are four services in the current configuration and four services in the new configuration. In this case, if the currently selected service is discontinued, then the receiver follows to the new service with the same Short Id if it is of the same type (e.g. both are audio services).

If the currently selected service is discontinued at the reconfiguration, then the receiver may try to find another source of that service on another frequency and/or system by using the information from data entity types 3 and 11.

6.4.6.2 Channel reconfigurations

A channel reconfiguration is one in which the following FAC channel parameters are altered: spectrum occupancy, interleaver depth and MSC mode; and when the robustness mode is changed. In this case the receiver is unable to follow the currently selected service without disruption to the audio output.

If the transmission is discontinued on the tuned frequency, then a reconfiguration shall be signalled with data entity type 10 taking a special value (see clause 6.4.3.11). In this specific case, the other data entity types that use the reconfiguration mechanism for the version flag shall not be signalled.

6.5 Text message application

Text messages can provide a highly valuable additional element to an audio service without consuming much data capacity. The text message is a basic part of DRM and consumes only 80 bits/s. This capacity can be saved if the service provider does not use text messaging.

6.5.1 Structure

The text message (when present) shall occupy the last four bytes of the lower protected part of each logical frame carrying an audio stream. The message is divided into a number of segments and UTF-8 character coding is used. The beginning of each segment of the message is indicated by setting all four bytes to the value 0xFF.

When no text message is available for insertion all four bytes shall be set to 0x00.

The text message may comprise up to 8 segments. Each segment consists of a header, a body and a CRC. The body shall contain 16 bytes of character data unless it is the last segment in which case it may contain less than 16 bytes.

Each segment is further divided into four-byte pieces which are placed into each successive frame. If the length of the last segment is not a multiple of four then the incomplete frame shall be padded with 0x00 bytes.

The structure of the segment is as follows:

Header	16 bits
Body	n × 8 bits
CRC	16 bits

The Header is made up as follows:

- toggle bit	1 bit
- first flag	1 bit
- last flag	1 bit
- command flag	1 bit
- field 1	4 bits
- field 2	4 bits
- rfa	4 bits

The following definitions apply:

Toggle bit: this bit shall be maintained in the same state as long as segments from the same message are being transmitted. When a segment from a different text message is sent for the first time, this bit shall be inverted with respect to its previous state. If a text message, which may consist of several segments, is repeated, then this bit shall remain unchanged.

First flag, Last flag: these flags are used to identify particular segments which form a succession of segments in a text message. The flags are assigned as follows:

First flag	Last flag	The segment is:
0	0	: an intermediate segment;
0	1	: the last segment;
1	0	: the first segment;
1	1	: the one and only segment.

Command flag: this 1-bit flag signals whether Field 1 contains the length of the body of the segment or a special command, as follows:

0: Field 1 signals the **length** of the body of the segment

1: Field 1 contains a special **command**

Field 1:

- **Length:** this 4-bit field, expressed as an unsigned binary number, specifies the number of bytes in the body minus 1. It shall normally take the value 15 except in the last segment.
- **Command:** this 4-bit field contains a special command, as follows (all other codes are reserved for future use):

  0 0 0 1: the message shall be removed from the display.

Field 2:

- if **First flag** = "1":

  this field contains the value "1111";

- if **First flag** = "0":

  Rfa: this 1-bit field is reserved for future additions. The bit shall be set to zero until it is defined.

  SegNum (Segment number)**:** this 3-bit field, expressed as an unsigned binary number, specifies the sequence number of the current segment minus 1. (The second segment of a label corresponds to SegNum = 1, the third segment to SegNum = 2, etc.) The value 0 is reserved for future use.

Rfa: this 4-bit field is reserved for future additions. These bits shall be set to zero until they are defined.

Body: this field shall be coded as a string of characters (maximum 16). If the last character of a message segment is a multibyte character and not all bytes fit into the body then the character shall continue in the next message segment. This field shall be omitted when the C flag = "1" (special command).

The following additional codes may be used:

- Code 0x0A may be inserted to indicate a preferred line break.
- Code 0x0B may be inserted to indicate the end of a headline. Headlines shall be restricted to a maximum length of 2 × 16 displayable characters (including hyphens introduced as a result of a control code 0x1F) and may contain 0 or 1 preferred line breaks codes (Code 0x0A). There may not be more than sixteen characters before any line-break and there may not be more than sixteen characters after any line-break.
- Code 0x1F (hex) may be inserted to indicate a preferred word break. This code may be used to display long words comprehensibly.

CRC (Cyclic Redundancy Check): this 16-bit CRC shall be calculated on the header and the body. It shall use the generator polynomial $G_{16}(x) = x^{16} + x^{12} + x^5 + 1$.

6.6 Packet mode

Data services generally consist of either streams of information, in either synchronous or asynchronous form, or files of information. A generalized packet delivery system allows the delivery of asynchronous streams and files for various services in the same data stream and allows the bit rate of the (synchronous) data stream to be shared on a frame-by-frame basis between the various services. Services can be carried by a series of single packets or as a series of data units. A data unit is a series of packets that are considered as one entity with regard to error handling - one received errored packet within a data unit causes the whole data unit to be rejected. This mechanism can be used to transfer files and also to allow simpler synchronization of asynchronous streams. The carriage of data applications is described in TS 101 968 [11].

The size of a packet mode data logical frame shall be a multiple of the packet size. The maximum length of a data unit is 8 215 bytes.

6.6.1 Packet structure

The packet is made up as follows:

header	8 bits
data field	*n* bytes
CRC	16 bits

The **header** contains information to describe the packet.

The **data field** contains the data intended for a particular service. The length of the data field is indicated by use of data entity 5, see clause 6.4.3.6.

The **CRC** (Cyclic Redundancy Check): this 16-bit CRC shall be calculated on the header and the data field. It shall use the generator polynomial $G_{16}(x) = x^{16} + x^{12} + x^5 + 1$ (see annex D).

6.6.1.1 Header

The header consists of the following fields:

first flag	1 bit
last flag	1 bit
packet Id	2 bits
padded packet indicator (PPI)	1 bit
continuity index (CI)	3 bits

The following definitions apply:

First flag, Last flag: these flags are used to identify particular packets which form a succession of packets. The flags are assigned as follows:

First flag	Last flag	The packet is:
0	0	: an intermediate packet;
0	1	: the last packet of a data unit;
1	0	: the first packet of a data unit;
1	1	: the one and only packet of a data unit.

Packet Id: this 2-bit field indicates the Packet Id of this packet.

Padded Packet Indicator: this 1-bit flag indicates whether the data field carries padding or not, as follows:

0: no padding is present: all data bytes in the data field are useful;

1: padding is present: the first byte gives the number of useful data bytes in the data field.

Continuity index: this 3-bit field shall increment by one modulo-8 for each packet with this packet Id.

6.6.1.2 Data field

The data field contains the useful data intended for a particular service.

If the Padded Packet Indicator (PPI) field of the header is 0, then all bytes of the data field are useful bytes.

If the PPI is 1 then the first byte indicates the number of useful bytes that follow, and the data field is completed with padding bytes of value 0x00.

Packets with no useful data are permitted if no packet data is available to fill the logical frame. The PPI shall be set to 1 and the first byte of the data field shall be set to 0 to indicate no useful data. The first and last flags shall be set to 1. The continuity index shall be incremented for these empty packets. If less than 4 sub-streams are used within the data stream then an unused packet id shall be used. Empty packets using a packet id of <p> shall not be inserted during the transmission of a DRM data unit using the same packet id <p>.

6.6.2 Asynchronous streams

Asynchronous streams can be used to transport byte-oriented information. Both single packets and data units can be used to transport asynchronous streams.

Applications that use the single packet transport mechanism shall be able to deal with missing data packets. The first and last flags indicate intermediate packets.

Applications that use the data unit transport mechanism can carry a collection of bytes that are related in a data unit and then make use of the error handling of data units for synchronization purposes.

6.6.3 Files

The file may be carried in a data unit.

Applications that use this transport mechanism shall provide a mechanism to identify each object.

The first and last flags are used to indicate the series of packets that make up the data unit. The continuity index is used to determine whether any intermediate packets have been lost.

6.6.4 Choosing the packet length

A data stream for packet mode may contain one or more packets per logical frame, and the packets may belong to one or more services. However, all packets contained in the stream shall have the same length to minimize the propagation of errors. The choice of the packet length depends on various factors, but the following should be taken into account:

- The overhead of signalling the header and CRC is fixed per packet. Therefore the larger the packet, the lower the ratio of overhead to useful data.
- The amount of padding carried in packets is related to the size of the files compared to the packet size or the transit delay requirements for asynchronous streams. Large packets are less efficient at transporting many small objects.

7 Channel coding and modulation

7.1 Introduction

The DRM system consists of three different channels, the MSC, SDC and FAC. Because of the different needs of these channels different coding and mapping schemes shall be applied. An overview of the encoding process is shown in figure 19.

The coding is based on a multilevel coding scheme for which the principle is explained in clause 7.3. Due to different error protection needs within one service or for different services within one multiplex different mapping schemes and combinations of code rates are applicable: Unequal Error Protection (UEP) and Equal Error Protection (EEP) are available and can be combined with hierarchical modulation. Equal error protection uses a single code rate to protect all the data in a channel. EEP is mandatory for the FAC and SDC. Instead of EEP, unequal error protection can be used with two code rates to allow the data in the Main Service Channel to be assigned to the higher protected part and the lower protected part. When using hierarchical modulation three mapping strategies are applicable to the MSC: the Standard Mapping (SM), the symmetrical Hierarchical Mapping (HMsym) and a mixture of the previous two mappings (HMmix) that results in the real component of the constellation following a Hierarchical Mapping and the imaginary part following a standard one. The Hierarchical Mappings split the decodable data stream into two parts: a Very Strongly Protected Part (VSPP) and a Standard Protected Part (SPP). The SM method only consists of a SPP. In any case, up to two different overall code rates shall be applied to the SPP of the MSC. For the FAC and SDC only SM is allowed. The application of the coding to the different channels is described in clause 7.5.

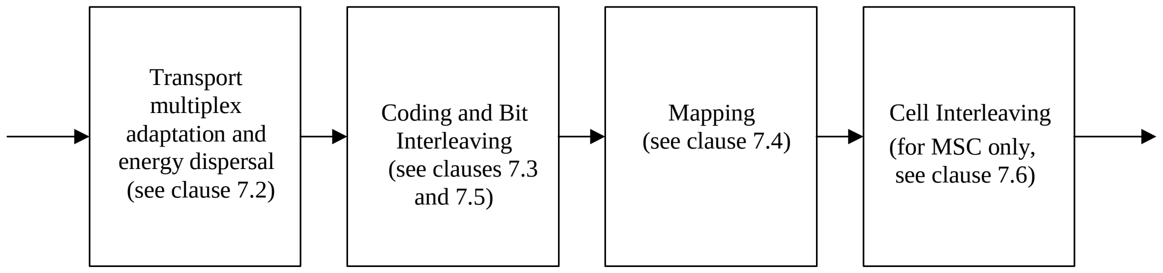

Transport multiplex adaptation and energy dispersal (see clause 7.2)
Coding and Bit Interleaving (see clauses 7.3 and 7.5)
Mapping (see clause 7.4)
Cell Interleaving (for MSC only, see clause 7.6)

Figure 19: Functional block diagram of the coding and interleaving

7.2 Transport multiplex adaptation and energy dispersal

7.2.1 Transport multiplex adaptation

The different channels (MSC, SDC, FAC) are processed in the channel coding independently. The vector length L for processing equals one FAC block for the FAC, one SDC block for the SDC or one multiplex frame for the MSC.

7.2.1.1 MSC

The number of bits L_{MUX} per multiplex frame is dependent on the robustness mode, spectrum occupancy and constellation:

- when using one protection level (EEP) it is given by:

$$L_{MUX} = L_2$$

- when using two protection levels (UEP) it is given by:

$$L_{MUX} = L_1 + L_2$$

where the number of bits of the higher protected part is L_1 and the number of bits of the lower protected part is L_2.

- when using HMsym or HMmix the number of very strongly protected bits is given by L_{VSPP}.

L_1, L_2 and L_{VSPP} are calculated as follows:

SM:

$$L_1 = \sum_{p=0}^{P_{\max}-1} 2N_1 R_p$$

$$L_2 = \sum_{p=0}^{P_{\max}-1} RX_p \left\lfloor \frac{2N_2 - 12}{RY_p} \right\rfloor$$

$$L_{VSPP} = 0$$

P_{max} is the number of levels (4-QAM: P_{max} = 1; 16-QAM: P_{max} = 2; 64-QAM: P_{max} = 3).

RX_p is the numerator of the code rate of each individual level, see table 65.

RY_p is the denominator of the code rate of each individual level, see table 65.

R_p is the code rate of each individual level, see table 65.

HMsym:

$$L_1 = \sum_{p=1}^{2} 2N_1 R_p$$

$$L_2 = \sum_{p=1}^{2} RX_p \left\lfloor \frac{2N_2 - 12}{RY_p} \right\rfloor,$$

$$L_{VSPP} = RX_0 \left\lfloor \frac{2(N_1 + N_2) - 12}{RY_0} \right\rfloor$$

P_{max} =3 is the number of levels for 64-QAM using HMsym.

NOTE: A hierarchical mapping scheme can only be used in a 64-QAM signal constellation.

RX_p is the numerator of the code rate of each individual level, see table 65.

RY_p is the denominator of the code rate of each individual level, see table 65.

R_p is the code rate of each individual level, see table 65.

HMmix:

$$L_1 = N_1 R_0^{\mathrm{Im}} + \sum_{p=1}^{2} N_1 (R_p^{\mathrm{Re}} + R_p^{\mathrm{Im}})$$

$$L_2 = RX_0^{\mathrm{Im}} \left\lfloor \frac{N_2 - 12}{RY_0} \right\rfloor + \sum_{p=1}^{2} RX_p^{\mathrm{Re}} \left\lfloor \frac{N_2 - 12}{RY_p} \right\rfloor + RX_p^{\mathrm{Im}} \left\lfloor \frac{N_2 - 12}{RY_p} \right\rfloor$$

$$L_{VSPP} = RX_o^{\mathrm{Re}} \left\lfloor \frac{N_1 + N_2 - 12}{RY_0^{\mathrm{Re}}} \right\rfloor$$

$P_{\max} = 3$ is the number of levels for 64-QAM using HMmix

$RX_p^{\mathrm{Re}}, RX_p^{\mathrm{Im}}$ are the numerators of the code rates of each individual level (see table 65) for the real and imaginary component respectively.

$RY_p^{\mathrm{Re}}, RY_p^{\mathrm{Im}}$ are the denominators of the code rates of each individual level (see table 65) for the real and imaginary component respectively.

$R_p^{\mathrm{Re}}, R_p^{\mathrm{Im}}$ are the code rates of each individual level (see table 65) for the real and imaginary component respectively

and $\lfloor\ \rfloor$ means round towards minus infinity.

The total number N_{MUX} of MSC OFDM cells per multiplex frame is given in clause 7.7.

The total number N_{MUX} of MSC OFDM cells per multiplex frame when using one protection level (EEP) equals N_2.

The total number N_{MUX} of MSC OFDM cells per multiplex frame when using two protection levels (UEP) equals the addition of the cells of the higher protected part and the lower protected part:

$$N_{MUX} = N_1 + N_2$$

N_1 is the number of OFDM cells used for the higher protected part.

N_2 is the number of OFDM cells used for the lower protected part including the tailbits.

To calculate the number N_1 of OFDM cells in the higher protected part (part A) the following formulae apply:

SM:

$$N_1 = \left\lceil \frac{8X}{2RY_{lcm} \sum_{p=0}^{P_{\max}-1} R_p} \right\rceil RY_{lcm}$$

HMsym:

$$N_1 = \left\lceil \frac{8X}{2RY_{lcm} \sum_{p=1}^{2} R_p} \right\rceil RY_{lcm}$$

HMmix:

$$N_1 = \left\lceil \frac{8X}{RY_{lcm}\left(R_0^{\mathrm{Im}} + \sum_{p=1}^{2}\left(R_p^{\mathrm{Re}} + R_p^{\mathrm{Im}}\right)\right)} \right\rceil RY_{lcm}$$

where:

X is the number of bytes in part A (as signalled in the SDC);

RY_{lcm} is taken from tables 67 and 68 for SM; from tables 69 and 70 for HMsym; and from tables 68, 70 and 71 for HMmix.

$\lceil\ \rceil$ means round towards plus infinity.

To calculate the number N_2 of OFDM cells in the lower protected part (part B) the following formula applies:

$$N_2 = N_{MUX} - N_1$$

The following restrictions shall be taken into account:

$$N_1 \in \{0, \ldots N_{MUX} - 20\}$$

$$N_2 \in \{20, \ldots N_{MUX}\}$$

7.2.1.2 FAC

The number of bits L_{FAC} per FAC block equals 72 bits in every mode.

The total number N_{FAC} of FAC OFDM cells per FAC block equals 65 in every mode.

7.2.1.3 SDC

The number of bits L_{SDC} per SDC block is dependent on the robustness mode, spectrum occupancy and constellation.

The total number N_{SDC} of SDC OFDM cells per SDC block are given in table 63.

The formulas given in clause 7.2.1.1 for the MSC are valid also for the SDC under the constraint of EEP and SM (only 4-QAM: $P_{max} = 1$, 16-QAM: $P_{max} = 2$), i.e. $L_{SDC} + L_2$ and $N_{SDC} = N_2$.

Table 63: Number of QAM cells N_{SDC} for SDC

Robustness mode	Spectrum occupancy					
	0	1	2	3	4	5
A	167	190	359	405	754	846
B	130	150	282	322	588	662
C	-	-	-	288	-	607
D	-	-	-	152	-	332

7.2.2 Energy dispersal

The purpose of the energy dispersal is to avoid the transmission of signal patterns which might result in an unwanted regularity in the transmitted signal.

For the SDC and FAC, the output of the energy dispersal shall form the input stream u_i to the corresponding multilevel coding process.

The output of the energy dispersal acting on the MSC multiplex frame shall form the standard protected input stream u_i to the multilevel coding process for the MSC. The output of the energy dispersal acting on the hierarchical frame (if present) shall form the very strongly protected input stream u'_i to the same multilevel coding process.

Energy dispersal shall be applied on the different channels (MSC, SDC, FAC) in order to reduce the possibility that systematic patterns result in unwanted regularity in either the transmitted signal or in any digital processing, this by providing a deterministic selective complementing of bits.

The individual inputs of the energy dispersal scramblers shown in figure 20 shall be scrambled by a modulo-2 addition with a pseudo-random binary sequence (PRBS), prior to channel encoding.

The PRBS is defined as the output of the feedback shift register of figure 20. It shall use a polynomial of degree 9, defined by:

$$P(X) = X^9 + X^5 + 1$$

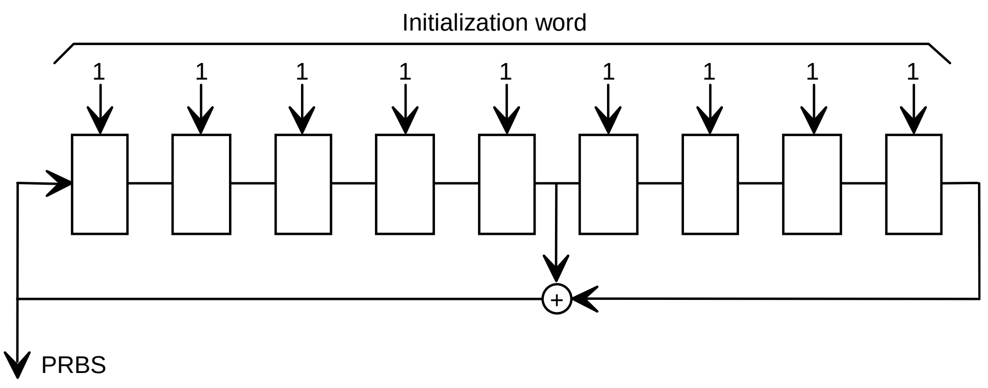


Figure 20: PRBS generator

The initialization word shall be applied in such a way that the first bit of the PRBS is obtained when the outputs of all shift register stages are set to value "1"; the first 16 bits of the PRBS are given in table 64.

Table 64: First 16 bits of the PRBS

bit index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bit value	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0

The FAC, SDC and MSC shall be processed by the energy dispersal scramblers as follows:

- The vector length for processing equals one FAC block for the FAC, one SDC block for the SDC and one multiplex frame and one hierarchical frame for the MSC.
- Each FAC block consists of 72 bits, the block lengths for the SDC and MSC are dependent on the robustness mode, spectrum occupancy and constellation, see clause 7.2.1.
- The four blocks shall be processed independently. The input vector shall be scrambled with the PRBS, the first bit of the vector being added modulo 2 to the PRBS bit of index 0.

The scramblers of the different channels are reset as follows:

- FAC: every FAC block;
- SDC: every SDC block;
- MSC: every multiplex frame for the standard protected part, every hierarchical frame for the very strongly protected part.

7.3 Coding

Clause 7.3.1 explains the structure of multilevel coding for the different mappings, clause 7.3.2 defines the component code of the multilevel coding scheme and clause 7.3.3 the corresponding bit-wise interleaving.

7.3.1 Multilevel coding

The channel encoding process is based on a multilevel coding scheme. The principle of multilevel coding is the joint optimization of coding and modulation to reach the best transmission performance. This denotes that more error prone bit positions in the QAM mapping get a higher protection. The different levels of protection are reached with different component codes which are realized with punctured convolutional codes, derived from the same mother code.

The decoding in the receiver can be done either straightforwardly or through an iterative process. Consequently the performance of the decoder with errored data can be increased with the number of iterations and hence is dependent on the decoder implementation.

Depending on the signal constellation and mapping used, five different schemes are applicable. The 1-level scheme shall be considered as a special case of the multilevel coding scheme. Different mapping schemes are only applicable to the 64-QAM constellation as depicted in figures 27, 28 and 29. For the standard mapping and symmetrical hierarchical modulation (SM and HMsym), identical mappings shall be used for the real and imaginary components of the signal constellation. For the mixed mapping hierarchical modulation (HMmix) separate mappings shall be used for the real and imaginary components of the signal constellation.

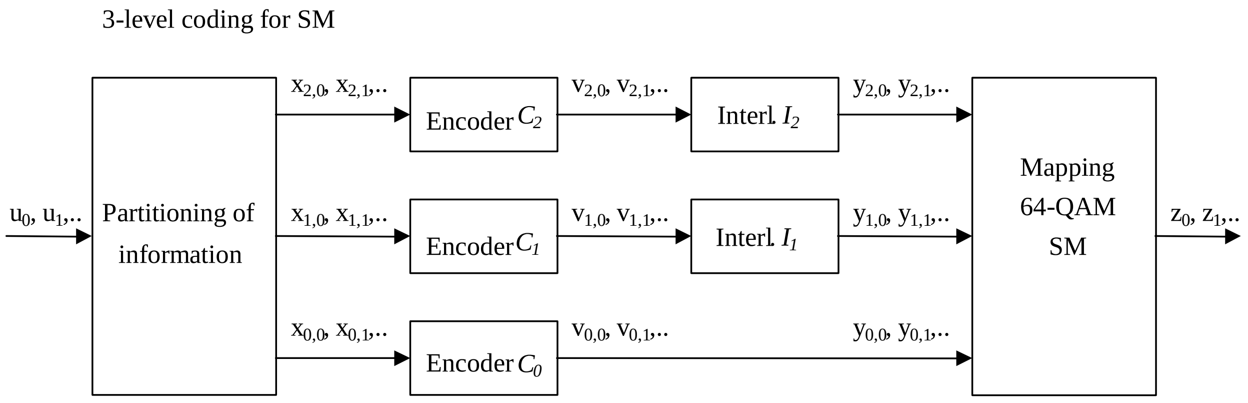


Figure 21: Multilevel coding with 3 levels for SM

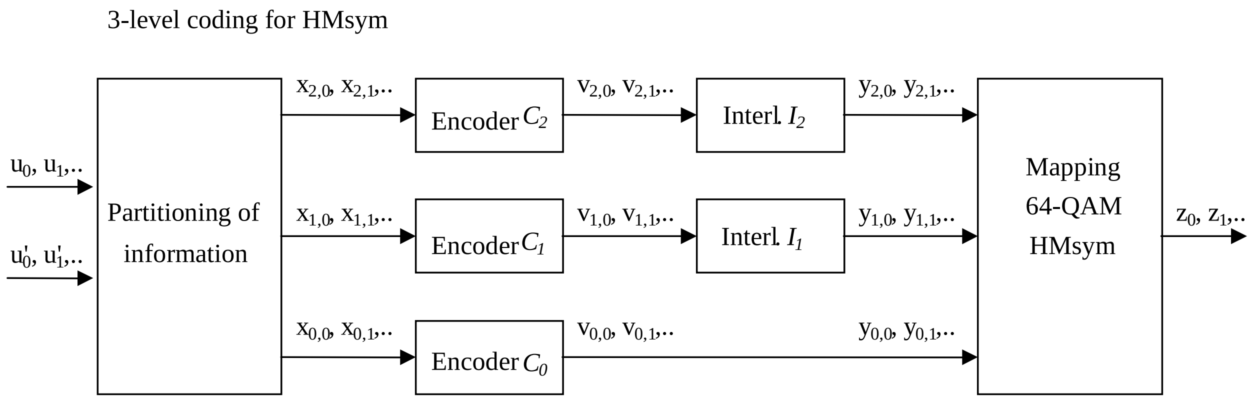


Figure 22: Multilevel coding with 3 levels for HMsym

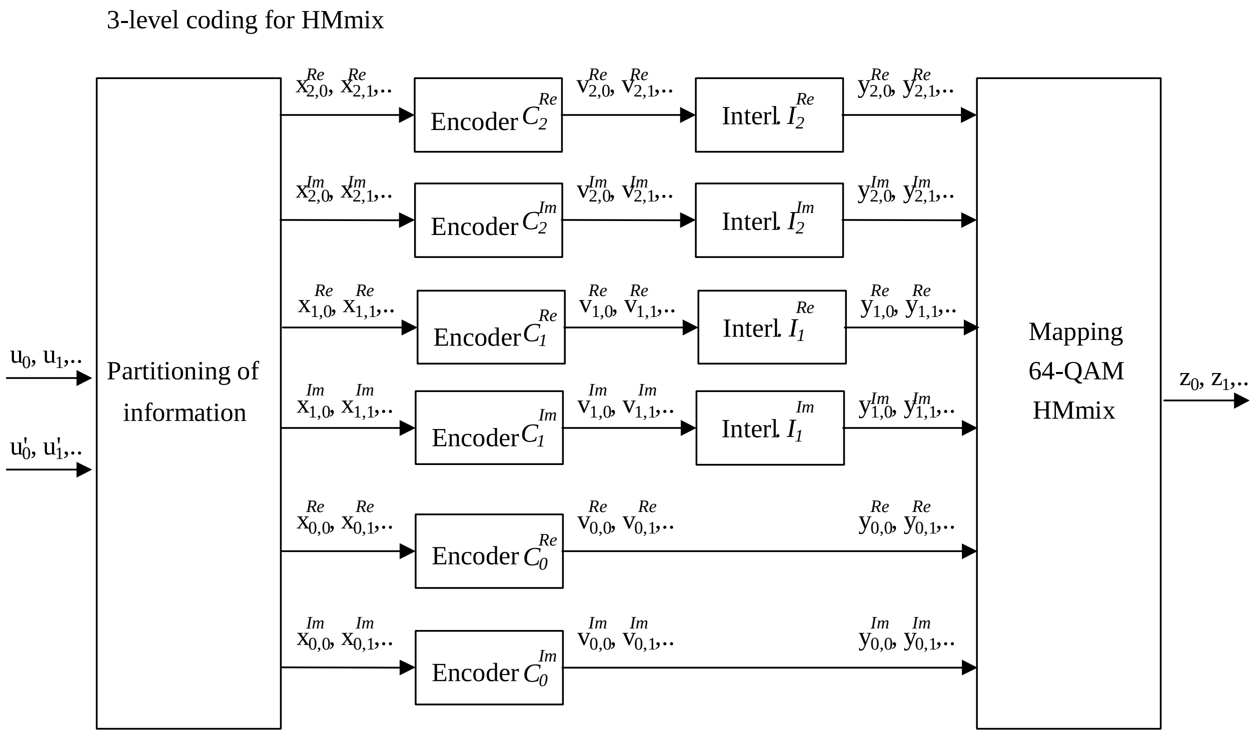

3-level coding for HMmix
$u_0, u_1, ..$
$u'_0, u'_1, ..$
Partitioning of information
$x^{Re}_{2,0}, x^{Re}_{2,1}, ..$
$x^{Im}_{2,0}, x^{Im}_{2,1}, ..$
$x^{Re}_{1,0}, x^{Re}_{1,1}, ..$
$x^{Im}_{1,0}, x^{Im}_{1,1}, ..$
$x^{Re}_{0,0}, x^{Re}_{0,1}, ..$
$x^{Im}_{0,0}, x^{Im}_{0,1}, ..$
Encoder C^{Re}_2
Encoder C^{Im}_2
Encoder C^{Re}_1
Encoder C^{Im}_1
Encoder C^{Re}_0
Encoder C^{Im}_0
$v^{Re}_{2,0}, v^{Re}_{2,1}, ..$
$v^{Im}_{2,0}, v^{Im}_{2,1}, ..$
$v^{Re}_{1,0}, v^{Re}_{1,1}, ..$
$v^{Im}_{1,0}, v^{Im}_{1,1}, ..$
$v^{Re}_{0,0}, v^{Re}_{0,1}, ..$
$v^{Im}_{0,0}, v^{Im}_{0,1}, ..$
Interl. I^{Re}_2
Interl. I^{Im}_2
Interl. I^{Re}_1
Interl. I^{Im}_1
$y^{Re}_{2,0}, y^{Re}_{2,1}, ..$
$y^{Im}_{2,0}, y^{Im}_{2,1}, ..$
$y^{Re}_{1,0}, y^{Re}_{1,1}, ..$
$y^{Im}_{1,0}, y^{Im}_{1,1}, ..$
$y^{Re}_{0,0}, y^{Re}_{0,1}, ..$
$y^{Im}_{0,0}, y^{Im}_{0,1}, ..$
Mapping 64-QAM HMmix
$z_0, z_1, ..$

Figure 23: Multilevel coding with 3 levels for HMmix

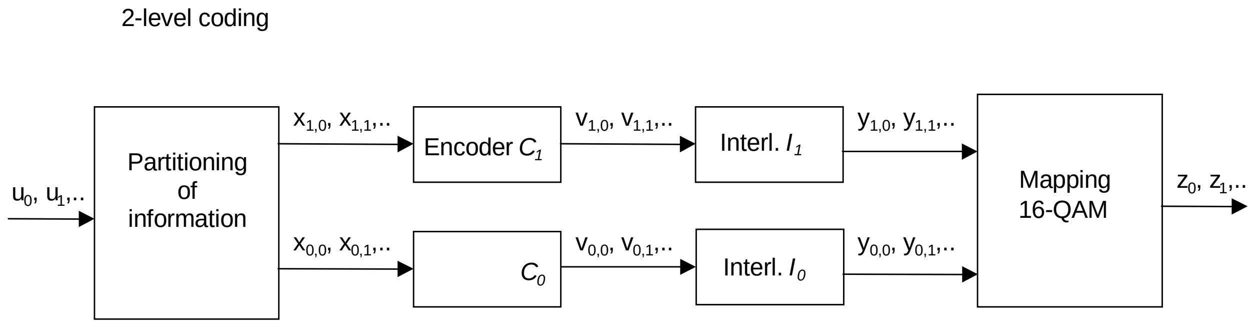


Figure 24: Multilevel coding with 2 levels (SM)

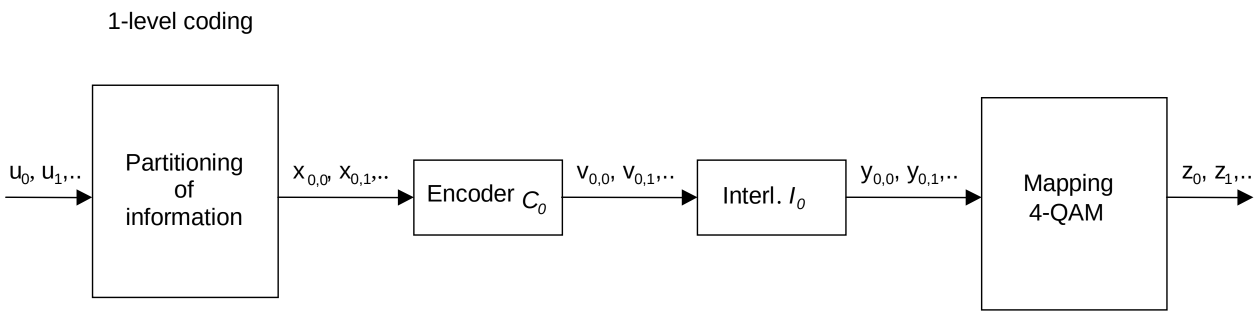


Figure 25: Multilevel coding with 1 level (SM)

7.3.1.1 Partitioning of bitstream in SM

The bitstream (u_i) shall be partitioned into several streams ($x_{p,i}$) according the number of levels. The bits of the higher protected part shall be fed to the encoders on p = 0,...P_{max}-1, then the bits of the lower protected part shall be fed to the encoders on p = 0,...P_{max}-1. This results in:

$$(x_{0,0}, x_{0,1}, \dots x_{0,M_{0,1}-1}, x_{1,0}, x_{1,1}, \dots x_{1,M_{1,1}-1}, x_{2,0}, x_{2,1}, \dots x_{2,M_{2,1}-1}, x_{0,M_{0,1}} x_{0,M_{0,1}+1}, \dots x_{0,M_{0,1}+M_{0,2}-1},$$
$$x_{1,M_{1,1}} x_{1,M_{1,1}+1}, \dots x_{1,M_{1,1}+M_{1,2}-1}, x_{2,M_{2,1}} x_{2,M_{2,1}+1}, \dots x_{2,M_{2,1}+M_{2,2}-1}) = (u_0, u_1, \dots u_{L_{VSPP}+L_1+L_2-1})$$

for the 3-level coding,

$$(x_{0,0}, x_{0,1}, \dots x_{0,M_{0,1}-1}, x_{1,0}, x_{1,1}, \dots x_{1,M_{1,1}-1},$$
$$x_{0,M_{0,1}} x_{0,M_{0,1}+1}, \dots x_{0,M_{0,1}+M_{0,2}-1}, x_{1,M_{1,1}} x_{1,M_{1,1}+1}, \dots x_{1,M_{1,1}+M_{1,2}-1}) = (u_0, u_1, \dots u_{L_1+L_2-1})$$

for the 2-level coding,

$$(x_{0,0}, x_{0,1}, \dots x_{0,M_{0,1}-1}, x_{0,M_{0,1}} x_{0,M_{0,1}+1}, \dots x_{0,M_{0,1}+M_{0,2}-1}) = (u_0, u_1, \dots u_{L_1+L_2-1})$$

for the 1-level coding.

When using only one protection level (EEP) the elements with negative indexes shall not be taken into account.

The number of bits on each level *p* is calculated for the higher protected part and lower protected part by:

$$M_{p,1} = 2N_1R_p \text{ where } p \in \{0,1,2\}$$

$$M_{p,2} = RX_p \left\lfloor \frac{2N_2 - 12}{RY_p} \right\rfloor \text{ where } p \in \{0,1,2\}$$

NOTE: The actual number of bits in the higher protected part (L_1) can be greater than the number signalled in the SDC. This means that some bits belonging to part B of the multiplex frame are in fact protected at the higher level.

The total number of bits on each level *p* is:

$$M_p = M_{p,1} + M_{p,2}$$

From these formulas it can be derived that the input bitstreams ($x_{p,i}$) to the encoders C_p have different lengths according to their code rate so that all the encoder output bitstreams ($v_{p,i}$) have the same length.

The overall code rate for each protection part for the SM is approximately:

$$R_{all} = \frac{\sum_{p=0}^{P_{\max}-1} R_p}{P_{\max}},$$

when using P_{max} levels.

7.3.1.2 Partitioning of bitstream in HMsym

The bitstream of the SPP (u_i) shall be partitioned into two streams ($x_{p,i}$). The bits of the higher protected part shall be fed to the encoders on p = 1 then p = 2, then the bits of the lower protected part shall be fed to the encoders on p = 1 then p = 2. This results in:

$$(x_{1,0}, x_{1,1}, \ldots x_{1,M_{1,1}-1}, x_{2,0}, x_{2,1}, \ldots x_{2,M_{2,1}-1}, x_{1,M_{1,1}} x_{1,M_{1,1}+1}, \ldots x_{1,M_{1,1}+M_{1,2}-1},$$
$$x_{2,M_{2,1}} x_{2,M_{2,1}+1}, \ldots x_{2,M_{2,1}+M_{2,2}-1}) = \left(u_0, u_1, \ldots u_{L_1+L_2-1}\right)$$

When using only one protection level (EEP) the elements with negative indexes shall not be taken into account.

The bitstream of the VSPP (u'_i) shall be sent to the encoder on level 0:

$$(x_{0,0}, x_{0,1}, \ldots x_{0,M_{0,2}-1}) = \left(u'_0, u'_1, \ldots u'_{L_{VSPP}-1}\right)$$

The number of bits on each level *p* is calculated for the higher protected part and lower protected part by:

$$M_{p,1} = 2N_1 R_p \text{ where } p \in \{1,2\}$$

$$M_{p,2} = RX_p \left\lfloor \frac{2N_2 - 12}{RY_p} \right\rfloor \text{ where } p \in \{1,2\}$$

and

$$M_{0,1} = 0$$

$$M_{0,2} = RX_0 \left\lfloor \frac{2(N_1 + N_2) - 12}{RY_0} \right\rfloor = L_{VSPP}$$

The total number of bits on each level *p* is:

$$M_p = M_{p,1} + M_{p,2}$$

From these formulas it can be derived that the input bitstreams ($x_{p,i}$) to the encoders C_p have different lengths according to their code rate so that all the encoder output bitstreams ($v_{p,i}$) have the same length.

The overall code rate for each protection part for the HMsym is approximately:

$$R_{VSPP} = R_0$$

$$R_{SPP,all} = (R_1 + R_2)/2$$

7.3.1.3 Partitioning of bitstream in HMmix

The bitstream of the SPP (u_i) shall be partitioned into five streams $(x_{p,i}^{\mathrm{Re}}, x_{p,i}^{\mathrm{Im}})$. The bits of the higher protected part shall be fed to the encoders on p = 0,...2, then the bits of the lower protected part shall be fed to the encoders on p = 0,...2. This results in:

$$(x_{0,0}^{\mathrm{Im}}, x_{0,1}^{\mathrm{Im}},..., x_{0,M_{0,1}^{\mathrm{Im}}-1}^{\mathrm{Im}}, x_{1,0}^{\mathrm{Re}}, x_{1,1}^{\mathrm{Re}},\ldots, x_{1,M_{1,1}^{\mathrm{Re}}-1}^{\mathrm{Re}}, x_{1,0}^{\mathrm{Im}}, x_{1,1}^{\mathrm{Im}},..., x_{1,M_{1,1}^{\mathrm{Im}}-1}^{\mathrm{Im}}, x_{2,0}^{\mathrm{Re}}, x_{2,1}^{\mathrm{Re}},\ldots, x_{2,M_{2,1}^{\mathrm{Re}}-1}^{\mathrm{Re}}, x_{2,0}^{\mathrm{Im}}, x_{2,1}^{\mathrm{Im}},..., x_{2,M_{2,1}^{\mathrm{Im}}-1}^{\mathrm{Im}},$$

$$x_{0,M_{0,1}^{\mathrm{Im}}}^{\mathrm{Im}}, x_{0,M_{0,1}^{\mathrm{Im}}+1}^{\mathrm{Im}},..., x_{0,M_{0,1}^{\mathrm{Im}}+M_{0,2}^{\mathrm{Im}}-1}^{\mathrm{Im}}, x_{1,M_{1,1}^{\mathrm{Re}}}^{\mathrm{Re}}, x_{1,M_{1,1}^{\mathrm{Re}}+1}^{\mathrm{Re}},\ldots, x_{1,M_{1,1}^{\mathrm{Re}}+M_{1,2}^{\mathrm{Re}}-1}^{\mathrm{Re}}, x_{1,M_{1,1}^{\mathrm{Im}}}^{\mathrm{Im}}, x_{1,M_{1,1}^{\mathrm{Im}}+1}^{\mathrm{Im}},..., x_{1,M_{1,1}^{\mathrm{Im}}+M_{1,1}^{\mathrm{Im}}-1}^{\mathrm{Im}},$$

$$x_{2,M_{2,1}^{\mathrm{Re}}}^{\mathrm{Re}}, x_{2,M_{2,1}^{\mathrm{Re}}+1}^{\mathrm{Re}},\ldots, x_{2,M_{2,1}^{\mathrm{Re}}+M_{2,2}^{\mathrm{Re}}-1}^{\mathrm{Re}}, x_{2,M_{2,1}^{\mathrm{Im}}}^{\mathrm{Im}}, x_{2,M_{2,1}^{\mathrm{Im}}+1}^{\mathrm{Im}},..., x_{2,M_{2,1}^{\mathrm{Im}}+M_{2,1}^{\mathrm{Im}}-1}^{\mathrm{Im}}) = \left(u_0, u_1,\ldots u_{L_1+L_2-1}\right)$$

The bits of the VSPP (u'_i) shall be fed to the encoder for the real part on level p = 0:

$$(x_{0,0}^{\mathrm{Re}}, x_{0,1}^{\mathrm{Re}},,\ldots, x_{0,M_{0,2}^{\mathrm{Re}}-1}^{\mathrm{Re}}) = \left(u'_0, u'_1,\ldots u'_{L_{VSPP}-1}\right)$$

When using only one protection level (EEP) the elements with negative indexes shall not be taken into account.

The number of bits on each level *p* is calculated for the higher protected and lower protected parts for the real and imaginary component by:

$$M_{0,1}^{\mathrm{Re}} = 0\,,\ M_{0,1}^{\mathrm{Im}} = N_1 R_0^{\mathrm{Im}}$$

$$M_{0,2}^{\mathrm{Re}} = RX_0^{\mathrm{Re}} \left\lfloor \frac{N_1 + N_2 - 12}{RY_0^{\mathrm{Re}}} \right\rfloor = L_{VSSP}\,,\ M_{0,2}^{\mathrm{Im}} = RX_0^{\mathrm{Im}} \left\lfloor \frac{N_2 - 12}{RY_0^{\mathrm{Im}}} \right\rfloor$$

$$M_{p,1}^{\mathrm{Re}} = N_1 R_p^{\mathrm{Re}} \text{ and } M_{p,1}^{\mathrm{Im}} = N_1 R_p^{\mathrm{Im}} \text{ for } p \in \{1,2\}$$

$$M_{p,2}^{\mathrm{Re}} = RX_p^{\mathrm{Re}} \left\lfloor \frac{N_2 - 12}{RY_p^{\mathrm{Re}}} \right\rfloor \text{ and } M_{p,2}^{\mathrm{Im}} = RX_p^{\mathrm{Im}} \left\lfloor \frac{N_2 - 12}{RY_p^{\mathrm{Im}}} \right\rfloor \text{ for } p \in \{1,2\}$$

The total number of bits on each level *p* in the real and imaginary component results in:

$$M_p^{\mathrm{Re}} = M_{p,1}^{\mathrm{Re}} + M_{p,2}^{\mathrm{Re}} \text{ and } M_p^{\mathrm{Im}} = M_{p,1}^{\mathrm{Im}} + M_{p,2}^{\mathrm{Im}} \text{ for } p \in \{0,1,2\}$$

From these formulas it can be derived that the input bitstreams $(x_{p,i}^{\mathrm{Re}})$ and $(x_{p,i}^{\mathrm{Im}})$ to the encoders C_p^{Re} and C_p^{Im} respectively have different lengths according to their code rate so that all the encoder output bitstreams $p \in \{0,1,2\}$ have the same length.

The overall code rate for the HMmix schemes of each protection part is approximately:

$$R_{VSPP} = R_0^{\mathrm{Re}}$$

$$R_{SPP,all} = (R_0^{\mathrm{Im}} + R_1^{\mathrm{Re}} + R_1^{\mathrm{Im}} + R_2^{\mathrm{Re}} + R_2^{\mathrm{Im}})/5$$

7.3.2 Component code

The component code C_p is based on punctured convolutional coding with a mother code of rate 1/4 and constraint length 7. The mother convolutional encoder generates from the vector $\left(x_{p,i}\right)_{i=0}^{M_p-1} = \left(a_i\right)_{i=0}^{I-1}$ a codeword

$\left\{\left(b_{0,i}, b_{1,i}, b_{2,i}, b_{3,i}\right)\right\}_{i=0}^{I+5}$. This codeword is defined by:

$$\begin{aligned} b_{0,i} &= a_i \oplus a_{i-2} \oplus a_{i-3} \oplus a_{i-5} \oplus a_{i-6}; \\ b_{1,i} &= a_i \oplus a_{i-1} \oplus a_{i-2} \oplus a_{i-3} \oplus a_{i-6}; \\ b_{2,i} &= a_i \oplus a_{i-1} \oplus a_{i-4} \oplus a_{i-6}; \\ b_{3,i} &= a_i \oplus a_{i-2} \oplus a_{i-3} \oplus a_{i-5} \oplus a_{i-6}; \end{aligned}$$

for i = 0, 1, 2, ..., I + 5.

When i does not belong to the set {0, 1, 2,..., I-1}, a_i is equal to zero by definition.

The encoding can be achieved using the convolutional encoder presented in figure 26.

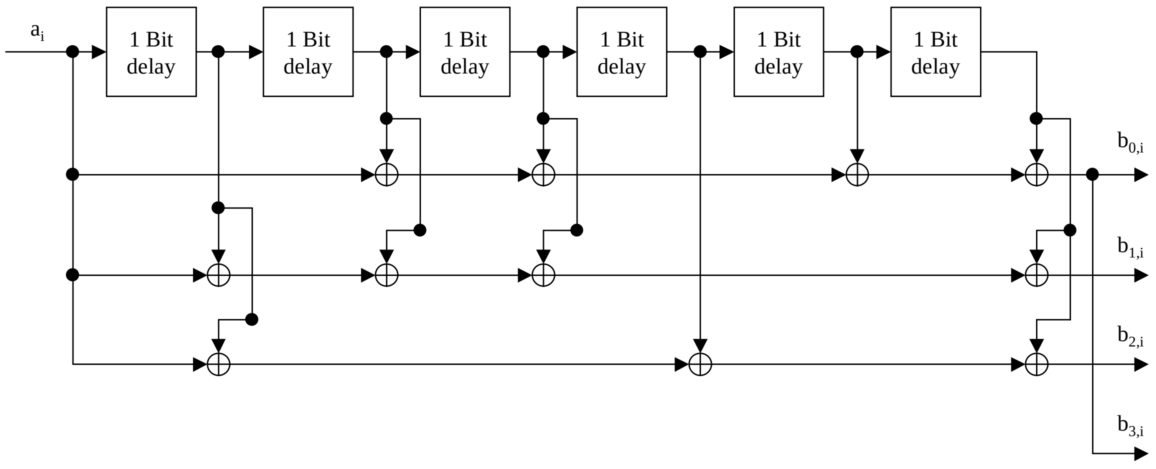


Figure 26: Convolutional encoder

The octal forms of the generator polynomials are 133, 171, 145 and 133, respectively.

The vector $\left(a_{-6}, a_{-5}, a_{-4}, a_{-3}, a_{-2}, a_{-1}\right)$ corresponds to the all-zero initial state of the shift register and the vector $\left(a_I, a_{I+1}, a_{I+2}, a_{I+3}, a_{I+4}, a_{I+5}\right)$ corresponds to the all-zero final state of the shift register.

In addition to the mother code the system shall allow punctured rates. Table 65 shows the puncturing patterns.

Table 65: Puncturing patterns

Code rates R_p	Numerator RX_p	Denominator RY_p	Puncturing pattern	Transmitted sequence
1/4	1	4	B_0: 1 B_1: 1 B_2: 1 B_3: 1	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{3,0}$ ...
3/10	3	10	B_0: 1 1 1 B_1: 1 1 1 B_2: 1 1 1 B_3: 1 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{3,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ ...
1/3	1	3	B_0: 1 B_1: 1 B_2: 1 B_3: 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ ...
4/11	4	11	B_0: 1 1 1 1 B_1: 1 1 1 1 B_2: 1 1 1 0 B_3: 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ $b_{0,3}$ $b_{1,3}$ ...
1/2	1	2	B_0: 1 B_1: 1 B_2: 0 B_3: 0	$b_{0,0}$ $b_{1,0}$ ...
4/7	4	7	B_0: 1 1 1 1 B_1: 1 0 1 0 B_2: 0 1 0 0 B_3: 0 0 0 0	$b_{0,0}$ $b_{1,0}b_{0,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{0,3}$ ...
3/5	3	5	B_0: 1 1 1 B_1: 1 0 1 B_2: 0 0 0 B_3: 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ $b_{0,2}$ $b_{1,2}$...
2/3	2	3	B_0: 1 1 B_1: 1 0 B_2: 0 0 B_3: 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ ...
8/11	8	11	B_0: 1 1 1 1 1 1 1 1 B_1: 1 0 0 1 0 0 1 0 B_2: 0 0 0 0 0 0 0 0 B_3: 0 0 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$ $b_{1,3}$ $b_{0,4}$ $b_{0,5}$ $b_{0,6}$ $b_{1,6}$ $b_{0,7}$ ...
3/4	3	4	B_0: 1 1 1 B_1: 1 0 0 B_2: 0 0 0 B_3: 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ $b_{0,2}$ ...
4/5	4	5	B_0: 1 1 1 1 B_1: 1 0 0 0 B_2: 0 0 0 0 B_3: 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$ $b_{0,4}$ ...

Code rates R_p	Numerator RX_p	Denominator RY_p	Puncturing pattern	Transmitted sequence
7/8	7	8	B_0: 1 1 1 1 1 1 1 B_1: 1 0 0 0 0 0 0 B_2: 0 0 0 0 0 0 0 B_3: 0 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$ $b_{0,4}$ $b_{0,5}$ $b_{0,6}$...
8/9	8	9	B_0: 1 1 1 1 1 1 1 1 B_1: 1 0 0 0 0 0 0 0 B_2: 0 0 0 0 0 0 0 0 B_3: 0 0 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$ $b_{0,4}$ $b_{0,5}$ $b_{0,6}$ $b_{0,7}$...

For the FAC, all bits are punctured according to table 65. For the MSC and the SDC, the last 24 bits (the tailbits) of the serial mother codeword shall be punctured as follows. The index r_p shall be used with table 66 to find the puncturing vector for the tailbits for each level. This index is calculated with the following formula:

SM and HMsym:

$$r_p = (2N_2 - 12) - RY_p \left\lfloor \frac{2N_2 - 12}{RY_p} \right\rfloor \text{ for } p \in \{0,1,2\}$$

HMmix:

$$r_0^{\text{Re}} = (N_1 + N_2 - 12) - RY_0^{\text{Re}} \left\lfloor \frac{N_1 + N_2 - 12}{RY_0^{\text{Re}}} \right\rfloor,$$

$$r_p^{\text{Re}} = (N_2 - 12) - RY_p^{\text{Re}} \left\lfloor \frac{N_2 - 12}{RY_p^{\text{Re}}} \right\rfloor \text{ for } p \in \{1,2\}$$

$$r_p^{\text{Im}} = (N_2 - 12) - RY_p^{\text{Im}} \left\lfloor \frac{N_2 - 12}{RY_p^{\text{Im}}} \right\rfloor \text{ for } p \in \{0,1,2\}$$

Table 66: Puncturing patterns of the tailbits

r_p	Puncturing pattern	Transmitted sequence
0	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 0 0 0 0 0 0 B_3: 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{0,1}$ $b_{1,1}$ $b_{0,2}$ $b_{1,2}$ $b_{0,3}$ $b_{1,3}$ $b_{0,4}$ $b_{1,4}$ $b_{0,5}$ $b_{1,5}$ ···
1	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 0 0 0 0 0 B_3: 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{0,1}$ $b_{1,1}$ $b_{0,2}$ $b_{1,2}$ $b_{0,3}$ $b_{1,3}$ $b_{0,4}$ $b_{1,4}$ $b_{0,5}$ $b_{1,5}$ ···
2	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 0 0 1 0 0 B_3: 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{0,1}$ $b_{1,1}$ $b_{0,2}$ $b_{1,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{0,4}$ $b_{1,4}$ $b_{0,5}$ $b_{1,5}$ ···
3	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 0 1 0 0 B_3: 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{0,4}$ $b_{1,4}$ $b_{0,5}$ $b_{1,5}$ ···
4	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 0 1 1 0 B_3: 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ ···
5	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 1 1 1 0 B_3: 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ ···
6	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 1 1 1 1 B_3: 0 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ $b_{2,5}$ ···
7	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 1 1 1 1 B_3: 1 0 0 0 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{3,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ $b_{2,5}$ ···
8	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 1 1 1 1 B_3: 1 0 0 1 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{3,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{3,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ $b_{2,5}$ ···
9	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 1 1 1 1 B_3: 1 1 0 1 0 0	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{3,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{3,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{3,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ $b_{2,5}$ ···
10	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 1 1 1 1 B_3: 1 1 0 1 0 1	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{3,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{3,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$ $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{3,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ $b_{2,5}$, $b_{3,5}$, ···
11	B_0: 1 1 1 1 1 1 B_1: 1 1 1 1 1 1 B_2: 1 1 1 1 1 1 B_3: 1 1 1 1 0 1	$b_{0,0}$ $b_{1,0}$ $b_{2,0}$ $b_{3,0}$ $b_{0,1}$ $b_{1,1}$ $b_{2,1}$ $b_{3,1}$ $b_{0,2}$ $b_{1,2}$ $b_{2,2}$, $b_{3,2}$, $b_{0,3}$ $b_{1,3}$ $b_{2,3}$ $b_{3,3}$ $b_{0,4}$ $b_{1,4}$ $b_{2,4}$ $b_{0,5}$ $b_{1,5}$ $b_{2,5}$, $b_{3,5}$, ···

The puncturing shall be performed as follows:

SM:

The higher protected part of the transmitted sequence is punctured according to table 65 resulting in $\left(v_{p,0} \ldots v_{p,2N_1-1}\right)$.

The lower protected part of the transmitted sequence is punctured according to table 65 resulting in $\left(v_{p,2N_1} \ldots v_{p,2(N_1+N_2)-13-r_p}\right)$.

The tailbits of the transmitted sequence are punctured according to table 66 resulting in $\left(v_{p,2(N_1+N_2)-12-r_p} \ldots v_{p,2(N_1+N_2)-1}\right)$.

NOTE 1: If there is only one protection level the higher protected part is absent.

HMsym:

The VSPP part of the transmitted sequence is punctured according to table 65 resulting in:

$\left(v_{0,0} \ldots v_{0,2(N_1+N_2)-13-r_0}\right)$.

The tailbits for the VSPP part of the transmitted sequence are punctured according to table 66 resulting in:

$\left(v_{0,2(N_1+N_2)-12-r_0} \ldots v_{0,2(N_1+N_2)-1}\right)$.

The higher protected part of the SPP part of the transmitted sequence is punctured according to table 65 resulting in:

$\left(v_{p,0} \ldots v_{p,2N_1-1}\right)$ for $p \in \{1,2\}$.

The lower protected part of the SPP part of the transmitted sequence is punctured according to table 65 resulting in:

$\left(v_{p,2N_1} \ldots v_{p,2(N_1+N_2)-13-r_p}\right)$ for $p \in \{1,2\}$.

The tailbits for the SPP part of the transmitted sequence is punctured according to table 66 resulting in:

$\left(v_{p,2(N_1+N_2)-12-r_p} \ldots v_{p,2(N_1+N_2)-1}\right)$ for $Y(p) = \left(y_{p,0}, y_{p,1}, y_{p,2}, \ldots y_{p,2(N_1+N_2)-1}\right)$.

NOTE 2: If there is only one protection level the higher protected part is absent.

HMmix:

The VSPP part of the transmitted sequence is punctured according to table 65 resulting in:

$\left(v^{\mathrm{Re}}_{0,0} \ldots v^{\mathrm{Re}}_{0,2(N_1+N_2)-13-r_0}\right)$.

The tailbits for the VSPP part of the transmitted sequence are punctured according to table 66 resulting in:

$\left(v^{\mathrm{Re}}_{0,2(N_1+N_2)-12-r_0} \ldots v^{\mathrm{Re}}_{0,2(N_1+N_2)-1}\right)$.

The real component of the higher protected part of the SPP part of the transmitted sequence is punctured according to table 65 resulting in:

$\left(v^{\mathrm{Re}}_{p,0} \ldots v^{\mathrm{Re}}_{p,2N_1-1}\right)$ for $Xin = 2N_2$.

The real component of the lower protected part of the SPP part of the transmitted sequence is punctured according to table 65 resulting in:

$\left(v^{\mathrm{Re}}_{p,2N_1} \ldots v^{\mathrm{Re}}_{p,2(N_1+N_2)-13-r_p}\right)$ for $p \in \{1,2\}$.

The tailbits for the SPP part of the transmitted sequence are punctured according to table 66 resulting in:

$$\left(v^{\mathrm{Re}}_{p,2(N_1+N_2)-12-r_p} \ldots v^{\mathrm{Re}}_{p,2(N_1+N_2)-1}\right).$$

NOTE 3: If there is only one protection level the higher protected part is absent.

The imaginary component of the higher protected part of the SPP part of the transmitted sequence is punctured according to table 44 resulting in:

$$\left(v^{\mathrm{Im}}_{p,0} \ldots v^{\mathrm{Im}}_{p,2N_1-1}\right) \text{ for } p \in \{0,1,2\}.$$

The imaginary component of the lower protected part of the SPP part of the transmitted sequence is punctured according to table 65 resulting in:

$$\left(v^{\mathrm{Im}}_{p,2N_1} \ldots v^{\mathrm{Im}}_{p,2(N_1+N_2)-13-r_p}\right) \text{ for } p \in \{0,1,2\}.$$

The tailbits for the SPP part of the transmitted sequence are punctured according to table 66 resulting in:

$$\left(v^{\mathrm{Im}}_{p,2(N_1+N_2)-12-r_p} \ldots v^{\mathrm{Im}}_{p,2(N_1+N_2)-1}\right) \text{ for } p \in \{0,1,2\}.$$

NOTE 4: If there is only one protection level the higher protected part is absent.

7.3.3 Bit interleaving

Bit-wise interleaving shall be applied for some of the levels of the coding scheme according to figures 21 to 25. The same basic algorithm which results in a pseudo random bit ordering shall be used independently for the FAC, SDC and MSC.

The permutation $\Pi_p(i)$ is obtained from the following relations:

for 64-QAM: $t_1 = 13$, $t_2 = 21$

for 16-QAM: $t_0 = 13$, $t_1 = 21$

for 4-QAM: $t_0 = 21$

$p \in \{0,1,2\}$

$s = 2^{\lceil \log 2(x_{in}) \rceil}$

$q = s/4 - 1$

the number of input bits x_{in} is defined below

and $\lceil\ \rceil$ means round towards plus infinity

```
$\Pi_p(0) = 0$;
for $i = 1, 2, \ldots, x_{in} - 1$:
    $\Pi_p(i) = (t_p\ \Pi_p(i-1) + q)(\mathrm{mod}\, s)$;
        while $\Pi_p(i) \ge x_{in}$:
            $\Pi_p(i) = (t_p\ \Pi_p(i) + q)(\mathrm{mod}\, s)$.
```

7.3.3.1 FAC

The block size shall be in every case the same for the interleaver I_p with p = 0 only. The number of elements per bit interleaver x_{in} equals $2N_{FAC}$. The input vector is defined by:

$$V(p) = \left(v_{p,0}, v_{p,1}, v_{p,2}, \ldots v_{p,2N_{FAC}-1}\right)$$

The interleaved output vector is the subset of the permutations Π_p (i) defined by:

$$Y(p) = \left(y_{p,0}, y_{p,1}, y_{p,2}, \ldots y_{p,2N_{FAC}-1}\right)$$

The output elements are selected from the input elements according to:

$$y_{p,i} = v_{p,\Pi_p(i)} \cdot$$

7.3.3.2 SDC

The block size shall be the same for each interleaver I_p. The number of elements per bit interleaver x_{in} equals $2N_{SDC}$. For each bit interleaver, the input vector is defined by:

$$V(p) = \left(v_{p,0}, v_{p,1}, v_{p,2}, \ldots v_{p,2N_{SDC}-1}\right)$$

The interleaved output vector is the subset of the permutations Π_p (i) defined by:

$$Y(p) = \left(y_{p,0}, y_{p,1}, y_{p,2}, \ldots y_{p,2N_{SDC}-1}\right)$$

The output elements are selected from the input elements according to:

$$y_{p,i} = v_{p,\Pi_p(i)} \cdot$$

7.3.3.3 MSC

SM and HMsym:

The block size shall be the same for each interleaver I_p, but shall be dependent on the robustness mode, spectrum occupancy and the constellation. The number of elements per bit interleaver equals $2(N_1 + N_2)$. For each bit interleaver, the input vector is defined by:

$$V(p) = \left(v_{p,0}, v_{p,1}, v_{p,2}, \ldots v_{p,2(N_1+N_2)-1}\right) = \left(v_{1,p,0}, v_{1,p,1}, \ldots v_{1,p,2N_1-1}, v_{2,p,0}, v_{2,p,1}, \ldots v_{2,p,2N_2-1}\right)$$

The interleaved output vector is the subset of the two permutations Π_p (i) defined by:

$$Y(p) = \left(y_{p,0}, y_{p,1}, y_{p,2}, \ldots y_{p,2(N_1+N_2)-1}\right) = \left(y_{1,p,0}, y_{1,p,1}, \ldots y_{1,p,2N_1-1}, y_{2,p,0}, y_{2,p,1}, \ldots y_{2,p,2N_2-1}\right)$$

The two parts with different protection levels shall not overlap in the interleaving process. Therefore the interleaved lower protected part shall be appended to the interleaved higher protected part where the output elements are selected from the input elements according to:

$$y_{1,p,i} = v_{1,p,\Pi_p(i)} \text{ and } y_{2,p,i} = v_{2,p,\Pi_p(i)}$$

for each part respectively.

The number of input bits used for the permutation for the higher protected part is $x_{in} = 2N_1$, and for the lower protected part is $x_{in} = 2N_2$.

HMmix:

The block size shall be the same for each interleaver I_p^{Re} and I_p^{Im} but shall be dependent on the robustness mode, spectrum occupancy and the constellation. The number of elements per bit interleaver equals $(N_1 + N_2)$. For each bit interleaver, the input vectors for the real and imaginary components are defined by:

$$V^{\mathrm{Re}}(p) = \left(v_{p,0}^{\mathrm{Re}}, v_{p,1}^{\mathrm{Re}}, v_{p,2}^{\mathrm{Re}}, \ldots v_{p,N_1+N_2-1}^{\mathrm{Re}}\right) = \left(v_{1,p,0}^{\mathrm{Re}}, v_{1,p,1}^{\mathrm{Re}}, \ldots v_{1,p,N_1-1}^{\mathrm{Re}}, v_{2,p,0}^{\mathrm{Re}}, v_{2,p,1}^{\mathrm{Re}}, \ldots v_{2,p,N_2-1}^{\mathrm{Re}}\right) \text{ or}$$

$$V^{\mathrm{Im}}(p) = \left(v_{p,0}^{\mathrm{Im}}, v_{p,1}^{\mathrm{Im}}, v_{p,2}^{\mathrm{Im}}, \ldots v_{p,N_1+N_2-1}^{\mathrm{Im}}\right) = \left(v_{1,p,0}^{\mathrm{Im}}, v_{1,p,1}^{\mathrm{Im}}, \ldots v_{1,p,N_1-1}^{\mathrm{Im}}, v_{2,p,0}^{\mathrm{Im}}, v_{2,p,1}^{\mathrm{Im}}, \ldots v_{2,p,N_2-1}^{\mathrm{Im}}\right) \text{ respectively.}$$

The interleaved output vectors for the real and imaginary components are the subsets of the two permutations $\Pi_p(i)$ defined by:

$$Y^{\mathrm{Re}}(p) = \left(y_{p,0}^{\mathrm{Re}}, y_{p,1}^{\mathrm{Re}}, y_{p,2}^{\mathrm{Re}}, \ldots y_{p,N_1+N_2-1}^{\mathrm{Re}}\right) = \left(y_{1,p,0}^{\mathrm{Re}}, y_{1,p,1}^{\mathrm{Re}}, \ldots y_{1,p,N_1-1}^{\mathrm{Re}}, y_{2,p,0}^{\mathrm{Re}}, y_{2,p,1}^{\mathrm{Re}}, \ldots y_{2,p,N_2-1}^{\mathrm{Re}}\right) \text{ or}$$

$$Y^{\mathrm{Im}}(p) = \left(y_{p,0}^{\mathrm{Im}}, y_{p,1}^{\mathrm{Im}}, y_{p,2}^{\mathrm{Im}}, \ldots y_{p,N_1+N_2-1}^{\mathrm{Im}}\right) = \left(y_{1,p,0}^{\mathrm{Im}}, y_{1,p,1}^{\mathrm{Im}}, \ldots y_{1,p,N_1-1}^{\mathrm{Im}}, y_{2,p,0}^{\mathrm{Im}}, y_{2,p,1}^{\mathrm{Im}}, \ldots y_{2,p,N_2-1}^{\mathrm{Im}}\right) \text{ respectively.}$$

The two parts with different protection levels shall not overlap in the interleaving process Therefore the interleaved lower protected part shall be appended to the interleaved higher protected part where the output elements are selected from the input elements according to:

$$y_{1,p,i}^{\mathrm{Re}} = v_{1,p,\Pi(i)}^{\mathrm{Re}}, \; y_{2,p,i}^{\mathrm{Re}} = v_{2,p,\Pi(i)}^{\mathrm{Re}}, \; y_{1,p,i}^{\mathrm{Im}} = v_{1,p,\Pi(i)}^{\mathrm{Im}} \text{ and } y_{2,p,i}^{\mathrm{Im}} = v_{2,p,\Pi(i)}^{\mathrm{Im}}$$

for each part respectively.

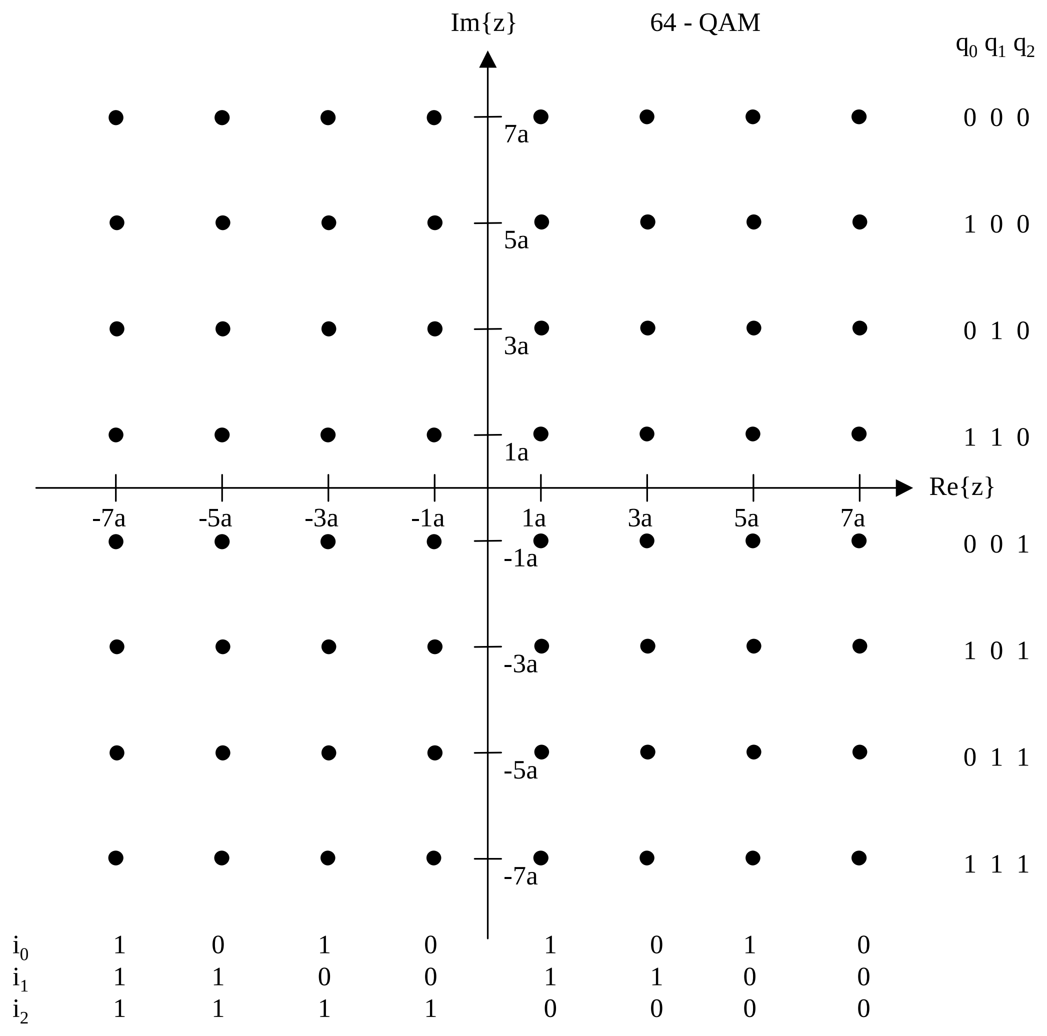
The number of input bits used for the permutation for the higher protected parts is $x_{in} = N_1$, and for the lower protected parts is $x_{in} = N_2$.

7.4 Signal constellations and mapping

The mapping strategy for each OFDM cell is dependent of the assignment to the channel (FAC, SDC, MSC) and the robustness mode. All data cells are either 4-QAM, 16-QAM or 64-QAM.

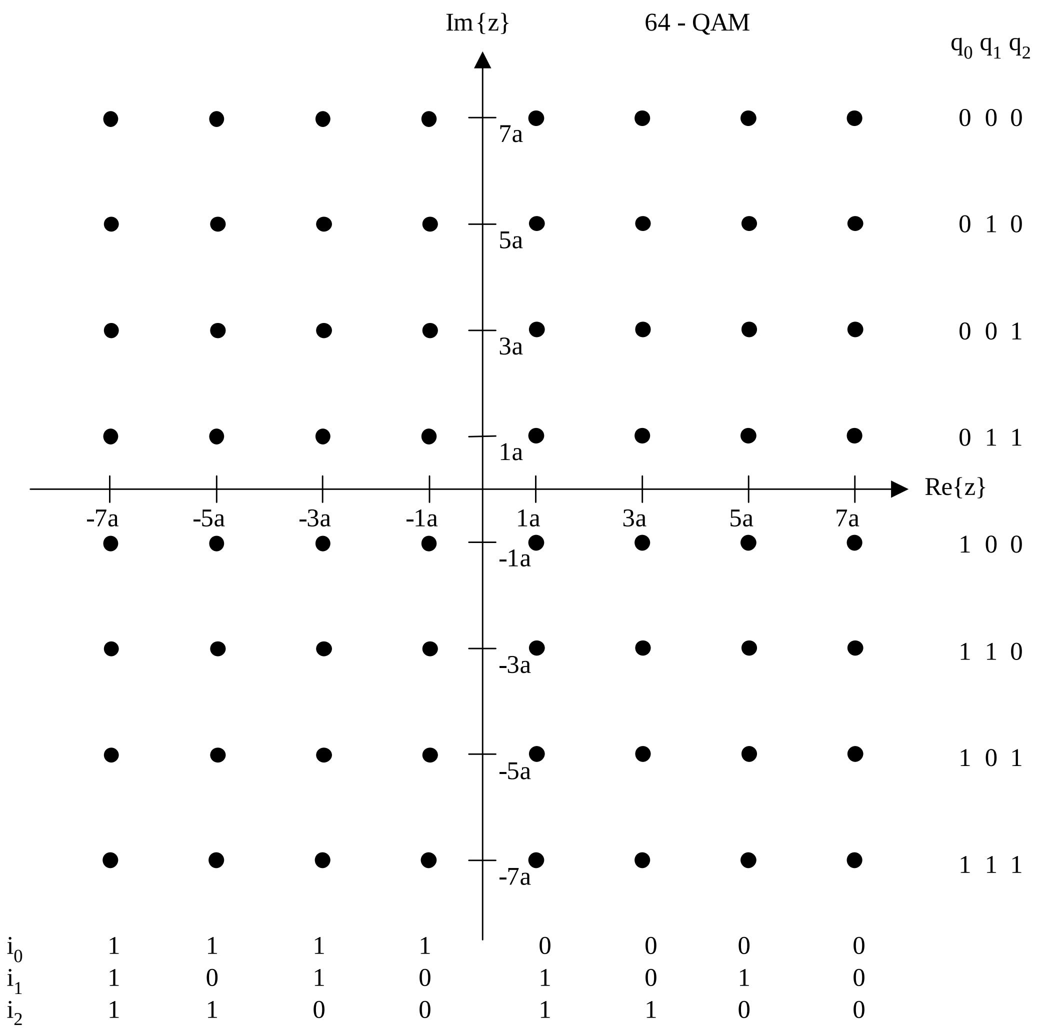
The default method for mapping shall be performed according to figures 27 to 31.

The y'_i denote the bits representing a complex modulation symbol z.



Bit ordering: $\{i_0\ i_1\ i_2\ q_0\ q_1\ q_2\} = \{y'_0\ y'_1\ y'_2\ y'_3\ y'_4\ y'_5\}$

Figure 27: SM 64-QAM mapping with corresponding bit pattern



Bit ordering: $\{i_0\, i_1\, i_2\, q_0\, q_1\, q_2\} = \{y'_0\, y'_1\, y'_2\, y'_3\, y'_4\, y'_5\}$

Figure 28: HMsym 64-QAM mapping with corresponding bit pattern

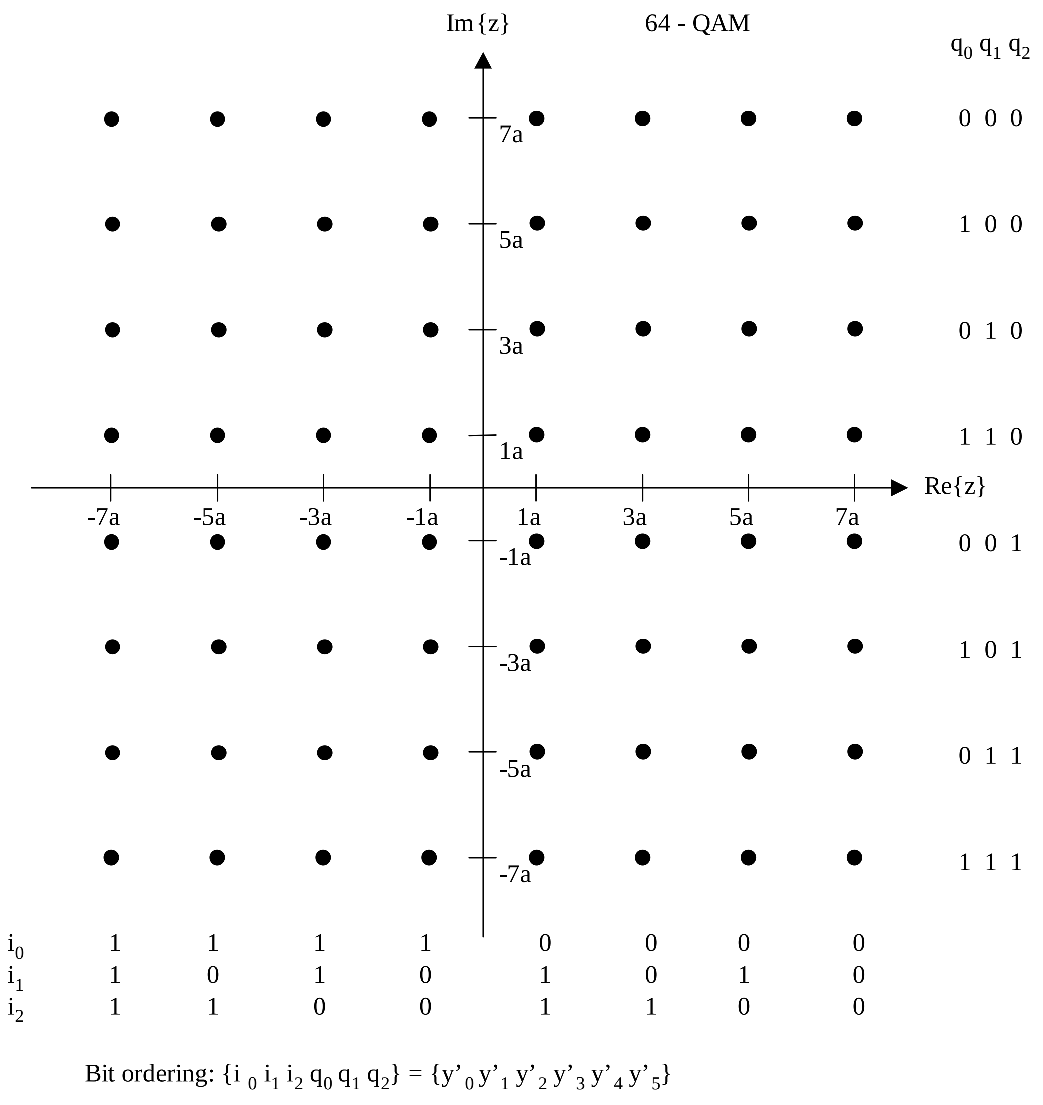
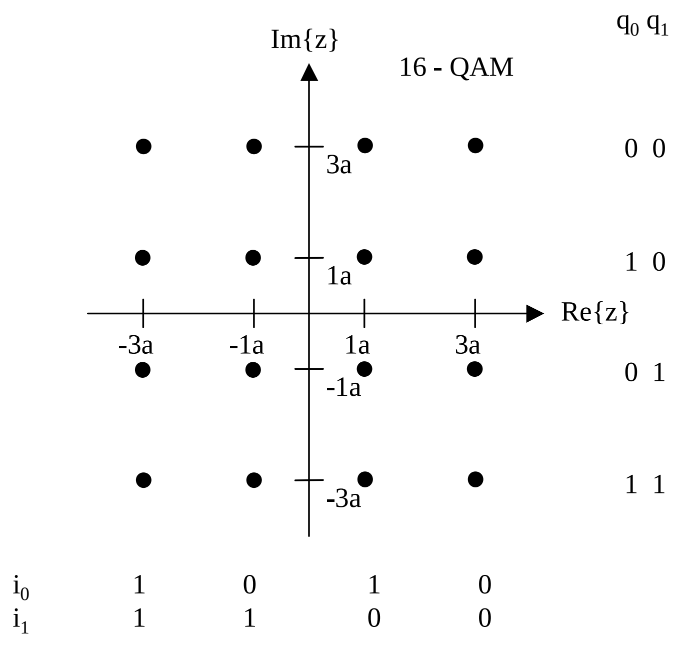
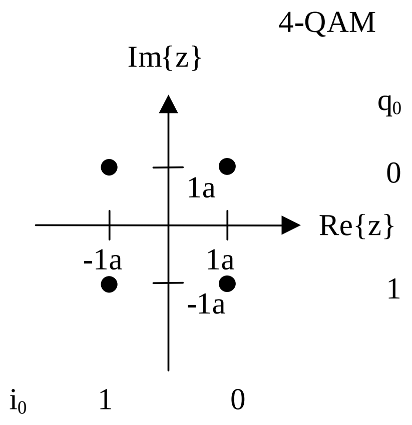


Figure 29: HMmix 64-QAM mapping with corresponding bit pattern



Bit ordering: $\{i_0\ i_1\ q_0\ q_1\} = \{y'_0\ y'_1\ y'_2\ y'_3\}$

Figure 30: SM 16-QAM mapping with corresponding bit pattern



Bit ordering: $\{i_0\ q_0\} = \{y'_0\ y'_1\}$

Figure 31: SM 4-QAM mapping with corresponding bit pattern

NOTE: Left hand bit is the first in time

For 64-QAM, the normalization factor is $a = \frac{1}{\sqrt{42}}$.

For 16-QAM, the normalization factor is $a = \frac{1}{\sqrt{10}}$.

For 4-QAM, the normalization factor is $a = \frac{1}{\sqrt{2}}$.

The data stream at the output of the interleaver consists of a number of bit words. These are mapped onto one signal point in the signal diagram according a complex number z. For SM and HMsym the 64-QAM diagram shall be used according figure 27 and 28 respectively. The bits shall be mapped accordingly:

$$\left(y'_0\, y'_1\, y'_2\, y'_3\, y'_4\, y'_5\right) = \left(y_{0,0}\, y_{1,0}\, y_{2,0}\, y_{0,1}\, y_{1,1}\, y_{2,1}\right).$$

For HMmix the 64-QAM diagram shall be used according to figure 29. The bits shall be mapped accordingly:

$$\left(y'_0\, y'_1\, y'_2\, y'_3\, y'_4\, y'_5\right) = \left(y^{\mathrm{Re}}_{0,0}\, y^{\mathrm{Re}}_{1,0}\, y^{\mathrm{Re}}_{2,0}\, y^{\mathrm{Im}}_{0,0}\, y^{\mathrm{Im}}_{1,0}\, y^{\mathrm{Im}}_{2,0}\right).$$

The 16-QAM diagram shall be used according figure 30. The bits shall be mapped accordingly:

$$\left(y'_0\, y'_1\, y'_2\, y'_3\right) = \left(y_{0,0}\, y_{1,0}\, y_{0,1}\, y_{1,1}\right)$$

The 4-QAM diagram shall be used according figure 31. The bits shall be mapped accordingly:

$$\left(y'_0\, y'_1\right) = \left(y_{0,0} y_{0,1}\right)$$

7.5 Application of coding to the channels

7.5.1 Coding the MSC

The MSC may use either 64-QAM or 16-QAM mapping. 64-QAM provides high spectral efficiency whereas 16-QAM provides a more robust error performance. In each case, a range of code rates is available to provide the most appropriate level of error correction for a given transmission. The available combinations of constellation and code rate provide a large degree of flexibility over a wide range of transmission channels. Unequal error protection can be used to provide two levels of protection for the MSC. For the case of 64-QAM, hierarchical modulation may be used to provide a third level of error robustness for a part of the MSC.

7.5.1.1 SM

Two protection levels within one multiplex frame are possible resulting in the use of two overall code rates. The number of input bits L_{MUX} per multiplex frame are calculated with the formulas of clause 7.2.

The MSC shall be encoded according to clause 7.3. The overall code rates and code rates for each level are defined in tables 67 and 68. The protection level is signalled in the multiplex description data entity of the SDC (see clause 6.4.3.1).

Two overall code rates are defined for 16-QAM as follows:

Table 67: Code rate combinations for the MSC with 16-QAM

Protection level	R_{all}	R_0	R_1	RY_{lcm}
0	0,5	1/3	2/3	3
1	0,62	1/2	3/4	4

Four overall code rates are defined for 64-QAM as follows:

Table 68: Code rate combinations for the MSC with 64-QAM

Protection level	R_{all}	R_0	R_1	R_2	RY_{lcm}
0	0,5	1/4	1/2	3/4	4
1	0,6	1/3	2/3	4/5	15
2	0,71	1/2	3/4	7/8	8
3	0,78	2/3	4/5	8/9	45

NOTE: These code rates are also used for the imaginary part of HMmix.

One or two overall code rates shall be applied to one multiplex frame. When using two overall code rates, both shall belong to the same constellation.

Annex J provides the number of input bits per multiplex frame for EEP.

7.5.1.2 HMsym

Two protection levels are possible resulting in the use of two overall code rates. The number of input bits L_{MUX} per multiplex frame is calculated with the formulas of clause 7.2.

The MSC shall be encoded according to clause 7.3. The overall code rates and code rates for each level for the SPP are defined in table 69 and for the VSPP in table 49. The protection level is signalled in the multiplex description data entity of the SDC (see clause 6.4.3.1).

Four overall code rates are defined for the SPP as follows.

Table 69: Code rate combinations for the SPP of MSC with HMsym 64-QAM

Protection level	R_{all}	R_1	R_2	RY_{lcm}
0	0,45	3/10	3/5	10
1	0,55	4/11	8/11	11
2	0,72	4/7	7/8	56
3	0,78	2/3	8/9	9
NOTE: These code rates are also used for the real part of HMmix.				

Four overall code rates are defined independently for the VSPP as follows.

Table 70: Code rate combinations for the VSPP of MSC with HMsym 64-QAM

Protection level	R_0
0	1/2
1	4/7
2	3/5
3	2/3
NOTE: These code rates are also used for the real part of HMmix.	

Annex J provides the number of input bits per multiplex frame for EEP.

7.5.1.3 HMmix

Two protection levels are possible resulting in the use of two overall code rates. The number of input bits L_{MUX} per multiplex frame is calculated with the formulas of clause 7.2.

The MSC shall be encoded according to clause 7.3. The protection level is signalled in the multiplex description data entity of the SDC (see clause 6.4.3.1).

Four overall code rates are defined for the SPP as shown in table 71 and the four possible code rates for the VSPP are defined independently as shown in table 70.

Table 71: Code rate combinations for the SPP of MSC with HMmix 64-QAM

Protection level	R_{all}	R_0^{Im}	R_1^{Re}	R_1^{Im}	R_2^{Re}	R_2^{Im}	RY_{lcm}
0	0,48	1/4	3/10	1/2	3/5	3/4	20
1	0,58	1/3	4/11	2/3	8/11	4/5	165
2	0,71	1/2	4/7	3/4	7/8	7/8	56
3	0,78	2/3	2/3	4/5	8/9	8/9	45

Annex J provides the number of input bits per multiplex frame for EEP.

7.5.2 Coding the SDC

The SDC may use either 16-QAM or 4-QAM mapping. 16-QAM provides greater capacity whereas 4-QAM provides a more robust error performance. In each case, a fixed code rate is applied.

The constellation should be chosen with respect to the MSC parameters to provide more robustness for the SDC than for the MSC. When using hierarchical modulation, the SDC shall be coded using 4-QAM.

The number of input bits L_{SDC} per SDC block is calculated as given in clause 7.2.

For 16-QAM the combination given in table 72 shall be used.

Table 72: Code rate combinations for the SDC with 16-QAM

R_{all}	R_0	R_1
0,5	1/3	2/3

For 4-QAM the code rate given in table 73 shall be used.

Table 73: Code rate for the SDC with 4-QAM

R_{all}	R_0
0,5	1/2

Annex J provides the number of input bits per SDC block.

Error detection with a CRC is described in clause 6.

7.5.3 Coding the FAC

The FAC shall use 4-QAM mapping. A fixed code rate shall be applied.

The number of input bits L_{FAC} per FAC block is calculated as given in clause 7.2.

The code rate given in table 74 shall be used.

Table 74: Code rate for the FAC

R_{all}	R_0
0,6	3/5

Error detection with a CRC is described in clause 6.

7.6 MSC cell interleaving

A cell-wise interleaving shall be applied to the QAM symbols (cells) of the MSC after multilevel encoding with the possibility to choose low or high interleaving depth (denoted here as short or long interleaving) according to the predicted propagation conditions. The basic interleaver parameters are adapted to the size of a multiplex frame which corresponds to N_{MUX} cells.

For propagation channels with moderate time-selective behaviour (typical ground wave propagation in LF and MF) the short interleaving provides sufficient time- and frequency diversity for proper operation of the decoding process in the receiver (spreading of error bursts). The same block interleaving scheme as used for bit interleaving in the multilevel encoder (see clause 7.3.3) is always applied to the N_{MUX} cells of a multiplex frame.

The input vector of the block interleaver corresponding to the N_{MUX} QAM cells $z_{n,\,i}$ of multiplex frame n is given by

$$\mathbf{Z}_n = \left(z_{n,0}, z_{n,1}, z_{n,2}, \ldots, z_{n,\mathrm{N}_{\mathrm{MUX}}-1} \right)$$

The output vector with the same number of cells or elements, respectively, is given by:

$$\hat{\mathbf{Z}}_n = \left(\hat{z}_{n,0}, \hat{z}_{n,1}, \hat{z}_{n,2}, \ldots, \hat{z}_{n,\mathrm{N}_{\mathrm{MUX}}-1} \right)$$

where the output elements are selected from the input elements according to:

$$\hat{z}_{n,\,i} = z_{n,\,\Pi(i)}\,.$$

The permutation $\Pi(i)$ is obtained from the following relations:

$s = 2^{\lceil \log 2(\mathrm{N}_{\mathrm{MUX}}) \rceil}$, $\lceil\ \rceil$ means round towards plus infinity;

$q = s/4 - 1$;

$t_0 = 5$;

$\Pi(0) = 0$;

```
for $i = 1, 2, \ldots, \mathrm{N}_{\mathrm{MUX}} - 1$:
    $\Pi(i) = \left(t_0\ \Pi(i-1) + q\right)\left(\mathrm{mod}\ s\right)$;
        while $\Pi(i) \geq \mathrm{N}_{\mathrm{MUX}}$:
            $\Pi(i) = \left(t_0\ \Pi(i) + q\right)\left(\mathrm{mod}\ s\right)$.
```

For severe time- and frequency-selective channels as being typical for signal transmissions in the HF short wave frequency bands the interleaving depth can be increased by an additional simple convolutional interleaving scheme. For this the interleaving depth D is defined in integer multiples of multiplex frames. As a good trade-off between performance and processing delay a value of D = 5 has been chosen.

The output vector for long interleaving with $\mathrm{N}_{\mathrm{MUX}}$ cells carrying complex QAM symbols is computed in almost the same way as for short interleaving. The only exception is that the permutation is based not only on the current but also on the last D-1 multiplex frames. The permutation $\Pi(i)$ as defined before is used again to determine the relation between the indices within the output vector $\hat{\mathbf{Z}}_n$ and the D input vectors $\mathbf{Z}_n, \mathbf{Z}_{n-1}, \ldots, \mathbf{Z}_{n-\mathrm{D}+1}$.

The output elements are selected from the input elements according to:

$$\hat{z}_{n,\,i} = z_{n-\Gamma(i),\,\Pi(i)}$$

For given value i the selection of the input vector number $n - \Gamma(i)$ for the correspondent element $\Pi(i)$ is done with the following formula:

$\Gamma(i) = i\ (\mathrm{mod}\,\mathrm{D})$ for $i = 0, 1, 2, \ldots, \mathrm{N}_{\mathrm{MUX}} - 1$.

Taking into consideration the transmission of the full content of a multiplex frame the overall delay of the pure interleaving/deinterleaving process is given by approximately 2 × 400 ms, i.e. 800 ms, for the short interleaving. In the case of the long interleaving it corresponds to about 2,4 s. In addition to that the delay is increased during transmission due to the fact that the SDC block is inserted at the beginning of a transmission super frame. With a depth of D = 5 multiplex frames for long interleaving the maximum additional increase in delay is given by the time duration of two SDC blocks.

7.7 Mapping of MSC cells on the transmission super frame structure

The content of three consecutive interleaved multiplex frames (with N_{MUX} QAM cells each) is mapped onto a transmission super frame, i.e. the corresponding number N_{SFU} of useful MSC cells is fixed as an integer multiple of 3. Due to the fact that the number of FAC and synchronization cells is varying from OFDM symbol to OFDM symbol a small loss N_L of 1 or 2 cells can occur compared with the number of available cells in a transmission super frame which is given by:

$$N_{SFA} = N_{SFU} + N_L = 3 \times N_{MUX} + N_L$$

Tables 75 to 78 give the values for the different robustness modes and bandwidths.

Table 75: Number of QAM cells for MSC for Mode A

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
Number of available MSC cells per super frame N_{SFA}	3 778	4 268	7 897	8 877	16 394	18 354
Number of useful MSC cells per super frame N_{SFU}	3 777	4 266	7 896	8 877	16 392	18 354
Number of MSC cells per multiplex frame N_{MUX}	1 259	1 422	2 632	2 959	5 464	6 118
Cell loss per super frame N_L	1	2	1	0	2	0

Table 76: Number of QAM cells for MSC for Mode B

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
Number of available MSC cells per super frame N_{SFA}	2 900	3 330	6 153	7 013	12 747	14 323
Number of useful MSC cells per super frame N_{SFU}	2 898	3 330	6 153	7 011	12 747	14 322
Number of MSC cells per multiplex frame N_{MUX}	966	1 110	2 051	2 337	4 249	4 774
Cell loss per super frame N_L	2	0	0	2	0	1

Table 77: Number of QAM cells for MSC for Mode C

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
Number of available MSC cells per super frame N_{SFA}	-	-	-	5 532	-	11 603
Number of useful MSC cells per super frame N_{SFU}	-	-	-	5 532	-	11 601
Number of MSC cells per multiplex frame N_{MUX}	-	-	-	1 844	-	3 867
Cell loss per super frame N_L	-	-	-	0	-	2

Table 78: Number of QAM cells for MSC for Mode D

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
Number of available MSC cells per super frame N_{SFA}	-	-	-	3 679	-	7 819
Number of useful MSC cells per super frame N_{SFU}	-	-	-	3 678	-	7 818
Number of MSC cells per multiplex frame N_{MUX}	-	-	-	1 226	-	2 606
Cell loss per super frame N_L	-	-	-	1	-	1

So the overall data vector for the useful MSC cells in transmission super frame *m* can be described by:

$$\begin{aligned}\mathbf{S}_m &= \left(s_{m,0}, s_{m,1}, s_{m,2}, \ldots, s_{m,\mathrm{N_{SFU}}-1}\right)\\ &= \left(\hat{\mathbf{Z}}_{3*m}, \hat{\mathbf{Z}}_{3*m+1}, \hat{\mathbf{Z}}_{3*m+2}\right)\\ &= \left(\hat{z}_{3*m,0}, \hat{z}_{3*m,1}, \ldots, \hat{z}_{3*m,\mathrm{N_{MUX}}-1}, \hat{z}_{3*m+1,0}, \hat{z}_{3*m+1,1}, \ldots, \hat{z}_{3*m+1,\mathrm{N_{MUX}}-1}, \hat{z}_{3*m+2,0}, \hat{z}_{3*m+2,1}, \ldots, \hat{z}_{3*m+2,\mathrm{N_{MUX}}-1}\right).\end{aligned}$$

In the case that N_L is unequal to 0 one or two dummy cells, i.e. $\left(\tilde{z}_{m,0}\right)$ or $\left(\tilde{z}_{m,0}, \tilde{z}_{m,1}\right)$, are attached at the end of $\mathbf{S}_m$. Their complex values (i.e. the corresponding QAM symbols) are as defined in table 79.

Table 79: QAM symbols for MSC dummy cells

Number of dummy cells N_L per transmission super frame	Complex values of the dummy cells (QAM symbols)	
	$\tilde{z}_{m,0}$	$\tilde{z}_{m,1}$
1	$a \times (1 + j\,1)$	
2	$a \times (1 + j\,1)$	$a \times (1 - j\,1)$

The value of *a* in table 79 is dependent on the signal constellation chosen for the MSC (see clause 7.4).

8 Transmission structure

8.1 Transmission frame structure and modes

The transmitted signal is organized in transmission super frames.

Each transmission super frame consists of three transmission frames.

Each transmission frame has duration T_f, and consists of N_s OFDM symbols.

Each OFDM symbol is constituted by a set of K carriers and transmitted with a duration T_s.

The spacing between adjacent carriers is $1/T_u$.

The symbol duration is the sum of two parts:

- a useful part with duration T_u;
- a guard interval with duration T_g.

The guard interval consists in a cyclic continuation of the useful part, T_u, and is inserted before it.

The OFDM symbols in a transmission frame are numbered from 0 to N_s - 1.

All symbols contain data and reference information.

Since the OFDM signal comprises many separately modulated carriers, each symbol can in turn be considered to be divided into cells, each cell corresponding to the modulation carried on one carrier during one symbol.

An OFDM frame contains:

- pilot cells;
- control cells;
- data cells.

The pilot cells can be used for frame, frequency and time synchronization, channel estimation, and robustness mode identification.

The transmitted signal is described by the following expression:

$$x(t) = \mathrm{Re}\left\{ e^{j2\pi f_R t} \sum_{r=0}^{\infty} \sum_{s=0}^{N_s-1} \sum_{k=K_{\min}}^{K_{\max}} c_{r,s,k}\ \psi_{r,s,k}(t)_{\ k} \right\}$$

where:

$$\psi_{r,s,k}(t) = \begin{cases} e^{j2\pi \frac{k}{T_u}(t - T_g - sT_s - N_s r T_s)} & (s + N_s\ r)T_s \le t \le (s + N_s\ r + 1)T_s \\ 0 & otherwise \end{cases}$$

and:

N_s	number of OFDM symbols per transmission frame;
k	denotes the carrier number (= $K_{\min}$, … , $K_{\max}$);
s	denotes the OFDM symbol number (= 0... N_s - 1);
r	denotes the transmission frame number (= 0...infinity);
K	is the number of transmitted carriers ($\le K_{\max} - K_{\min}$);
T_s	is the symbol duration;
T_u	is the duration of the useful part of a symbol;
T_g	is the duration of the guard interval;
f_R	is the reference frequency of the RF signal;
$c_{r,s,k}$	complex cell value for carrier k in symbol s of frame number r.

The $c_{r,s,k}$ values depend on the type of cell, as defined below.

For data/control cells (MSC, SDC, FAC), $c_{r,s,k} = z$ where z is the constellation point for each cell as given by the mappings defined in clause 7.

For each reference cell, a defined phase and amplitude is transmitted, $c_{r,s,k} = a_{s,k} U_{s,k}$, where

$a_{s,k}$ is the amplitude, which always takes one of the values $\{1, \sqrt{2}, 2\}$, and

$U_{s,k} = e^{j2\pi\vartheta_{s,k}}$ is a unit-amplitude term of phase $\vartheta_{s,k}$.

$a_{s,k}$ and $\vartheta_{s,k}$ are defined for each type of reference cell in clause 8.4.

8.2 Propagation-related OFDM parameters

OFDM parameters must be chosen to match propagation conditions and the coverage area that the operator wants to serve.

Various sets of OFDM parameters are therefore defined for different conditions of propagation and their parameter values are listed in table 80.

Table 80: Numerical values of the OFDM parameters

Robustness mode	Duration T_u	Carrier spacing $1/T_u$	Duration of guard interval T_g	Duration of symbol $T_s = T_u + T_g$	T_g/T_u	Number of symbols per frame N_s
A	24 ms	$41\ ^{2/3}$ Hz	2,66 ms	26,66 ms	1/9	15
B	21,33 ms	$46\ ^{7/8}$ Hz	5,33 ms	26,66 ms	1/4	15
C	14,66 ms	$68\ ^{2/11}$ Hz	5,33 ms	20 ms	4/11	20
D	9,33 ms	$107\ ^{1/7}$ Hz	7,33 ms	16,66 ms	11/14	24

8.3 Signal bandwidth related parameters

8.3.1 Parameter definition

The OFDM parameters depend upon the available frequency bandwidth, the number of carriers K, and their location with respect to the reference frequency (named DC, in relation with the traditional carrier used in AM analogue transmissions).

The Spectrum occupancy defines the nominal channel bandwidth. The group of carriers carrying the FAC is always to the right (higher in frequency) with respect to the reference frequency, f_R, which is an integer multiple of 1 kHz. Table 81 relates the spectrum occupancy parameter, signalled in the FAC (see clause 6.3), to the nominal channel bandwidth, and figures 32 and 33 show the position of the carriers.

Table 81: Relationship between spectrum occupancy parameter and channel bandwidth

	Spectrum occupancy					
	0	**1**	**2**	**3**	**4**	**5**
Channel bandwidth (kHz)	4,5	5	9	10	18	20

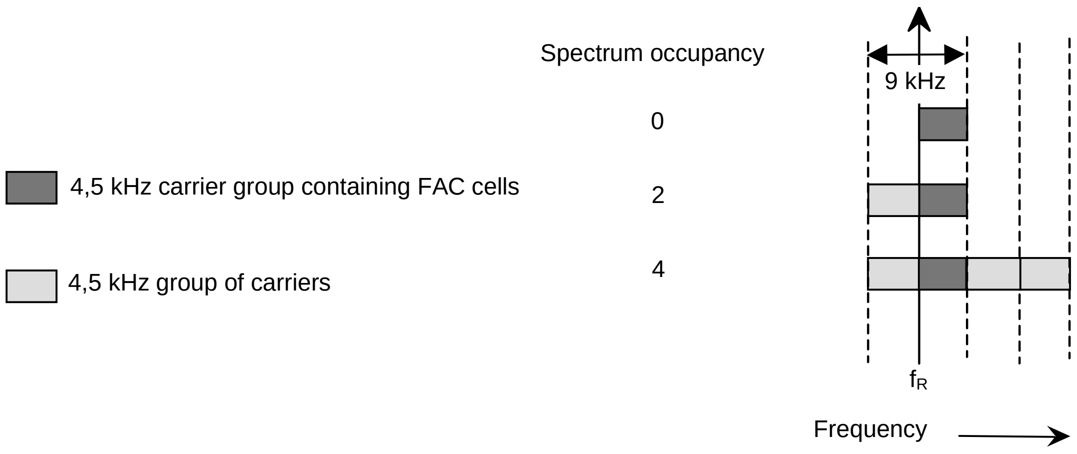

Spectrum occupancy
9 kHz
0
2
4
4,5 kHz carrier group containing FAC cells
4,5 kHz group of carriers
f_R
Frequency

Figure 32: Spectrum occupancy for 9 kHz channels

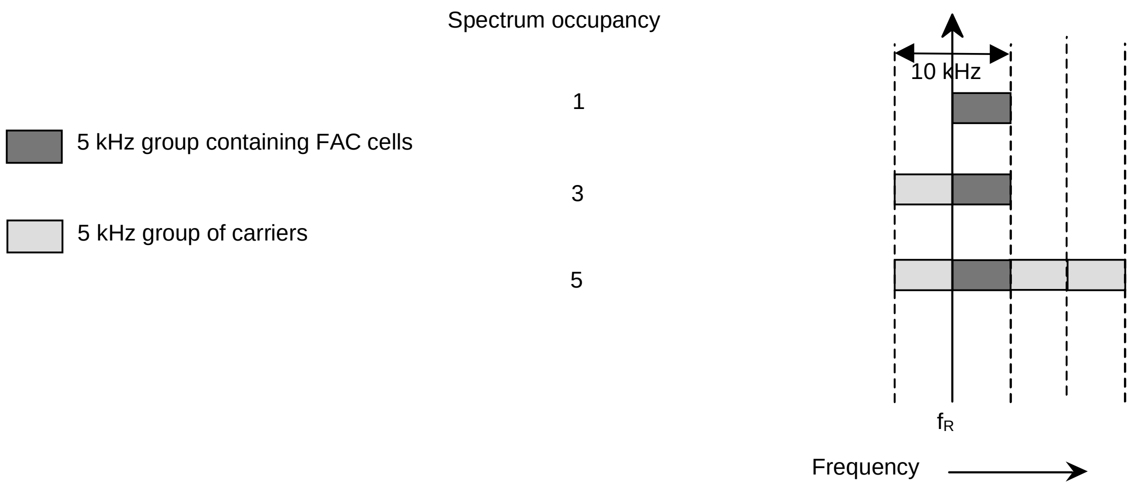


Figure 33: Spectrum occupancy for 10 kHz channels

The carriers are indexed by k ∈ [K_{min}, K_{max}], k = 0 being the DC carrier and determined by the following values depending on the choice made and related to the occupied bandwidth.

Carriers with k < 0 are said to be to the left of DC, and k > 0, to the right of DC.

Table 82 presents the lower and upper carrier numbers for each robustness mode and nominal bandwidth.

Table 82: Carrier numbers for each mode

Robustness mode	**Carrier**	**Spectrum occupancy**					
		0	**1**	**2**	**3**	**4**	**5**
A	K_{min}	2	2	-102	-114	-98	-110
	K_{max}	102	114	102	114	314	350
B	K_{min}	1	1	-91	-103	-87	-99
	K_{max}	91	103	91	103	279	311
C	K_{min}	-	-	-	-69	-	-67
	K_{max}	-	-	-	69	-	213
D	K_{min}	-	-	-	-44	-	-43
	K_{max}	-	-	-	44	-	135

Depending on the mode, the DC carrier and certain carriers around DC are not used, as detailed in table 83.

Table 83: Unused carriers according to robustness mode

Robustness mode	**Unused carrier(s)**
A	$k \in \{-1,0,1\}$
B	$k \in \{0\}$
C	$k \in \{0\}$
D	$k \in \{0\}$

8.3.2 Simulcast transmission

The DRM signal is designed to work in the same broadcast bands as AM signals. Simulcast transmission of services using DRM and AM can be performed by the juxtaposition of the analogue AM signal (DSB or VSB or SSB) and a DRM digital signal. Many arrangements are possible and some are illustrated in annex K.

The spectrum occupancy number relates to the DRM signal. A broadcaster may choose to signal the presence of the AM simulcast by use of the frequency information data entity in the SDC (see clause 6.4.3.4).

8.4 Pilot cells

8.4.1 Functions and derivation

Some cells within the OFDM transmission frame are modulated with known fixed phases and amplitudes.

These cells are pilot cells for channel estimation and synchronization. The positions, amplitudes and phases of these cells are carefully chosen to optimize the performance, especially the initial synchronization duration and reliability. The phases are defined, directly or indirectly, in 1 024ths of a cycle, i.e.

$U_{s,k} = e^{j2\pi\vartheta_{s,k}} = e^{\frac{j2\pi\vartheta_{1024}[s,k]}{1024}}$, where $\vartheta_{1024}[s,k]$ takes integer values and is either explicitly tabulated or derived using integer arithmetic, as defined in the following clauses.

8.4.2 Frequency references

These cells are used by the receiver to detect the presence of the received signal and to estimate its frequency offset. They may also be used for channel estimation and various tracking processes.

8.4.2.1 Cell positions

Frequency references are located at frequencies which are common to all the system variants.

There are three frequency references, which are 750 Hz, 2 250 Hz, and 3 000 Hz as referenced to the DC carrier, as defined in table 84.

Table 84: Carrier numbers for frequency references

Robustness mode	Carrier numbers
A	18, 54, 72
B	16, 48, 64
C	11, 33, 44
D	7, 21, 28

They shall be present in all symbols of each transmission frame.

8.4.2.2 Cell gains and phases

All frequency reference cells shall have a power gain of 2, i.e. $a_{s,k} = \sqrt{2}$, in order to optimize performances at low signal to noise ratio and be compatible when the same cell functions as both a frequency reference and a time reference.

The phases are defined as follows. For the first symbol in the frame (i.e. $s = 0$), the phases $\vartheta_{1024}[s,k]$ are given in table 85.

Table 85: Cell phases for frequency references

Robustness mode	Carrier index, k	Phase index, $\vartheta_{1024}[0,k]$
A	18	205
	54	836
	72	215
B	16	331
	48	651
	64	555
C	11	214
	33	392
	44	242
D	7	788
	21	1 014
	28	332

For subsequent symbols, the phases are chosen in order to ensure the tones are continuous, which is achieved by applying the following rules.

For robustness modes A, B and C, and carrier 28 only of mode D:

$\vartheta_{1024}[s,k] = \vartheta_{1024}[0,k]$

For robustness mode D, carriers 7 and 21:

$\vartheta_{1024}[s,k] = \vartheta_{1024}[0,k]$, for even values of s, and

$\vartheta_{1024}[s,k] = (\vartheta_{1024}[0,k]+512) \bmod 1024$, for odd values of s.

NOTE: This is equivalent to the complex value $U_{s,k}$ multiplied by - 1 for odd values of s.

8.4.3 Time references

These cells are located in the first OFDM symbol of each transmission frame, i.e. $s = 0$.

The time reference cells are mainly used for performing ambiguity resolution since guard time correlation provides a fast and frequency insensitive estimation of time of arrival with a periodicity of one symbol. They are used for determining the first symbol of a transmission frame. They can also be used for frequency-offset estimation.

8.4.3.1 Cell positions and phases

The tables 86 to 89 define the phases of the time reference cells, and the phases of the frequency reference cells for the first symbol of the transmission frame.

$\vartheta_{1024}[0,k]$ is the phase index in 1 024ths of a cycle.

Table 86: Phase of time reference cells for Mode A

Carrier index, k	Phase index, $\vartheta_{1024}[0,k]$
17	973
18 *	205
19	717
21	264
28	357
29	357
32	952
33	440
39	856
40	88
41	88
53	68
54 *	836
55	836
56	836
60	1 008
61	1 008
63	752
71	215
72 *	215
73	727
NOTE: Carrier numbers marked with an asterisk "*" also serve as frequency references (see clause 8.4.2.1); the definitions of phase index are consistent.	

Table 87: Phase of time reference cells for Mode B

Carrier index k	Phase index, $\vartheta_{1024}[0,k]$
14	304
16 *	331
18	108
20	620
24	192
26	704
32	44
36	432
42	588
44	844
48 *	651
49	651
50	651
54	460
56	460
62	944
64 *	555
66	940
68	428
NOTE: Carrier numbers marked with an asterisk "*" also serve as frequency references (see clause 8.4.2.1); the definitions of phase index are consistent.	

Table 88: Phase of time reference cells for Mode C

Carrier index k	Phase index, $\vartheta_{1024}[0,k]$
8	722
10	466
11 *	214
12	214
14	479
16	516
18	260
22	577
24	662
28	3
30	771
32	392
33 *	392
36	37
38	37
42	474
44 *	242
45	242
46	754
NOTE: Carrier numbers marked with an asterisk "*" also serve as frequency references (see clause 8.4.2.1); the definitions of phase index are consistent.	

Table 89: Phase of time reference cells for Mode D

Carrier index k	Phase index, $\vartheta_{1024}[0,k]$
5	636
6	124
7 *	788
8	788
9	200
11	688
12	152
14	920
15	920
17	644
18	388
20	652
21 *	1 014
23	176
24	176
26	752
27	496
28 *	332
29	432
30	964
32	452
NOTE: Carrier numbers marked with an asterisk "*" also serve as frequency references (see clause 8.4.2.1); the definitions of phase index are consistent.	

8.4.3.2 Cell gains

All time reference cells have a power gain of 2,0 in order to optimize performance at low signal to noise ratio, i.e. $a_{s,k} = \sqrt{2}$.

8.4.4 Gain references

The gain reference cells are mainly used for coherent demodulation. These cells are scattered throughout the overall time frequency pattern and are used by the receiver to estimate the channel response.

8.4.4.1 Cell positions

In a transmission frame, for the symbol of index s (ranging from 0 to N_s - 1), carriers for which index k belongs to the subsets as defined in table 90 are gain references.

Table 90: Carrier indices k for gain reference cells

Robustness mode	Subset	Condition	Periodicity of the gain reference pattern
A	k = 2 + 4 × (s mod 5) + 20 × p	p integer $k_{\min} \le k \le k_{\max}$	5 symbols
B	k = 1 + 2 × (s mod 3) + 6 × p	p integer $k_{\min} \le k \le k_{\max}$	3 symbols
C	k = 1 + 2 × (s mod 2) + 4 × p	p integer $k_{\min} \le k \le k_{\max}$	2 symbols
D	k = 1 + (s mod 3) + 3 × p	p integer $k_{\min} \le k \le k_{\max}$	3 symbols

NOTE: The gain reference cell patterns have been chosen such that the edge carriers are included as gain reference cell positions.

Annex L gives some example figures illustrating the position of the gain reference cells.

8.4.4.2 Cell gains

Gain reference cells mostly have a power gain of 2 (i.e. $a_{s,k} = \sqrt{2}$), in order to optimize performances at low signal to noise ratio. However, gain reference cells close to the band lower and upper edges are over-boosted by a further power gain of 2 (i.e. overall power gain of 4, so that the amplitude $a_{s,k} = 2$) as defined in table 91.

Table 91: Carrier numbers given a power boost of 4, i.e. $a_{s,k} = 2$

Robustness mode	Spectrum occupancy					
	0	**1**	**2**	**3**	**4**	**5**
A	2, 6, 98, 102	2, 6, 110, 114	-102, -98, 98, 102	-114, -110, 110, 114	-98, -94, 310, 314	-110, -106, 346, 350
B	1, 3, 89, 91	1, 3, 101, 103	-91, -89, 89, 91	-103, -101, 101, 103	-87, -85, 277, 279	-99, -97, 309, 311
C	-	-	-	-69, -67, 67, 69	-	-67, -65, 211, 213
D	-	-	-	-44, -43, 43, 44	-	-43, -42, 134, 135

8.4.4.3 Cell phases

In some cases gain references fall in locations which coincide with those already defined for either frequency or time references. In these cases, the phase definitions given in clauses 8.4.2 and 8.4.3 take precedence.

In all other locations, the phases of the gain reference cells are obtained by integer arithmetic applied to small tables of values, as defined in the following procedure.

8.4.4.3.1 Procedure for calculation of cell phases

The procedure is:

First, compute values of *m*, *n* and *p* for each cell, where the carrier number is *k* and the symbol number is *s*:

$$\begin{aligned} n &= s \bmod y, \\ m &= \lfloor s / y \rfloor \\ p &= \frac{k - k_0 - nx}{xy} \end{aligned}$$

x, y, and k_0 are constants which are defined for each robustness mode in table 92.

Table 92: Definition of x, y, k_0

Robustness mode	*x*	*y*	k_0
A	4	5	2
B	2	3	1
C	2	2	1
D	1	3	1

NOTE 1: The value of *p* obtained by this procedure is an integer, as a consequence of the definition of reference cell locations in clause 8.4.4.1; while the values of *n* and *m* are integer by definition of the mathematical operations producing them.

Secondly, calculate the integer phase index by the following formula:

$$\vartheta_{1024}[s,k] = \left(4Z_{256}[n,m] + pW_{1024}[n,m] + p^2(1+s)Q_{1024}\right) \bmod 1024$$

Q_{1024} and the small tables $Z_{256}[n,m]$ and $W_{1024}[n,m]$ are defined for each robustness mode in the following clauses.

NOTE 2: The values in the tables $Z_{256}[n,m]$ and $W_{1024}[n,m]$ may be represented precisely as 8- and 10-bit unsigned integers respectively; similarly Q_{1024} may be represented precisely as a 10-bit unsigned integer.

8.4.4.3.2 Robustness Mode A

The $W_{1024}[n,m]$ matrix is defined as:

$$W_{1024}[n,m] = \{ \begin{array}{rrr} \{228, & 341, & 455\}, \\ \{455, & 569, & 683\}, \\ \{683, & 796, & 910\}, \\ \{910, & 0, & 114\}, \\ \{114, & 228, & 341\}\} \end{array}$$

The $Z_{256}[n,m]$ matrix is defined as:

$$Z_{256}[n,m] = \{ \begin{array}{rrr} \{0, & 81, & 248\}, \\ \{18, & 106, & 106\}, \\ \{122, & 116, & 31\}, \\ \{129, & 129, & 39\}, \\ \{33, & 32, & 111\}\} \end{array}$$

$Q_{1024} = 36$.

8.4.4.3.3 Robustness Mode B

The $W_{1024}[n,m]$ matrix is defined as:

$$W_{1024}[n,m] = \{ \begin{array}{rrrrr} \{512, & 0, & 512, & 0, & 512\}, \\ \{0, & 512, & 0, & 512, & 0\}, \\ \{512, & 0, & 512, & 0, & 512\}\} \end{array}$$

The $Z_{256}[n,m]$ matrix is defined as:

$$Z_{256}[n,m] = \{ \begin{array}{rrrrr} \{0, & 57, & 164, & 64, & 12\}, \\ \{168, & 255, & 161, & 106, & 118\}, \\ \{25, & 232, & 132, & 233, & 38\}\} \end{array}$$

$Q_{1024} = 12$.

8.4.4.3.4 Robustness Mode C

The $W_{1024}[n,m]$ matrix is defined as:

$$W_{1024}[n,m] = \{ \begin{array}{rrrrrrrrrr} \{465, & 372, & 279, & 186, & 93, & 0, & 931, & 838, & 745, & 652\}, \\ \{931, & 838, & 745, & 652, & 559, & 465, & 372, & 279, & 186, & 93\}\} \end{array}$$

The $Z_{256}[n,m]$ matrix is defined as:

$$Z_{256}[n,m] = \{ \begin{array}{rrrrrrrrrr} \{0, & 76, & 29, & 76, & 9, & 190, & 161, & 248, & 33, & 108\}, \\ \{179, & 178, & 83, & 253, & 127, & 105, & 101, & 198, & 250, & 145\}\} \end{array}$$

$Q_{1024} = 12$.

8.4.4.3.5 Robustness Mode D

The $W_{1024}[n,m]$ matrix is defined as:

$$W_{1024}[n,m] = \{ \begin{matrix} \{366, & 439, & 512, & 585, & 658, & 731, & 805, & 878\}, \\ \{731, & 805, & 878, & 951, & 0, & 73, & 146, & 219\}, \\ \{73, & 146, & 219, & 293, & 366, & 439, & 512, & 585\}\} \end{matrix}$$

The $Z_{256}[n,m]$ matrix is defined as:

$$Z_{256}[n,m] = \{ \begin{matrix} \{0, & 240, & 17, & 60, & 220, & 38, & 151, & 101\}, \\ \{110, & 7, & 78, & 82, & 175, & 150, & 106, & 25\}, \\ \{165, & 7, & 252, & 124, & 253, & 177, & 197, & 142\}\} \end{matrix}$$

$Q_{1024} = 14$.

8.5 Control cells

8.5.1 General

The control cells consist of two parts:

- the Fast Access Channel (FAC), integrated in every transmission frame. It is used to quickly obtain the necessary information for the receiver to be able to demodulate the DRM signal;
- the Service Description Channel (SDC), repeated every transmission super-frame. It contains all the additional information that describes the services currently available. The SDC is also used for Alternative Frequency Switching (AFS).

Figure 34 describes the time-frequency location of these signals.

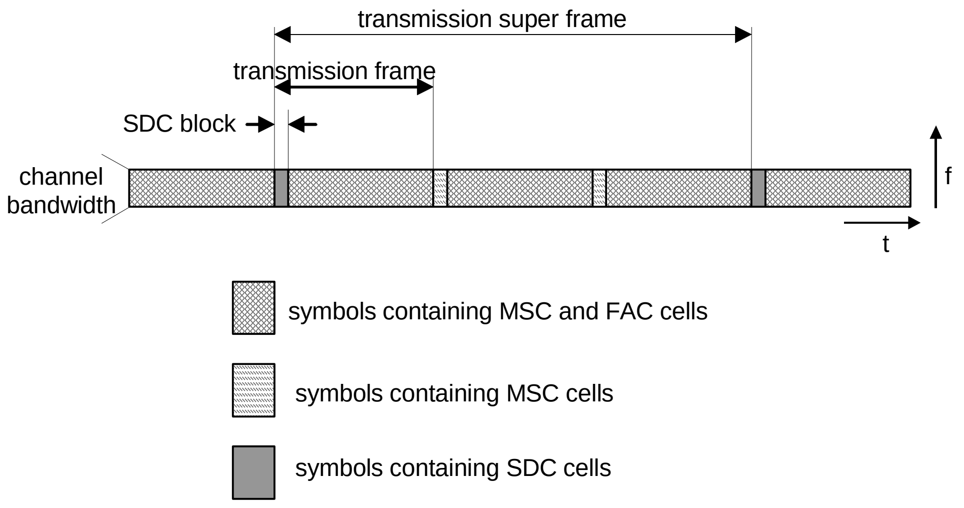

transmission super frame
transmission frame
SDC block
channel bandwidth
f
t
symbols containing MSC and FAC cells
symbols containing MSC cells
symbols containing SDC cells

Figure 34: Time-frequency location of FAC and SDC signals

8.5.2 FAC cells

8.5.2.1 Cell positions

The cells used for FAC are cells that are neither frequency references, nor time references, nor gain references, nor data cells in the symbols that do not contain the SDC.

FAC cells convey highly protected QAM symbols that allow fast detection by the receiver of the type of signal it is currently receiving.

There are always 65 FAC cells. Tables 93 to 96 give the position of the FAC cells for each robustness mode.

Table 93: Position of the FAC cells in robustness Mode A

Symbol	Carrier number
0	
1	
2	26, 46, 66, 86
3	10, 30, 50, 70, 90
4	14, 22, 34, 62, 74, 94
5	26, 38, 58, 66, 78
6	22, 30, 42, 62, 70, 82
7	26, 34, 46, 66, 74, 86
8	10, 30, 38, 50, 58, 70, 78, 90
9	14, 22, 34, 42, 62, 74, 82, 94
10	26, 38, 46, 66, 86
11	10, 30, 50, 70, 90
12	14, 34, 74, 94
13	38, 58, 78
14	

Table 94: Position of the FAC cells in robustness Mode B

Symbol	Carrier number
0	
1	
2	13, 25, 43, 55, 67
3	15, 27, 45, 57, 69
4	17, 29, 47, 59, 71
5	19, 31, 49, 61, 73
6	9, 21, 33, 51, 63, 75
7	11, 23, 35, 53, 65, 77
8	13, 25, 37, 55, 67, 79
9	15, 27, 39, 57, 69, 81
10	17, 29, 41, 59, 71, 83
11	19, 31, 43, 61, 73
12	21, 33, 45, 63, 75
13	23, 35, 47, 65, 77
14	

Table 95: Position of the FAC cells in robustness Mode C

Symbol	Carrier number
0	
1	
2	
3	9, 21, 45, 57
4	23, 35, 47
5	13, 25, 37, 49
6	15, 27, 39, 51
7	5, 17, 29, 41, 53
8	7, 19, 31, 43, 55
9	9, 21, 45, 57
10	23, 35, 47
11	13, 25, 37, 49
12	15, 27, 39, 51
13	5, 17, 29, 41, 53
14	7, 19, 31, 43, 55
15	9, 21, 45, 57
16	23, 35, 47
17	13, 25, 37, 49
18	15, 27, 39, 51
19	

Table 96: Position of the FAC cells in robustness Mode D

Symbol	Carrier number
0	
1	
2	
3	9, 18, 27
4	10, 19
5	11, 20, 29
6	12, 30
7	13, 22, 31
8	5, 14, 23, 32
9	6, 15, 24, 33
10	16, 25, 34
11	8, 17, 26, 35
12	9, 18, 27, 36
13	10, 19, 37
14	11, 20, 29
15	12, 30
16	13, 22, 31
17	5, 14, 23, 32
18	6, 15, 24, 33
19	16, 25, 34
20	8, 17, 26, 35
21	9, 18, 27, 36
22	10, 19, 37
23	

8.5.2.2 Cell gains and phases

The $c_{r,s,k}$ values are normalized modulation values of the constellation point z according to the modulation alphabet used for the FAC (4-QAM) (see figure 31).

Successive constellation points are assigned to the FAC cells of a transmission frame in order of increasing carrier index *k*, starting from the most negative *k*; then in time order starting from the first FAC bearing symbol of the frame.

8.5.3 SDC cells

8.5.3.1 Cell positions

The cells used for SDC are all the cells in the SDC symbols which are neither frequency references, nor time references, nor gain references for which $k_{\min} \le k \le k_{\max}$ and k does not belong to the subset of unused carriers defined above. For robustness modes A and B, the SDC symbols are symbols 0 and 1 of each transmission super frame. For robustness modes C and D, the SDC symbols are symbols 0, 1 and 2 of each transmission super frame.

8.5.3.2 Cell gains and phases

The $c_{r,s,k}$ values are normalized modulation values of the constellation point z according to the modulation alphabet used for the SDC (16- or 4-QAM, see figures 30 and 31).

Successive constellation points are assigned to the SDC cells of a transmission super frame in order of increasing carrier index *k*, starting from the most negative *k*; then in time order starting from the first SDC bearing symbol of the super frame.

8.6 Data cells

8.6.1 Cell positions

Data cells are all cells which are neither pilot cells, nor control cells; for which $k_{\min} \le k \le k_{\max}$ and k does not belong to the subset of unused carriers defined above.

8.6.2 Cell gains and phases

The $c_{r,s,k}$ values are the normalized modulation values of the constellation point z according to the modulation alphabet used for the MSC (64- or 16-QAM, see figures 27 to 30) taken from the vector S_m (see clause 7.7).

Successive elements $s_{m,i}$ are assigned to the cells of a transmission super frame in order of increasing carrier index *k*, starting from the most negative *k*; then in time order starting from the first non-SDC symbol of the super frame.

Annex A (informative): Simulated system performance

The table gives simulated system performance anticipating perfect channel estimation and the absence of phase noise and quantization effects. Channel decoding is assumed to be done with a multistage decoder with two iterations.

The results are given for five of the channels of annex B, whereby the associated OFDM modes are the Mode A for the channels 1 and 2, the Mode B mode for the channels 3 to 5. The associated code rate is R = 0,6 and the modulation is 64-QAM. The signal power includes pilots and the guard interval.

Table A.1: Required C/N for a transmission to achieve a BER = 1 x 10^{-4} after the channel decoder for the MSC

Channel model	Code rate R = 0,6
Channel 1	14,9 dB
Channel 2	16,5 dB
Channel 3	23,2 dB
Channel 4	22,3 dB
Channel 5	20,4 dB

Annex B (informative): Definition of channel profiles

The channels to be considered are the LF, MF and HF broadcast radio transmission channels. In principle all three are multipath channels because the surface of the earth and the ionosphere are involved in the mechanism of electromagnetic wave propagation.

The approach is to use stochastic time-varying models with a stationary statistics and define models for good, moderate and bad conditions by taking appropriate parameter values of the general model. One of those models with adaptable parameters is the Wide Sense Stationary Uncorrelated Scattering model (WSSUS model). The justification for the stationary approach with different parameter sets is, that results on real channels lead to BER curves between best and worst cases found in the simulation.

The channel models have been generated from the following equations where e(t) and s(t) are the complex envelopes of the input and output signals respectively:

$$s(t) = \sum_{k=1}^{n} \rho_k c_k(t) e(t - \Delta_k) \qquad \text{(B.1)}$$

This is a tapped delay-line where:

- ρ_k is the attenuation of the path number k - listed in table B.1
- Δ_k is the relative delay of the path number k - listed in table B.1
- the time-variant tap weights $\{c_k(t)\}$ are zero mean complex-valued stationary gaussian random processes. The magnitudes $|c_k(t)|$ are Rayleigh-distributed and the phases Φ (t) are uniformly distributed.

For each weight $\{c_k(t)\}$ there is one stochastic process, characterized by its variance and its power density spectrum (PDS). The variance is a measure for the average signal power which is received via this path and is defined by the relative attenuation ρ_k - listed in table B.1 - and the PDS determines the average speed of variation in time. The width of the PDS is quantified by a number and is referred to as the Doppler spread D_{sp} of that path - listed in table B.1.

There might be also a non-zero centre frequency of the PDS, which can be interpreted as an average frequency shift or Doppler shift D_{sh} - listed in table B.1.

The PDS is modelled by filtering of white noise (i.e. with constant PDS) and is equal to:

$$\phi_{n_t n_t}(f) = N_0 \cdot |H(f)|^2 \qquad \text{(B.2)}$$

H(f) is the transfer function of the filter. The stochastic processes belonging to every individual path then become Rayleigh processes. For the ionospheric path, a Gaussian shape has proven to be a good approach with respect to real observations.

The Doppler profile on each path k is then defined as:

$$|H(f)|^2 = \frac{1}{\sqrt{2\pi\sigma_d^2}} \; e^{-\frac{(f - Dsh)^2}{2\sigma_d^2}} \qquad \text{(B.3)}$$

The Doppler spread is specified as 2-sided and contains 68 % of the power:

$$D_{sp} = 2\,\sigma_d \qquad \text{(B.4)}$$

Table B.1: Set of channels

Channel no 1: AWGN		**good** **typical/moderate** **bad**		**LF, MF,HF** **LF, var.SNR**
	path 1	path 2	path 3	path 4
Delay (Δ_k)	0			
Path gain, rms (ρ_k)	1			
Doppler shift (D_{sh})	0			
Doppler spread (D_{sp})	0			

Channel no 2: Rice with delay		**good** **typical/moderate** **bad**		**MF, HF**
	path 1	path 2	path 3	path 4
Delay (Δ_k)	0	1 ms		
Path gain, rms (ρ_k)	1	0,5		
Doppler shift (D_{sh})	0	0		
Doppler spread (D_{sp})	0	0,1 Hz		

Channel no 3: US Consortium		**good** **typical/moderate** **bad**		**HF** **MF**
	path 1	path 2	path 3	path 4
Delay (Δ_k)	0	0,7 ms	1,5 ms	2,2 ms
Path gain, rms (ρ_k)	1	0,7	0,5	0,25
Doppler shift (D_{sh})	0,1 Hz	0,2 Hz	0,5 Hz	1,0 Hz
Doppler spread (D_{sp})	0,1 Hz	0,5 Hz	1,0 Hz	2,0 Hz

Channel no 4: CCIR Poor		**good** **typical/moderate** **bad**		**HF**
	path 1	path 2	path 3	path 4
Delay (Δ_k)	0	2 ms		
Path gain, rms (ρ_k)	1	1		
Doppler shift (D_{sh})	0	0		
Doppler spread (D_{sp})	1 Hz	1 Hz		

Channel no 5		**good** **typical/moderate** **bad**		**HF**
	path 1	path 2	path 3	path 4
Delay (Δ_k)	0	4 ms		
Path gain, rms (ρ_k)	1	1		
Doppler shift (D_{sh})	0	0		
Doppler spread (D_{sp})	2 Hz	2 Hz		

Channel no 6		**good** **typical/moderate** **bad**		**HF**
	path 1	path 2	path 3	path 4
Delay (Δ_k)	0	2 ms	4 ms	6 ms
Path gain, rms (ρ_k)	0,5	1	0,25	0,0625
Doppler shift (D_{sh})	0	1,2 Hz	2,4 Hz	3,6 Hz
Doppler spread (D_{sp})	0,1 Hz	2,4 Hz	4,8 Hz	7,2 Hz

Annex C (informative): Example of mapping of logical frames to multiplex frames

There are many service and stream combinations possible within the DRM system. One example is illustrated in this annex.

This example DRM signal contains two services: an audio service (service A) and a data service (service D). The audio service also carries a data application.

UEP is applied to the audio service. The data application carried with the audio service uses the lower protection. The data service uses the higher protection. The code rates chosen are 0,5 and 0,6 corresponding to protection level 0 and 1 respectively.

Service A consists of two streams: stream 0 carries the audio, stream 1 carries the data application.

Service D consists of one stream: stream 2.

Stream 0 is carried in logical frames L0, stream 1 is carried in logical frames L1 and stream 2 is carried in logical frames L2.

L0 has 266 bytes in the higher protected part (part A) with protection level 0, and 798 bytes in the lower protected part (part B) with protection level 1.

L1 has 59 bytes in the lower protected part (part B) with protection level 1.

L2 has 19 bytes in the higher protected part (part A) with protection level 0.

The resulting multiplex frame is illustrated in figure C.1.

Protection level 0		**Protection level 1**	
Stream 0	Stream 2	Stream 0	Stream 1
266 bytes	19 Bytes	798 Bytes	59 Bytes

Figure C.1

The multiplex description data entity is coded as follows:

Field name	Field size	Field value
length	7	9
version number	1	0
type	4	0
protection level for part A	2	0
protection level for part B	2	1
data length of part A (stream 0)	12	266
data length of part B (stream 0)	12	798
data length of part A (stream 1)	12	0
data length of part B (stream 1)	12	59
data length of part A (stream 2)	12	19
data length of part B (stream 2)	12	0

Annex D (normative): Calculation of the CRC word

The implementation of Cyclic Redundancy Check codes (CRC-codes) allows the detection of transmission errors at the receiver side. For this purpose CRC words shall be included in the transmitted data. These CRC words shall be defined by the result of the procedure described in this annex.

A CRC code is defined by a polynomial of degree *n*:

$$G_n(x) = x^n + g_{n-1}x^{n-1} + \ldots + g_2x^2 + g_1x + 1$$

with $n \geq 1$:

and: $$g_i \in \{0,1\}, \quad i = 1.....n-1$$

The CRC calculation may be performed by means of a shift register containing *n* register stages, equivalent to the degree of the polynomial (see figure D.1). The stages are denoted by b_0 ... b_{n-1}, where b_0 corresponds to 1, b_1 to *x*, b_2 to x^2, ..., b_{n-1} to x^{n-1}. The shift register is tapped by inserting XORs at the input of those stages, where the corresponding coefficients g_i of the polynomial are "1".

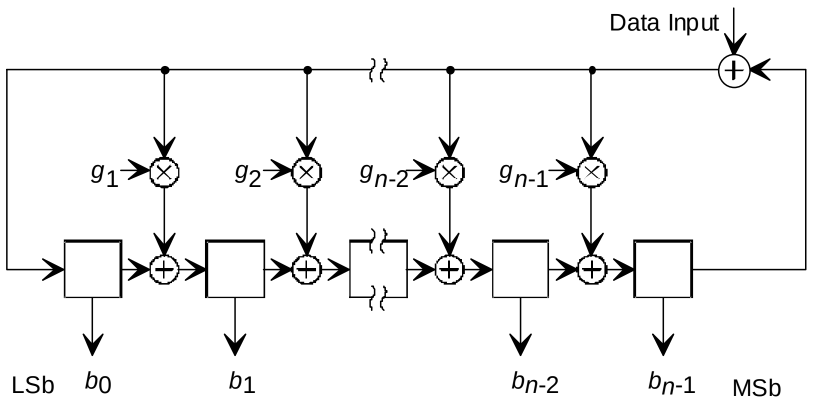


Figure D.1: General CRC block diagram

At the beginning of the CRC calculation, all register stage contents are initialized to all ones. After applying the first bit of the data block (MSb first) to the input, the shift clock causes the register to shift its content by one stage towards the MSb stage (b_{n-1}), while loading the tapped stages with the result of the appropriate XOR operations. The procedure is then repeated for each data bit. Following the shift after applying the last bit (LSb) of the data block to the input, the shift register contains the CRC word which is then read out. Data and CRC word are transmitted with MSb first. The CRC shall be inverted (1's complement) prior to transmission.

The CRC codes used in the DRM system are based on the following polynomials:

- $G_{16}(x) = x^{16} + x^{12} + x^5 + 1$
- $G_8(x) = x^8 + x^4 + x^3 + x^2 + 1$
- $G_6(x) = x^6 + x^5 + x^3 + x^2 + x + 1$
- $G_5(x) = x^5 + x^4 + x^2 + x + 1$
- $G_3(x) = x^3 + x + 1$
- $G_2(x) = x^2 + x + 1$
- $G_1(x) = x + 1$

The assignment of the polynomials to the respective applications is given in each clause.

Annex E (informative): Indicative RF Protection ratios

Protection ratios are required for:

- AM interfered with by DRM digital signals;
- DRM digital signals interfered with by AM;
- DRM digital signals interfered with by DRM digital signals.

Indicative RF protection ratios for different channel spacings are given in the following tables. However, these values depend very much on the development of system components, e.g. on receiver and transmitter characteristics. Therefore, the given values have to be considered as preliminary until measurement results and calculations are available, which are based on the performance of a broader quantity of equipment.

Table E.1: AM interfered with by DRM digital signals

Δf/kHz	**Relative RF Protection Ratios a_{rf}/dB (a_{af} = 0 dB)**			
0	6,6	6,1	6,3	5,8
5	3,2	2,7	3,2	2,8
9	-32,6	-27,7	-27,5	-23,2
10	-41,1	-36,4	-33,6	-31,2
15	-54,2	-53,6	-43,5	-42,3
18	-56,9	-56,8	-46,9	-45,6
20	-56,8	-56,5	-48,8	-47,6
Wanted	**AM**			
Interferer	**DRM Synthesizer**		**DRM Spectrum Mask**	
Mode	Mode A 9 kHz 64-QAM	Mode B 10 kHz 64-QAM	Mode A 9 kHz 64-QAM	Mode B 10 kHz 64-QAM

These values are valid for AM signals with high compression and a modern AM receiver with an RF bandwidth of 4,4 kHz and a slope of 35 dB/octave (attenuation at a frequency separation of 5 kHz: 41,5 dB).

The values shown in the table above are derived from the measured spectrum distributions of the synthesizer signals and the AM spectrum masks from ITU-R Recommendation SM.328-10 [10], which were adapted to the bandwidths and modulation of the DRM signals.

Table E.2: DRM digital signals interfered with by AM

Δf/kHz	**RF Protection Ratios a_{rf}/dB**					
0	0,6	7,1	1,2	7,5	-1,7	4,6
0,050	0,6	7,2				
0,300	3,4	29,5	1,0	8,5		
5	-7,4	-1,1	-3,9	1,8		
9	(-45,8)	(29,9)				
10	-59,6	-34,1	-53,4	-30,9		
15	< -60	-39,7	< -60	-37,5		
18	< -60		< -60			
20	< -60	-42,3	< -60	-39,8		
Wanted	**DRM**					
Mode	Mode A 9 kHz 64-QAM	Mode A 9 kHz 64-QAM	Mode B 10 kHz 64-QAM	Mode B 10 kHz 64-QAM	Mode B 10 kHz 16-QAM	Mode B 10 kHz 16-QAM
Interferer	**AM**					
Mode	m = 25 % (compr. normal)	m = 53 % (compr. high)	m = 25 % (compr. normal)	m = 53 % (compr. high)	m = 25 % (compr. normal)	m = 53 % (compr. high)

These values are valid for a wanted signal having a BER of 10^{-4} and an unwanted AM signal modulated with coloured noise according to ITU-R Recommendation BS.559-2 [9].

Table E.3: DRM digital signals interfered with by DRM digital signals

Δf/kHz	**RF Protection Ratios a $_{rf}$/dB**					
0	17,3	17,4	13,3	(17,3)	(17,4)	(13,3)
5	12,4	12,3		(13,0)	(12,7)	
9	-32,1			(0,8)		
10		-35,1			(-2,3)	
15		-39,1			(-27,6)	
18	-46,3			(-36,9)		
20		-44,4			(-34,9)	
Wanted	**DRM**			**DRM**		
Mode	Mode A 9 kHz 64-QAM	Mode B 10 kHz 64-QAM	Mode B 10 kHz 16-QAM	Mode A 9 kHz 64-QAM	Mode B 10 kHz 64-QAM	Mode B 10 kHz 16-QAM
Interferer	**DRM Synthesizer**			**DRM Spectrum Mask**		
Mode	Mode A 9 kHz 64-QAM	Mode B 10 kHz 64-QAM	Mode B 10 kHz 16-QAM	Mode A 9 kHz 64-QAM	Mode B 10 kHz 64-QAM	Mode B 10 kHz 16-QAM

These values are valid for a wanted signal having a BER of 10^{-4}.

The values in brackets shown in the table above are derived from a comparison of the measured spectrum distributions of the synthesizer signals and the AM spectrum masks from ITU-R Recommendation SM.328-10 [10], which were adapted to the bandwidths and modulation of the DRM signals.

Annex F (informative): Guidelines for transmitter implementation

Void.

Annex G (informative): Guidelines for receiver implementation

G.1 Alternative Frequency Checking and Switching (AFS)

Alternative frequency switching (AFS) provides the functionality of seamlessly checking for the availability of the same programme material on a differing frequency and then switching to it if it is valid. Alternative frequencies can be signalled by use of SDC data entity type 3. The various steps of this process are indicated in figure G.1.

AFS specific mathematical symbols are defined as follows:

T_d:	time delay at point of reception between the current and the possible alternative frequency.
T_{tune}:	time needed by the receiver to tune to the alternative frequency.
T_{check}:	time available to acquire the data required for the validation of the AF.

Procedure:

At the start of known SDC block on the tuned frequency, the receiver re-tunes to the alternative frequency. It acquires the data necessary to perform the AF-check and immediately tunes back to the original tuned frequency. This process has to be completed within the time interval T_{check} . Subsequently the validity of the alternative frequency can be computed before the next occurrence of the SDC. Subject to the validation of the alternative frequency, the receiver may choose to switch to the new frequency at this point without an interruption of service.

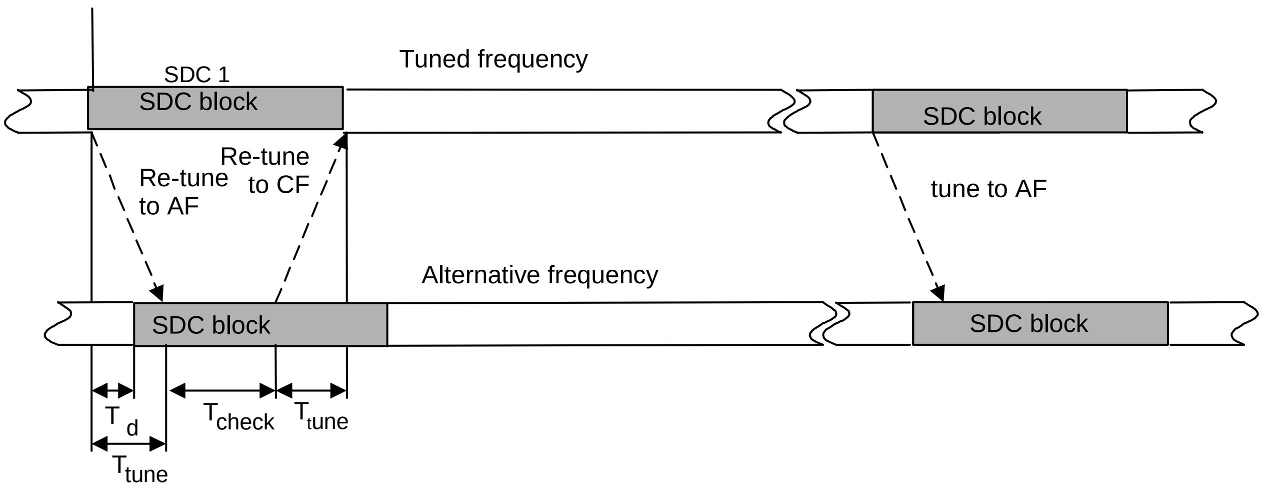


Figure G.1: Illustration of AFS function

The points at which the receiver may check the alternative frequencies are governed by the Identity field in the FAC in combination with the AFS index signalled in the SDC.

If the receiver detects a failure of the FAC CRC for the first transmission frame of the transmission super frame then it cannot perform an AFS check because the value of the Identity field is unknown.

For fully dynamic operation (see clause 6.4.5), no AFS is possible because the receiver has no knowledge of the data that will be sent in future SDC blocks.

For fully static operation, the AFS function may be performed every super transmission frame, provided that the receiver has stored all the different SDC blocks in the cycle. The number of SDC blocks in the cycle is given by the AFS index + 1.

For semi-dynamic operation, the AFS function may only be performed at certain transmission super frames. The following examples illustrate some of the many possibilities.

EXAMPLE 1:

Changing the content of the SDC block (A to B) with AFS index = 0:

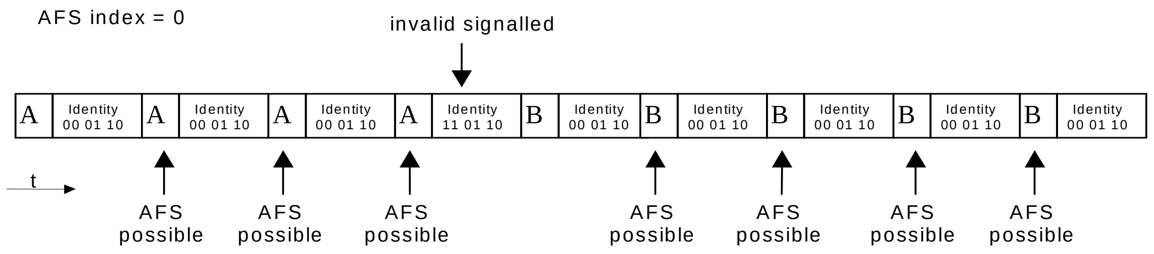


Figure G.2: Example 1

NOTE 1:

- very fast AFS possibility after tuning;
- very limited SDC data size when AFS feature should be used.

EXAMPLE 2:

Changing the content of both SDC blocks (A to C; B to D) with AFS index = 1:

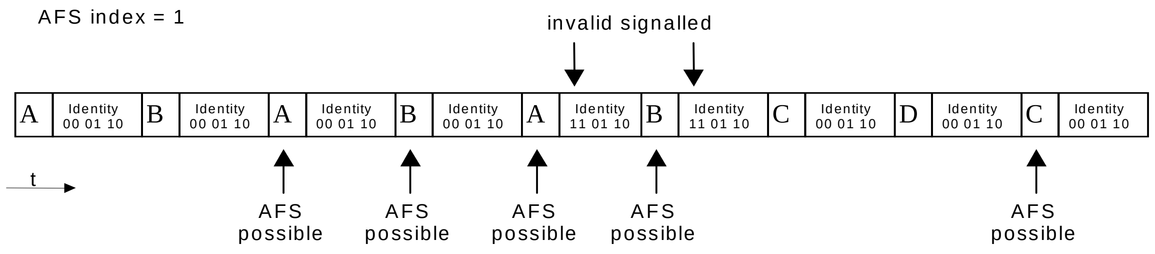


Figure G.3: Example 2

NOTE 2:

- while changing the SDC blocks in two consecutive SDC blocks is no AFS possible;
- with AFS = n the first AFS can occur after (n + 1) received SDC blocks.

EXAMPLE 3:

Changing the content of one SDC block (A to C) with AFS index = 1

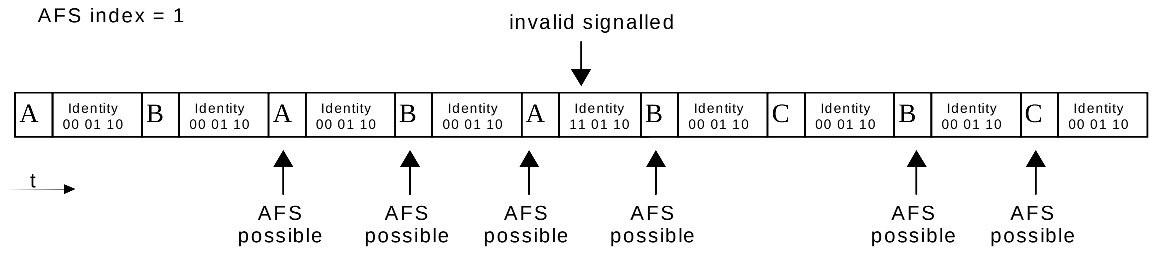


Figure G.4: Example 3

NOTE 3:

- only one AFS possibility is missed.

EXAMPLE 4:

Continuous changing of one SDC block (B to C to D...) with AFS index = 1

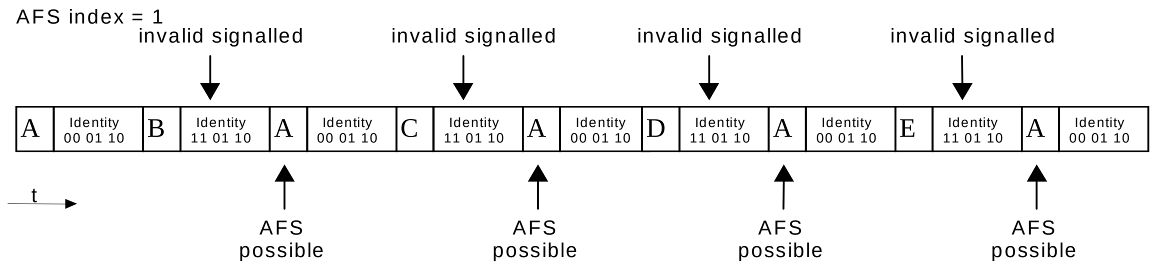


Figure G.5: Example 4

NOTE 4:

- only every second frame AFS is possible;
- SDC data size is increased.

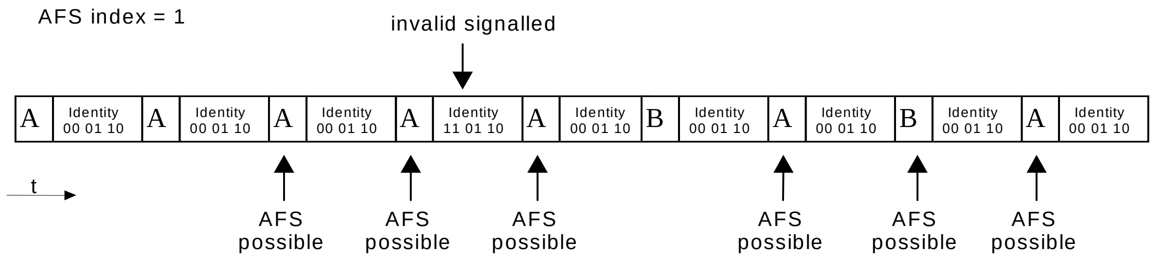


Figure G.6: Example 5

EXAMPLE 5:

Change of repetition rate of SDC block (without reconfiguration) with AFS index = 1

NOTE 5:

- first AFS possible after n frames after tuning;
- flexible SDC data size.

For other values of AFS index, similar schemes can be applied.

G.2 Character sets

DRM uses UTF-8 character coding, and so allows all known characters to be broadcast. However, receivers may be produced that do not support all characters. In this case the following behaviour is recommended:

- For the label (sent using SDC data entity type 1): if the receiver character set does not support the characters being broadcast (for example, the label is composed of Hindi characters, but the receiver only supports European characters) then the receiver should display the Service Identifier instead.
- For the text message (carried in the MSC): if the receiver character set does not support all of the characters being broadcast then the receiver should attempt to determine if any portion of the message can be displayed. This would be, for example, if at least 10 consecutive characters could be displayed.

Annex H (informative): Service capacity and bit rates

The following table gives the orders of magnitude of the available total bit rates, which depend upon the signal bandwidth, the protection mode, and the error correction code rates.

For a 64-QAM modulation, a coding rate of 0,6, for the MSC (EEP SM):

Robustness mode	**Spectrum occupancy**					
	0	**1**	**2**	**3**	**4**	**5**
A	11,3 kbit/s	12,8 kbit/s	23,6 kbit/s	26,6 kbit/s	49,1 kbit/s	55,0 kbit/s
B	8,7 kbit/s	10,0 kbit/s	18,4 kbit/s	21,0 kbit/s	38,2 kbit/s	43,0 kbit/s
C	-	-	-	16,6 kbit/s	-	34,8 kbit/s
D	-	-	-	11,0 kbit/s	-	23,4 kbit/s

For a 16-QAM modulation, a coding rate of 0,62, for the MSC (EEP SM):

Robustness mode	**Spectrum occupancy**					
	0	**1**	**2**	**3**	**4**	**5**
A	7,8 kbit/s	8,9 kbit/s	16,4 kbit/s	18,5 kbit/s	34,1 kbit/s	38,2 kbit/s
B	6,0 kbit/s	6,9 kbit/s	12,8 kbit/s	14,6 kbit/s	26,5 kbit/s	29,8 kbit/s
C	-	-	-	11,5 kbit/s	-	24,1 kbit/s
D	-	-	-	7,6 kbit/s	-	16,3 kbit/s

Minimum absolute (R = 0,50, 16-QAM, Mode B, 4,5 kHz) 4,8 kbit/s.

Maximum absolute (R = *0,78*, 64-QAM, Mode A, 20 kHz) 72,0 kbit/s.

Annex I (normative): SBR tables

I.1 SBR definitions and notation

I.1.1 Definitions

Band	(as in limiter band, noise floor band, etc.) a group of consecutive QMF channels
Chirp factor	the bandwidth expansion factor of the formants described by a LPC polynomial
Envelope	a vector of envelope scalefactors
Envelope scalefactor	an element representing the averaged energy of a signal over a region described by a frequency band and a time segment
Frame	basic unit that can be decoded on itself (header information is required for general decoder settings)
Frequency band	interval in frequency, group of consecutive QMF channels
Frequency border	frequency band delimiter, expressed as a specific QMF channel
Noise floor	a vector of noise floor scalefactors
Noise floor scalefactor	a noise floor element associated with a region described by a frequency band and a time segment, representing the ratio between the energy of the noise to be added to the envelope adjusted HF generated signal and the energy of the same
Patch	a number of adjoining QMF-channels moved to a different frequency location
SBR range	is the frequency range of the signal generated by the SBR algorithm
Time border	time segment delimiter, expressed as a specific time slot
Time segment	interval in time, group of consecutive time slots
Time/Frequency grid	a description of envelope time segments and associated frequency resolution tables as well as description of noise floor time segments
Time slot	finest resolution in time for envelopes and noise floors). In single rate mode, one time slot equals one subsample in the QMF domain. In multi rate mode, one time slot equals two subsamples in the QMF domain

I.1.2 Notation

Vectors are indicated by bold lower-case names, e.g. **vector**.

Matrices (and vectors of vectors) are indicated by bold upper-case single letter names, e.g. **M**.

Variables are indicated by italic, e.g. *variable*.

Functions are indicated as *func*(*x*).

Bitstream elements are indicated as multiple-word names with prefix "bs_", e.g. bs_bitstream_element.

For equations in the text, normal mathematical interpretation is assumed.

$\mathbf{Q}_{Mapped}\left(m - lsb, l\right) = \mathbf{Q}_{Orig}\left(i, k\left(l\right)\right), \mathbf{f}_{TableNoise}(i) \le m < \mathbf{f}_{TableNoise}(i+1), 0 \le i < N_Q, 0 \le l < L_E$
where $k(l)$ is defined by $\mathbf{t}\left(l\right) \ge \mathbf{t}_Q\left(k\left(l\right)\right), \mathbf{t}\left(l+1\right) \le \mathbf{t}_Q\left(k\left(l\right)+1\right)$

Hence the following example from the text, should be interpreted as follows:

$\mathbf{Q}_{Mapped}\left(m - lsb, l\right)$ equals $\mathbf{Q}_{Orig}\left(i, k\left(l\right)\right)$ for $\mathbf{f}_{TableNoise}(i) \le m < \mathbf{f}_{TableNoise}(i+1)$ and $0 \le i < N_Q$ and $0 \le l < L_E$. The function $k\left(l\right)$ returns, for a given l, a value for which $\mathbf{t}\left(l\right) \ge \mathbf{t}_Q\left(k\left(l\right)\right)$ and $\mathbf{t}\left(l+1\right) \le \mathbf{t}_Q\left(k\left(l\right)+1\right)$ is true. The result is a $\mathbf{Q}_{Mapped}$ matrix that is piecewise constant.

The expression $\sum_{k=a}^{b} f\left(k\right)$ evaluates to zero if $b < a$.

I.1.3 Conventions

Scalar operations:

$y = \left(z\right)\mathrm{mod}\left(x\right)$. y is x-modulus of z.

$y = x/z$. y is equal to x divided by z without rounding nor truncation.

Vector operations:

$\mathbf{y} = sort\left(\mathbf{x}\right)$. $\mathbf{y}$ is equal to the sorted vector $\mathbf{x}$, where $\mathbf{x}$ is sorted in ascending order.

$y = length\left(\mathbf{x}\right)$. y is the number of elements of the vector $\mathbf{x}$.

Constants:

ε A constant to avoid division by zero, e.g. 96 dB below maximum signal input.

$HI = 1$ Index used for envelope high frequency resolution.

$LO = 0$ Index used for envelope low frequency resolution.

$NOISE\_FLOOR\_OFFSET = 6$ Offset of noise floor.

$NO\_TIME\_SLOTS = 16$ Number of SBR envelope time slots that exist within an AAC frame.

$UN\_MAP = 0.000244141$ Un-mapping of the pan-value (balance from left to right channels in stereo) used for de-quantization of the envelope.

Variables:

ch is the current channel.

$\mathbf{E}_{Orig}$	has L_E columns where each column is of length N_{Low} or N_{High} depending on the frequency resolution for each envelope. Each element in $\mathbf{E}_{orig}$ contains the envelope scalefactors of the original signal
$\mathbf{f} = [f_0, ..., f_{L-1}]$	frequency resolution for all envelopes in the current frame, zero for low resolution, one for high resolution
$\mathbf{f}_{Master}$	is of length $N_{Master} + 1$ and contains QMF master frequency grouping information
$\mathbf{f}_{TableHigh}$	is of length $N_{High} + 1$ and contains frequency borders for high frequency resolution envelopes

$\mathbf{f}_{TableLow}$	is of length $N_{Low}+1$ and contains frequency borders for low frequency resolution envelopes
$\mathbf{f}_{TableLim}$	is of length N_L+1 and contains frequency borders used by the limiters
$\mathbf{f}_{TableNoise}$	is of length N_Q+1 and contains frequency borders used by noise floors
$\mathbf{F}=\left[\mathbf{f}_{TableLow},\mathbf{f}_{TableHigh}\right]$	has two column vectors containing the frequency border tables for low and high frequency resolution
Fs	sampling rate of the SBR enhanced signal
$k0$	the first QMF channel in the $\mathbf{f}_{Master}$ table
L_E	number of envelopes
L_Q	number of noise floors
lsb	the first QMF channel in the SBR range
M	number of QMF channels in the SBR range
$middleBorder$	points to a specific time border
$\mathbf{n}=[N_{Low},N_{High}]$	number of frequency bands for low and high frequency resolution
$N_{ShortEnvelope}$	maximum number of time slots that define a short envelope
N_L	number of limiter bands
N_Q	number of noise floor bands
N_{Master}	number of frequency bands for the master frequency resolution
$\mathbf{panOffset}=[12,24]$	offset-values for the envelope and noise floor data, when using coupled channels
$\mathbf{Q}_{Orig}$	has L_Q columns where each column is of length N_Q and contains the noise floor scalefactors
Reset	a variable in the encoder and the decoder set to one if certain bitstream elements have changed from the previous frame, otherwise set to zero
$\mathbf{t}_E$	is of length L_E+1 and contains start and stop time borders for all envelopes in the current frame
$\mathbf{t}_Q$	is of length L_Q+1 and contains start and stop time borders for all noise floors in the current frame
t_{HfAdj}	offset for the envelope adjuster module
t_{HfGen}	offset for the HF-generation module
$\mathbf{X}_{Low}$	is the complex input QMF bank subband matrix to the HF generator
$\mathbf{X}_{High}$	is the complex input QMF bank subband matrix to the HF adjuster

Y	is the complex output QMF bank subband matrix from the HF adjuster

I.2 SBR Frequency table

Table I.1: Coefficients c[i] of the QMF bank window

c[0] = -0,00055252865047	c[1] = -0,00056176925738	c[2] = -0,00049475180896	c[3] = -0,00048752279712
c[4] = -0,00048937912498	c[5] = -0,00050407143497	c[6] = -0,00052265642972	c[7] = -0,00054665656337
c[8] = -0,00056778025613	c[9] = -0,00058709304852	c[10] = -0,00061327473938	c[11] = -0,00063124935319
c[12] = -0,00065403333621	c[13] = -0,00067776907764	c[14] = -0,00069416146273	c[15] = -0,00071577364744
c[16] = -0,00072550431222	c[17] = -0,00074409418541	c[18] = -0,00074905980532	c[19] = -0,00076813719270
c[20] = -0,00077248485949	c[21] = -0,00078343322877	c[22] = -0,00077798694927	c[23] = -0,00078036647100
c[24] = -0,00078014496257	c[25] = -0,00077579773310	c[26] = -0,00076307935757	c[27] = -0,00075300014201
c[28] = -0,00073193571525	c[29] = -0,00072153919876	c[30] = -0,00069179375372	c[31] = -0,00066504150893
c[32] = -0,00063415949025	c[33] = -0,00059461189330	c[34] = -0,00055645763906	c[35] = -0,00051455722108
c[36] = -0,00046063254803	c[37] = -0,00040951214522	c[38] = -0,00035011758756	c[39] = -0,00028969811748
c[40] = -0,00020983373440	c[41] = -0,00014463809349	c[42] = -0,00006173344072	c[43] = 0,00001349497418
c[44] = 0,00010943831274	c[45] = 0,00020430170688	c[46] = 0,00029495311041	c[47] = 0,00040265402160
c[48] = 0,00051073884952	c[49] = 0,00062393761391	c[50] = 0,00074580258865	c[51] = 0,00086084433262
c[52] = 0,00098859883015	c[53] = 0,00112501551307	c[54] = 0,00125778846475	c[55] = 0,00139024948272
c[56] = 0,00154432198471	c[57] = 0,00168680832531	c[58] = 0,00183482654224	c[59] = 0,00198411407369
c[60] = 0,00214615835557	c[61] = 0,00230172547746	c[62] = 0,00246256169126	c[63] = 0,00262017586902
c[64] = 0,00278704643465	c[65] = 0,00294694477165	c[66] = 0,00311254206525	c[67] = 0,00327396134847
c[68] = 0,00344188741828	c[69] = 0,00360082681231	c[70] = 0,00376039229104	c[71] = 0,00392074323703
c[72] = 0,00408197531935	c[73] = 0,00422642692270	c[74] = 0,00437307196781	c[75] = 0,00452098527825
c[76] = 0,00466064606118	c[77] = 0,00479325608498	c[78] = 0,00491376035745	c[79] = 0,00503930226013
c[80] = 0,00514073539032	c[81] = 0,00524611661324	c[82] = 0,00534716811982	c[83] = 0,00541967759307
c[84] = 0,00548760401507	c[85] = 0,00554757145088	c[86] = 0,00559380230045	c[87] = 0,00562206432097
c[88] = 0,00564551969164	c[89] = 0,00563891995151	c[90] = 0,00562661141932	c[91] = 0,00559171286630
c[92] = 0,00554043639400	c[93] = 0,00547537830770	c[94] = 0,00538389758970	c[95] = 0,00527157587272
c[96] = 0,00513822754514	c[97] = 0,00498396877629	c[98] = 0,00481094690600	c[99] = 0,00460395301471
c[100] = 0,00438018617447	c[101] = 0,00412516423270	c[102] = 0,00384564081246	c[103] = 0,00354012465507
c[104] = 0,00320918858098	c[105] = 0,00284467578623	c[106] = 0,00245085400321	c[107] = 0,00202741761850
c[108] = 0,00157846825768	c[109] = 0,00109023290512	c[110] = 0,00058322642480	c[111] = 0,00002760451905
c[112] = -0,00054642808664	c[113] = -0,00115681355227	c[114] = -0,00180394725893	c[115] = -0,00248267236449
c[116] = -0,00319337783900	c[117] = -0,00394011240522	c[118] = -0,00472225962400	c[119] = -0,00553372111088
c[120] = -0,00637922932685	c[121] = -0,00726158168517	c[122] = -0,00817982333726	c[123] = -0,00913253296085
c[124] = -0,01011502154986	c[125] = -0,01113155480321	c[126] = -0,01218499959508	c[127] = -0,01327182200351
c[128] = 0,01439046660792	c[129] = 0,01554055533423	c[130] = 0,01673247129989	c[131] = 0,01794333813443
c[132] = 0,01918724313698	c[133] = 0,02045317933555	c[134] = 0,02174675502535	c[135] = 0,02306801692862
c[136] = 0,02441609920285	c[137] = 0,02578758475467	c[138] = 0,02718594296329	c[139] = 0,02860721736385
c[140] = 0,03005026574279	c[141] = 0,03150176087389	c[142] = 0,03297540810337	c[143] = 0,03446209487686
c[144] = 0,03596975605542	c[145] = 0,03748128504252	c[146] = 0,03900536794745	c[147] = 0,04053491705584
c[148] = 0,04206490946367	c[149] = 0,04360975421304	c[150] = 0,04514884056413	c[151] = 0,04668430272642
c[152] = 0,04821657200672	c[153] = 0,04973857556014	c[154] = 0,05125561555216	c[155] = 0,05276307465207
c[156] = 0,05424527683589	c[157] = 0,05571736482138	c[158] = 0,05716164501299	c[159] = 0,05859156836260
c[160] = 0,05998374801761	c[161] = 0,06134551717207	c[162] = 0,06268578081172	c[163] = 0,06397158980681
c[164] = 0,06522471064380	c[165] = 0,06643675122104	c[166] = 0,06760759851228	c[167] = 0,06870438283512
c[168] = 0,06976302447127	c[169] = 0,07076287107266	c[170] = 0,07170026731102	c[171] = 0,07256825833083
c[172] = 0,07336202550803	c[173] = 0,07410036424342	c[174] = 0,07474525581194	c[175] = 0,07531373362019
c[176] = 0,07580083586584	c[177] = 0,07619924793396	c[178] = 0,07649921704119	c[179] = 0,07670934904245
c[180] = 0,07681739756964	c[181] = 0,07682300113923	c[182] = 0,07672049241746	c[183] = 0,07650507183194
c[184] = 0,07617483218536	c[185] = 0,07573057565061	c[186] = 0,07515762552870	c[187] = 0,07446643947564
c[188] = 0,07364060057620	c[189] = 0,07267746427299	c[190] = 0,07158263647903	c[191] = 0,07035330735093
c[192] = 0,06896640131951	c[193] = 0,06745250215166	c[194] = 0,06576906686508	c[195] = 0,06394448059633
c[196] = 0,06196027790387	c[197] = 0,05981665708090	c[198] = 0,05751526919867	c[199] = 0,05504600343009
c[200] = 0,05240938217366	c[201] = 0,04959786763445	c[202] = 0,04663033051701	c[203] = 0,04347687821958
c[204] = 0,04014582784127	c[205] = 0,03664181168133	c[206] = 0,03295839306691	c[207] = 0,02908240060125
c[208] = 0,02503075618909	c[209] = 0,02079970728622	c[210] = 0,01637012582228	c[211] = 0,01176238327857
c[212] = 0,00696368621617	c[213] = 0,00197656014503	c[214] = -0,00320868968304	c[215] = -0,00857117491366
c[216] = -0,01412888273558	c[217] = -0,01988341292573	c[218] = -0,02582272888064	c[219] = -0,03195312745332
c[220] = -0,03827765720822	c[221] = -0,04478068215856	c[222] = -0,05148041767934	c[223] = -0,05837053268336
c[224] = -0,06544098531359	c[225] = -0,07269433008129	c[226] = -0,08013729344279	c[227] = -0,08775475365593

c[228] = -0,09555333528914	c[229] = -0,10353295311463	c[230] = -0,11168269317730	c[231] = -0,12000779846800
c[232] = -0,12850028503878	c[233] = -0,13715517611934	c[234] = -0,14597664911870	c[235] = -0,15496070710605
c[236] = -0,16409588556669	c[237] = -0,17338081721706	c[238] = -0,18281725485142	c[239] = -0,19239667457267
c[240] = -0,20212501768103	c[241] = -0,21197358538056	c[242] = -0,22196526964149	c[243] = -0,23206908706791
c[244] = -0,24230168845974	c[245] = -0,25264803095722	c[246] = -0,26310532994603	c[247] = -0,27366340405625
c[248] = -0,28432141891085	c[249] = -0,29507167170646	c[250] = -0,30590985751916	c[251] = -0,31682789136456
c[252] = -0,32781137272105	c[253] = -0,33887226938665	c[254] = -0,34999141229310	c[255] = -0,36115899031355
c[256] = 0,37237955463061	c[257] = 0,38363500139043	c[258] = 0,39492117615675	c[259] = 0,40623176767625
c[260] = 0,41756968968409	c[261] = 0,42891199207373	c[262] = 0,44025537543665	c[263] = 0,45159965356824
c[264] = 0,46293080852757	c[265] = 0,47424532146115	c[266] = 0,48552530911099	c[267] = 0,49677082545707
c[268] = 0,50798175000434	c[269] = 0,51912349702391	c[270] = 0,53022408956855	c[271] = 0,54125534487322
c[272] = 0,55220512585061	c[273] = 0,56307891401370	c[274] = 0,57385241316923	c[275] = 0,58454032354679
c[276] = 0,59511230862496	c[277] = 0,60557835389180	c[278] = 0,61591099320291	c[279] = 0,62612426956055
c[280] = 0,63619801077286	c[281] = 0,64612696959461	c[282] = 0,65590163024671	c[283] = 0,66551398801627
c[284] = 0,67496631901712	c[285] = 0,68423532934598	c[286] = 0,69332823767032	c[287] = 0,70223887193539
c[288] = 0,71094104263095	c[289] = 0,71944626349561	c[290] = 0,72774489002994	c[291] = 0,73582117582769
c[292] = 0,74368278636488	c[293] = 0,75131374561237	c[294] = 0,75870807608242	c[295] = 0,76586748650939
c[296] = 0,77277808813327	c[207] = 0,77942875190216	c[298] = 0,78583531203920	c[299] = 0,79197358416424
c[300] = 0,79784664137700	c[301] = 0,80344857518505	c[302] = 0,80876950044491	c[303] = 0,81381912706217
c[304] = 0,81857760046468	c[304] = 0,82304198905409	c[306] = 0,82722753473360	c[307] = 0,83110384571520
c[308] = 0,83469373618402	c[309] = 0,83797173378865	c[310] = 0,84095413924722	c[311] = 0,84362382812005
c[312] = 0,84598184698206	c[313] = 0,84803157770763	c[314] = 0,84978051984268	c[315] = 0,85119715249343
c[316] = 0,85230470352147	c[317] = 0,85310209497017	c[318] = 0,85357205739107	c[319] = 0,85373856005937
c[320] = 0,85357205739107	c[321] = 0,85310209497017	c[322] = 0,85230470352147	c[323] = 0,85119715249343
c[324] = 0,84978051984268	c[325] = 0,84803157770763	c[326] = 0,84598184698206	c[327] = 0,84362382812005
c[328] = 0,84095413924722	c[329] = 0,83797173378865	c[330] = 0,83469373618402	c[331] = 0,83110384571520
c[332] = 0,82722753473360	c[333] = 0,82304198905409	c[334] = 0,81857760046468	c[335] = 0,81381912706217
c[336] = 0,80876950044491	c[337] = 0,80344857518505	c[338] = 0,79784664137700	c[339] = 0,79197358416424
c[340] = 0,78583531203920	c[341] = 0,77942875190216	c[342] = 0,77277808813327	c[343] = 0,76586748650939
c[344] = 0,75870807608242	c[345] = 0,75131374561237	c[346] = 0,74368278636488	c[347] = 0,73582117582769
c[348] = 0,72774489002994	c[349] = 0,71944626349561	c[350] = 0,71094104263095	c[351] = 0,70223887193539
c[352] = 0,69332823767032	c[353] = 0,68423532934598	c[354] = 0,67496631901712	c[355] = 0,66551398801627
c[356] = 0,65590163024671	c[357] = 0,64612696959461	c[358] = 0,63619801077286	c[359] = 0,62612426956055
c[360] = 0,61591099320291	c[361] = 0,60557835389180	c[362] = 0,59511230862496	c[363] = 0,58454032354679
c[364] = 0,57385241316923	c[365] = 0,56307891401370	c[366] = 0,55220512585061	c[367] = 0,54125534487322
c[368] = 0,53022408956855	c[369] = 0,51912349702391	c[370] = 0,50798175000434	c[371] = 0,49677082545707
c[372] = 0,48552530911099	c[373] = 0,47424532146115	c[374] = 0,46293080852757	c[375] = 0,45159965356824
c[376] = 0,44025537543665	c[377] = 0,42891199207373	c[378] = 0,41756968968409	c[379] = 0,40623176767625
c[380] = 0,39492117615675	c[381] = 0,38363500139043	c[382] = 0,37237955463061	c[383] = 0,36115899031355
c[384] = -0,34999141229310	c[385] = -0,33887226938665	c[386] = -0,32781137272105	c[387] = -0,31682789136456
c[388] = -0,30590985751916	c[389] = -0,29507167170646	c[390] = -0,28432141891085	c[391] = -0,27366340405625
c[392] = -0,26310532994603	c[393] = -0,25264803095722	c[394] = -0,24230168845974	c[395] = -0,23206908706791
c[396] = -0,22196526964149	c[397] = -0,21197358538056	c[398] = -0,20212501768103	c[399] = -0,19239667457267
c[400] = -0,18281725485142	c[401] = -0,17338081721706	c[402] = -0,16409588556669	c[403] = -0,15496070710605
c[404] = -0,14597664911870	c[405] = -0,13715517611934	c[406] = -0,12850028503878	c[407] = -0,12000779846800
c[408] = -0,11168269317730	c[409] = -0,10353295311463	c[410] = -0,09555333528914	c[411] = -0,08775475365593
c[412] = -0,08013729344279	c[413] = -0,07269433008129	c[414] = -0,06544098531359	c[415] = -0,05837053268336
c[416] = -0,05148041767934	c[417] = -0,04478068215856	c[418] = -0,03827765720822	c[419] = -0,03195312745332
c[420] = -0,02582272888064	c[421] = -0,01988341292573	c[422] = -0,01412888273558	c[423] = -0,00857117491366
c[424] = -0,00320868968304	c[425] = 0,00197656014503	c[426] = 0,00696368621617	c[427] = 0,01176238327857
c[428] = 0,01637012582228	c[429] = 0,02079970728622	c[430] = 0,02503075618909	c[431] = 0,02908240060125
c[432] = 0,03295839306691	c[433] = 0,03664181168133	c[434] = 0,04014582784127	c[435] = 0,04347687821958
c[436] = 0,04663033051701	c[437] = 0,04959786763445	c[438] = 0,05240938217366	c[439] = 0,05504600343009
c[440] = 0,05751526919867	c[441] = 0,05981665708090	c[442] = 0,06196027790387	c[443] = 0,06394448059633
c[444] = 0,06576906686508	c[445] = 0,06745250215166	c[446] = 0,06896640131951	c[447] = 0,07035330735093
c[448] = 0,07158263647903	c[449] = 0,07267746427299	c[450] = 0,07364060057620	c[451] = 0,07446643947564
c[452] = 0,07515762552870	c[453] = 0,07573057565061	c[454] = 0,07617483218536	c[455] = 0,07650507183194
c[456] = 0,07672049241746	c[457] = 0,07682300113923	c[458] = 0,07681739756964	c[459] = 0,07670934904245
c[460] = 0,07649921704119	c[461] = 0,07619924793396	c[462] = 0,07580083586584	c[463] = 0,07531373362019
c[464] = 0,07474525581194	c[465] = 0,07410036424342	c[466] = 0,07336202550803	c[467] = 0,07256825833083
c[468] = 0,07170026731102	c[469] = 0,07076287107266	c[470] = 0,06976302447127	c[471] = 0,06870438283512
c[472] = 0,06760759851228	c[473] = 0,06643675122104	c[474] = 0,06522471064380	c[475] = 0,06397158980681
c[476] = 0,06268578081172	c[477] = 0,06134551717207	c[478] = 0,05998374801761	c[479] = 0,05859156836260
c[480] = 0,05716164501299	c[481] = 0,05571736482138	c[482] = 0,05424527683589	c[483] = 0,05276307465207
c[484] = 0,05125561555216	c[485] = 0,04973857556014	c[486] = 0,04821657200672	c[487] = 0,04668430272642

c[488] = 0,04514884056413	c[489] = 0,04360975421304	c[490] = 0,04206490946367	c[491] = 0,04053491705584
c[492] = 0,03900536794745	c[493] = 0,03748128504252	c[494] = 0,03596975605542	c[495] = 0,03446209487686
c[496] = 0,03297540810337	c[497] = 0,03150176087389	c[498] = 0,03005026574279	c[499] = 0,02860721736385
c[500] = 0,02718594296329	c[501] = 0,02578758475467	c[502] = 0,02441609920285	c[503] = 0,02306801692862
c[504] = 0,02174675502535	c[505] = 0,02045317933555	c[506] = 0,01918724313698	c[507] = 0,01794333813443
c[508] = 0,01673247129989	c[509] = 0,01554055533423	c[510] = 0,01439046660792	c[511] = 0,01327182200351
c[512] = -0,01218499959508	c[513] = -0,01113155480321	c[514] = -0,01011502154986	c[515] = -0,00913253296085
c[516] = -0,00817982333726	c[517] = -0,00726158168517	c[518] = -0,00637922932685	c[519] = -0,00553372111088
c[520] = -0,00472225962400	c[521] = -0,00394011240522	c[522] = -0,00319337783900	c[523] = -0,00248267236449
c[524] = -0,00180394725893	c[525] = -0,00115681355227	c[526] = -0,00054642808664	c[527] = 0,00002760451905
c[528] = 0,00058322642480	c[529] = 0,00109023290512	c[530] = 0,00157846825768	c[531] = 0,00202741761850
c[532] = 0,00245085400321	c[533] = 0,00284467578623	c[534] = 0,00320918858098	c[535] = 0,00354012465507
c[536] = 0,00384564081246	c[537] = 0,00412516423270	c[538] = 0,00438018617447	c[539] = 0,00460395301471
c[540] = 0,00481094690600	c[541] = 0,00498396877629	c[542] = 0,00513822754514	c[543] = 0,00527157587272
c[544] = 0,00538389758970	c[545] = 0,00547537830770	c[546] = 0,00554043639400	c[547] = 0,00559171286630
c[548] = 0,00562661141932	c[549] = 0,00563891995151	c[550] = 0,00564551969164	c[551] = 0,00562206432097
c[552] = 0,00559380230045	c[553] = 0,00554757145088	c[554] = 0,00548760401507	c[555] = 0,00541967759307
c[556] = 0,00534716811982	c[557] = 0,00524611661324	c[558] = 0,00514073539032	c[559] = 0,00503930226013
c[560] = 0,00491376035745	c[561] = 0,00479325608498	c[562] = 0,00466064606118	c[563] = 0,00452098527825
c[564] = 0,00437307196781	c[565] = 0,00422642692270	c[566] = 0,00408197531935	c[567] = 0,00392074323703
c[568] = 0,00376039229104	c[569] = 0,00360082681231	c[570] = 0,00344188741828	c[571] = 0,00327396134847
c[572] = 0,00311254206525	c[573] = 0,00294694477165	c[574] = 0,00278704643465	c[575] = 0,00262017586902
c[576] = 0,00246256169126	c[577] = 0,00230172547746	c[578] = 0,00214615835557	c[579] = 0,00198411407369
c[580] = 0,00183482654224	c[581] = 0,00168680832531	c[582] = 0,00154432198471	c[583] = 0,00139024948272
c[584] = 0,00125778846475	c[585] = 0,00112501551307	c[586] = 0,00098859883015	c[587] = 0,00086084433262
c[588] = 0,00074580258865	c[589] = 0,00062393761391	c[590] = 0,00051073884952	c[591] = 0,00040265402160
c[592] = 0,00029495311041	c[593] = 0,00020430170688	c[594] = 0,00010943831274	c[595] = 0,00001349497418
c[596] = -0,00006173344072	c[597] = -0,00014463809349	c[598] = -0,00020983373440	c[599] = -0,00028969811748
c[600] = -0,00035011758756	c[601] = -0,00040951214522	c[602] = -0,00046063254803	c[603] = -0,00051455722108
c[604] = -0,00055645763906	c[605] = -0,00059461189330	c[606] = -0,00063415949025	c[607] = -0,00066504150893
c[608] = -0,00069179375372	c[609] = -0,00072153919876	c[610] = -0,00073193571525	c[611] = -0,00075300014201
c[612] = -0,00076307935757	c[613] = -0,00077579773310	c[614] = -0,00078014496257	c[615] = -0,00078036647100
c[616] = -0,00077798694927	c[617] = -0,00078343322877	c[618] = -0,00077248485949	c[619] = -0,00076813719270
c[620] = -0,00074905980532	c[621] = -0,00074409418541	c[622] = -0,00072550431222	c[623] = -0,00071577364744
c[624] = -0,00069416146273	c[625] = -0,00067776907764	c[626] = -0,00065403333621	c[627] = -0,00063124935319
c[628] = -0,00061327473938	c[629] = -0,00058709304852	c[630] = -0,00056778025613	c[631] = -0,00054665656337
c[632] = -0,00052265642972	c[633] = -0,00050407143497	c[634] = -0,00048937912498	c[635] = -0,00048752279712
c[636] = -0,00049475180896	c[637] = -0,00056176925738	c[638] = -0,00055252865047	c[639] = 0

I.3 SBR Huffman tables

The function *huff_dec*() is used as:

data = *huff_dec*(*t_huff*, *codeword*),

where *t_huff* is the selected Huffman table and *codeword* is the word read from the bitstream. The returned value, *data* is the index in the Huffman table with an offset of the corresponding largest absolute value (LAV) of the table.

Table I.2: SBR Huffman tables overview

table name	dt_env	dt_noise	amp_res	LAV	Notes
t_huffman_env_1_5dB	1	dc	0	60	
f_huffman_env_1_5dB	0	dc	0	60	
t_huffman_env_bal_1_5dB	1	dc	0	24	
f_huffman_env_bal_1_5dB	0	dc	0	24	
t_huffman_env_3_0dB	1	dc	1	31	
f_huffman_env_3_0dB	0	dc	1	31	
t_huffman_env_bal_3_0dB	1	dc	1	12	
f_huffman_env_bal_3_0dB	0	dc	1	12	
t_huffman_noise_3_0dB	dc	1	dc	31	
f_huffman_noise_3_0dB	dc	0	dc	31	1
t_huffman_noise_bal_3_0dB	dc	1	dc	12	
f_huffman_noise_bal_3_0dB	dc	0	dc	12	1
NOTE: The Huffman tables of f_huffman_noise_3_0dB and f_huffman_noise_bal_3_0dB are the same as for f_huffman_env_3_0dB and f_huffman_env_bal_3_0dB, respectively.					

Table I.3: t_huffman_env_1_5dB

index	Length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x00000012	0x0003FFD6	61	0x00000003	0x00000004
1	0x00000012	0x0003FFD7	62	0x00000004	0x0000000C
2	0x00000012	0x0003FFD8	63	0x00000005	0x0000001C
3	0x00000012	0x0003FFD9	64	0x00000006	0x0000003C
4	0x00000012	0x0003FFDA	65	0x00000007	0x0000007C
5	0x00000012	0x0003FFDB	66	0x00000008	0x000000FC
6	0x00000013	0x0007FFB8	67	0x00000009	0x000001FC
7	0x00000013	0x0007FFB9	68	0x0000000A	0x000003FD
8	0x00000013	0x0007FFBA	69	0x0000000C	0x00000FFA
9	0x00000013	0x0007FFBB	70	0x0000000D	0x00001FF8
10	0x00000013	0x0007FFBC	71	0x0000000E	0x00003FF6
11	0x00000013	0x0007FFBD	72	0x0000000E	0x00003FF8
12	0x00000013	0x0007FFBE	73	0x0000000F	0x00007FF5
13	0x00000013	0x0007FFBF	74	0x00000010	0x0000FFEF
14	0x00000013	0x0007FFC0	75	0x00000011	0x0001FFE8
15	0x00000013	0x0007FFC1	76	0x00000010	0x0000FFF2
16	0x00000013	0x0007FFC2	77	0x00000013	0x0007FFD4
17	0x00000013	0x0007FFC3	78	0x00000013	0x0007FFD5
18	0x00000013	0x0007FFC4	79	0x00000013	0x0007FFD6
19	0x00000013	0x0007FFC5	80	0x00000013	0x0007FFD7
20	0x00000013	0x0007FFC6	81	0x00000013	0x0007FFD8
21	0x00000013	0x0007FFC7	82	0x00000013	0x0007FFD9
22	0x00000013	0x0007FFC8	83	0x00000013	0x0007FFDA
23	0x00000013	0x0007FFC9	84	0x00000013	0x0007FFDB
24	0x00000013	0x0007FFCA	85	0x00000013	0x0007FFDC
25	0x00000013	0x0007FFCB	86	0x00000013	0x0007FFDD
26	0x00000013	0x0007FFCC	87	0x00000013	0x0007FFDE
27	0x00000013	0x0007FFCD	88	0x00000013	0x0007FFDF
28	0x00000013	0x0007FFCE	89	0x00000013	0x0007FFE0
29	0x00000013	0x0007FFCF	90	0x00000013	0x0007FFE1
30	0x00000013	0x0007FFD0	91	0x00000013	0x0007FFE2
31	0x00000013	0x0007FFD1	92	0x00000013	0x0007FFE3
32	0x00000013	0x0007FFD2	93	0x00000013	0x0007FFE4
33	0x00000013	0x0007FFD3	94	0x00000013	0x0007FFE5
34	0x00000011	0x0001FFE6	95	0x00000013	0x0007FFE6
35	0x00000012	0x0003FFD4	96	0x00000013	0x0007FFE7
36	0x00000010	0x0000FFF0	97	0x00000013	0x0007FFE8
37	0x00000011	0x0001FFE9	98	0x00000013	0x0007FFE9
38	0x00000012	0x0003FFD5	99	0x00000013	0x0007FFEA
39	0x00000011	0x0001FFE7	100	0x00000013	0x0007FFEB
40	0x00000010	0x0000FFF1	101	0x00000013	0x0007FFEC
41	0x00000010	0x0000FFEC	102	0x00000013	0x0007FFED
42	0x00000010	0x0000FFED	103	0x00000013	0x0007FFEE
43	0x00000010	0x0000FFEE	104	0x00000013	0x0007FFEF
44	0x0000000F	0x00007FF4	105	0x00000013	0x0007FFF0
45	0x0000000E	0x00003FF9	106	0x00000013	0x0007FFF1
46	0x0000000E	0x00003FF7	107	0x00000013	0x0007FFF2
47	0x0000000D	0x00001FFA	108	0x00000013	0x0007FFF3
48	0x0000000D	0x00001FF9	109	0x00000013	0x0007FFF4
49	0x0000000C	0x00000FFB	110	0x00000013	0x0007FFF5
50	0x0000000B	0x000007FC	111	0x00000013	0x0007FFF6
51	0x0000000A	0x000003FC	112	0x00000013	0x0007FFF7
52	0x00000009	0x000001FD	113	0x00000013	0x0007FFF8
53	0x00000008	0x000000FD	114	0x00000013	0x0007FFF9
54	0x00000007	0x0000007D	115	0x00000013	0x0007FFFA
55	0x00000006	0x0000003D	116	0x00000013	0x0007FFFB
56	0x00000005	0x0000001D	117	0x00000013	0x0007FFFC
57	0x00000004	0x0000000D	118	0x00000013	0x0007FFFD
58	0x00000003	0x00000005	119	0x00000013	0x0007FFFE
59	0x00000002	0x00000001	120	0x00000013	0x0007FFFF
60	0x00000002	0x00000000			

Table I.4: f_huffman_env_1_5dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x00000013	0x0007FFE7	61	0x00000003	0x00000004
1	0x00000013	0x0007FFE8	62	0x00000004	0x0000000D
2	0x00000014	0x000FFFD2	63	0x00000005	0x0000001D
3	0x00000014	0x000FFFD3	64	0x00000006	0x0000003D
4	0x00000014	0x000FFFD4	65	0x00000008	0x000000FA
5	0x00000014	0x000FFFD5	66	0x00000008	0x000000FC
6	0x00000014	0x000FFFD6	67	0x00000009	0x000001FB
7	0x00000014	0x000FFFD7	68	0x0000000A	0x000003FA
8	0x00000014	0x000FFFD8	69	0x0000000B	0x000007F8
9	0x00000013	0x0007FFDA	70	0x0000000B	0x000007FA
10	0x00000014	0x000FFFD9	71	0x0000000B	0x000007FB
11	0x00000014	0x000FFFDA	72	0x0000000C	0x00000FF9
12	0x00000014	0x000FFFDB	73	0x0000000C	0x00000FFB
13	0x00000014	0x000FFFDC	74	0x0000000D	0x00001FF8
14	0x00000013	0x0007FFDB	75	0x0000000D	0x00001FFB
15	0x00000014	0x000FFFDD	76	0x0000000E	0x00003FF8
16	0x00000013	0x0007FFDC	77	0x0000000E	0x00003FF9
17	0x00000013	0x0007FFDD	78	0x00000010	0x0000FFF1
18	0x00000014	0x000FFFDE	79	0x00000010	0x0000FFF2
19	0x00000012	0x0003FFE4	80	0x00000011	0x0001FFEA
20	0x00000014	0x000FFFDF	81	0x00000011	0x0001FFEB
21	0x00000014	0x000FFFE0	82	0x00000012	0x0003FFE1
22	0x00000014	0x000FFFE1	83	0x00000012	0x0003FFE2
23	0x00000013	0x0007FFDE	84	0x00000012	0x0003FFEA
24	0x00000014	0x000FFFE2	85	0x00000012	0x0003FFE3
25	0x00000014	0x000FFFE3	86	0x00000012	0x0003FFE6
26	0x00000014	0x000FFFE4	87	0x00000012	0x0003FFE7
27	0x00000013	0x0007FFDF	88	0x00000012	0x0003FFEB
28	0x00000014	0x000FFFE5	89	0x00000014	0x000FFFE6
29	0x00000013	0x0007FFE0	90	0x00000013	0x0007FFE2
30	0x00000012	0x0003FFE8	91	0x00000014	0x000FFFE7
31	0x00000013	0x0007FFE1	92	0x00000014	0x000FFFE8
32	0x00000012	0x0003FFE0	93	0x00000014	0x000FFFE9
33	0x00000012	0x0003FFE9	94	0x00000014	0x000FFFEA
34	0x00000011	0x0001FFEF	95	0x00000014	0x000FFFEB
35	0x00000012	0x0003FFE5	96	0x00000014	0x000FFFEC
36	0x00000011	0x0001FFEC	97	0x00000013	0x0007FFE3
37	0x00000011	0x0001FFED	98	0x00000014	0x000FFFED
38	0x00000011	0x0001FFEE	99	0x00000014	0x000FFFEE
39	0x00000010	0x0000FFF4	100	0x00000014	0x000FFFEF
40	0x00000010	0x0000FFF3	101	0x00000014	0x000FFFF0
41	0x00000010	0x0000FFF0	102	0x00000013	0x0007FFE4
42	0x0000000F	0x00007FF7	103	0x00000014	0x000FFFF1
43	0x0000000F	0x00007FF6	104	0x00000012	0x0003FFEC
44	0x0000000E	0x00003FFA	105	0x00000014	0x000FFFF2
45	0x0000000D	0x00001FFA	106	0x00000014	0x000FFFF3
46	0x0000000D	0x00001FF9	107	0x00000013	0x0007FFE5
47	0x0000000C	0x00000FFA	108	0x00000013	0x0007FFE6
48	0x0000000C	0x00000FF8	109	0x00000014	0x000FFFF4
49	0x0000000B	0x000007F9	110	0x00000014	0x000FFFF5
50	0x0000000A	0x000003FB	111	0x00000014	0x000FFFF6
51	0x00000009	0x000001FC	112	0x00000014	0x000FFFF7
52	0x00000009	0x000001FA	113	0x00000014	0x000FFFF8
53	0x00000008	0x000000FB	114	0x00000014	0x000FFFF9
54	0x00000007	0x0000007C	115	0x00000014	0x000FFFFA
55	0x00000006	0x0000003C	116	0x00000014	0x000FFFFB
56	0x00000005	0x0000001C	117	0x00000014	0x000FFFFC
57	0x00000004	0x0000000C	118	0x00000014	0x000FFFFD
58	0x00000003	0x00000005	119	0x00000014	0x000FFFFE
59	0x00000002	0x00000001	120	0x00000014	0x000FFFFF
60	0x00000002	0x00000000			

Table I.5: t_huffman_env_bal_1_5dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x00000010	0x0000FFE4	25	0x00000002	0x00000002
1	0x00000010	0x0000FFE5	26	0x00000004	0x0000000E
2	0x00000010	0x0000FFE6	27	0x00000006	0x0000003E
3	0x00000010	0x0000FFE7	28	0x00000008	0x000000FE
4	0x00000010	0x0000FFE8	29	0x0000000B	0x000007FD
5	0x00000010	0x0000FFE9	30	0x0000000C	0x00000FFD
6	0x00000010	0x0000FFEA	31	0x0000000F	0x00007FF0
7	0x00000010	0x0000FFEB	32	0x00000010	0x0000FFE3
8	0x00000010	0x0000FFEC	33	0x00000010	0x0000FFF5
9	0x00000010	0x0000FFED	34	0x00000010	0x0000FFF6
10	0x00000010	0x0000FFEE	35	0x00000010	0x0000FFF7
11	0x00000010	0x0000FFEF	36	0x00000010	0x0000FFF8
12	0x00000010	0x0000FFF0	37	0x00000010	0x0000FFF9
13	0x00000010	0x0000FFF1	38	0x00000010	0x0000FFFA
14	0x00000010	0x0000FFF2	39	0x00000011	0x0001FFF6
15	0x00000010	0x0000FFF3	40	0x00000011	0x0001FFF7
16	0x00000010	0x0000FFF4	41	0x00000011	0x0001FFF8
17	0x00000010	0x0000FFE2	42	0x00000011	0x0001FFF9
18	0x0000000C	0x00000FFC	43	0x00000011	0x0001FFFA
19	0x0000000B	0x000007FC	44	0x00000011	0x0001FFFB
20	0x00000009	0x000001FE	45	0x00000011	0x0001FFFC
21	0x00000007	0x0000007E	46	0x00000011	0x0001FFFD
22	0x00000005	0x0000001E	47	0x00000011	0x0001FFFE
23	0x00000003	0x00000006	48	0x00000011	0x0001FFFF
24	0x00000001	0x00000000			

Table I.6: f_huffman_env_bal_1_5dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x00000012	0x0003FFE2	25	0x00000003	0x00000006
1	0x00000012	0x0003FFE3	26	0x00000005	0x0000001E
2	0x00000012	0x0003FFE4	27	0x00000006	0x0000003E
3	0x00000012	0x0003FFE5	28	0x00000009	0x000001FE
4	0x00000012	0x0003FFE6	29	0x0000000B	0x000007FD
5	0x00000012	0x0003FFE7	30	0x0000000C	0x00000FFE
6	0x00000012	0x0003FFE8	31	0x0000000F	0x00007FFA
7	0x00000012	0x0003FFE9	32	0x00000010	0x0000FFF6
8	0x00000012	0x0003FFEA	33	0x00000012	0x0003FFF1
9	0x00000012	0x0003FFEB	34	0x00000012	0x0003FFF2
10	0x00000012	0x0003FFEC	35	0x00000012	0x0003FFF3
11	0x00000012	0x0003FFED	36	0x00000012	0x0003FFF4
12	0x00000012	0x0003FFEE	37	0x00000012	0x0003FFF5
13	0x00000012	0x0003FFEF	38	0x00000012	0x0003FFF6
14	0x00000012	0x0003FFF0	39	0x00000012	0x0003FFF7
15	0x00000010	0x0000FFF7	40	0x00000012	0x0003FFF8
16	0x00000011	0x0001FFF0	41	0x00000012	0x0003FFF9
17	0x0000000E	0x00003FFC	42	0x00000012	0x0003FFFA
18	0x0000000B	0x000007FE	43	0x00000012	0x0003FFFB
19	0x0000000B	0x000007FC	44	0x00000012	0x0003FFFC
20	0x00000008	0x000000FE	45	0x00000012	0x0003FFFD
21	0x00000007	0x0000007E	46	0x00000012	0x0003FFFE
22	0x00000004	0x0000000E	47	0x00000013	0x0007FFFE
23	0x00000002	0x00000002	48	0x00000013	0x0007FFFF
24	0x00000001	0x00000000			

Table I.7: t_huffman_env_3_0dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x00000012	0x0003FFED	32	0x00000003	0x00000006
1	0x00000012	0x0003FFEE	33	0x00000005	0x0000001E
2	0x00000013	0x0007FFDE	34	0x00000007	0x0000007E
3	0x00000013	0x0007FFDF	35	0x00000009	0x000001FE
4	0x00000013	0x0007FFE0	36	0x0000000B	0x000007FD
5	0x00000013	0x0007FFE1	37	0x0000000D	0x00001FFB
6	0x00000013	0x0007FFE2	38	0x0000000E	0x00003FF9
7	0x00000013	0x0007FFE3	39	0x0000000E	0x00003FFC
8	0x00000013	0x0007FFE4	40	0x0000000F	0x00007FFA
9	0x00000013	0x0007FFE5	41	0x00000010	0x0000FFF6
10	0x00000013	0x0007FFE6	42	0x00000011	0x0001FFF5
11	0x00000013	0x0007FFE7	43	0x00000012	0x0003FFEC
12	0x00000013	0x0007FFE8	44	0x00000013	0x0007FFED
13	0x00000013	0x0007FFE9	45	0x00000013	0x0007FFEE
14	0x00000013	0x0007FFEA	46	0x00000013	0x0007FFEF
15	0x00000013	0x0007FFEB	47	0x00000013	0x0007FFF0
16	0x00000013	0x0007FFEC	48	0x00000013	0x0007FFF1
17	0x00000011	0x0001FFF4	49	0x00000013	0x0007FFF2
18	0x00000010	0x0000FFF7	50	0x00000013	0x0007FFF3
19	0x00000010	0x0000FFF9	51	0x00000013	0x0007FFF4
20	0x00000010	0x0000FFF8	52	0x00000013	0x0007FFF5
21	0x0000000E	0x00003FFB	53	0x00000013	0x0007FFF6
22	0x0000000E	0x00003FFA	54	0x00000013	0x0007FFF7
23	0x0000000E	0x00003FF8	55	0x00000013	0x0007FFF8
24	0x0000000D	0x00001FFA	56	0x00000013	0x0007FFF9
25	0x0000000C	0x00000FFC	57	0x00000013	0x0007FFFA
26	0x0000000B	0x000007FC	58	0x00000013	0x0007FFFB
27	0x00000008	0x000000FE	59	0x00000013	0x0007FFFC
28	0x00000006	0x0000003E	60	0x00000013	0x0007FFFD
29	0x00000004	0x0000000E	61	0x00000013	0x0007FFFE
30	0x00000002	0x00000002	62	0x00000013	0x0007FFFF
31	0x00000001	0x00000000			

Table I.8: f_huffman_env_3_0dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x00000014	0x000FFFF0	32	0x00000003	0x00000006
1	0x00000014	0x000FFFF1	33	0x00000005	0x0000001E
2	0x00000014	0x000FFFF2	34	0x00000008	0x000000FC
3	0x00000014	0x000FFFF3	35	0x00000009	0x000001FC
4	0x00000014	0x000FFFF4	36	0x0000000A	0x000003FC
5	0x00000014	0x000FFFF5	37	0x0000000B	0x000007FC
6	0x00000014	0x000FFFF6	38	0x0000000C	0x00000FFC
7	0x00000012	0x0003FFF3	39	0x0000000D	0x00001FFC
8	0x00000013	0x0007FFF5	40	0x0000000E	0x00003FFA
9	0x00000013	0x0007FFEE	41	0x0000000F	0x00007FF9
10	0x00000013	0x0007FFEF	42	0x0000000F	0x00007FFA
11	0x00000013	0x0007FFF6	43	0x00000010	0x0000FFF8
12	0x00000012	0x0003FFF4	44	0x00000010	0x0000FFF9
13	0x00000012	0x0003FFF2	45	0x00000011	0x0001FFF6
14	0x00000014	0x000FFFF7	46	0x00000011	0x0001FFF7
15	0x00000013	0x0007FFF0	47	0x00000012	0x0003FFF5
16	0x00000011	0x0001FFF5	48	0x00000012	0x0003FFF6
17	0x00000012	0x0003FFF0	49	0x00000012	0x0003FFF1
18	0x00000011	0x0001FFF4	50	0x00000014	0x000FFFF8
19	0x00000010	0x0000FFF7	51	0x00000013	0x0007FFF1
20	0x00000010	0x0000FFF6	52	0x00000013	0x0007FFF2
21	0x0000000F	0x00007FF8	53	0x00000013	0x0007FFF3
22	0x0000000E	0x00003FFB	54	0x00000014	0x000FFFF9
23	0x0000000C	0x00000FFD	55	0x00000013	0x0007FFF7
24	0x0000000B	0x000007FD	56	0x00000013	0x0007FFF4
25	0x0000000A	0x000003FD	57	0x00000014	0x000FFFFA
26	0x00000009	0x000001FD	58	0x00000014	0x000FFFFB
27	0x00000008	0x000000FD	59	0x00000014	0x000FFFFC
28	0x00000006	0x0000003E	60	0x00000014	0x000FFFFD
29	0x00000004	0x0000000E	61	0x00000014	0x000FFFFE
30	0x00000002	0x00000002	62	0x00000014	0x000FFFFF
31	0x00000001	0x00000000			

Table I.9: t_huffman_env_bal_3_0dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x0000000D	0x00001FF2	13	0x00000002	0x00000002
1	0x0000000D	0x00001FF3	14	0x00000005	0x0000001E
2	0x0000000D	0x00001FF4	15	0x00000006	0x0000003E
3	0x0000000D	0x00001FF5	16	0x00000009	0x000001FE
4	0x0000000D	0x00001FF6	17	0x0000000D	0x00001FF9
5	0x0000000D	0x00001FF7	18	0x0000000D	0x00001FFA
6	0x0000000D	0x00001FF8	19	0x0000000D	0x00001FFB
7	0x0000000C	0x00000FF8	20	0x0000000D	0x00001FFC
8	0x00000008	0x000000FE	21	0x0000000D	0x00001FFD
9	0x00000007	0x0000007E	22	0x0000000D	0x00001FFE
10	0x00000004	0x0000000E	23	0x0000000E	0x00003FFE
11	0x00000003	0x00000006	24	0x0000000E	0x00003FFF
12	0x00000001	0x00000000			

Table I.10: f_huffman_env_bal_3_0dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x0000000D	0x00001FF7	13	0x00000003	0x00000006
1	0x0000000D	0x00001FF8	14	0x00000005	0x0000001E
2	0x0000000D	0x00001FF9	15	0x00000006	0x0000003E
3	0x0000000D	0x00001FFA	16	0x00000009	0x000001FE
4	0x0000000D	0x00001FFB	17	0x0000000C	0x00000FFA
5	0x0000000E	0x00003FF8	18	0x0000000D	0x00001FF6
6	0x0000000E	0x00003FF9	19	0x0000000E	0x00003FFA
7	0x0000000B	0x000007FC	20	0x0000000E	0x00003FFB
8	0x00000008	0x000000FE	21	0x0000000E	0x00003FFC
9	0x00000007	0x0000007E	22	0x0000000E	0x00003FFD
10	0x00000004	0x0000000E	23	0x0000000E	0x00003FFE
11	0x00000002	0x00000002	24	0x0000000E	0x00003FFF
12	0x00000001	0x00000000			

Table I.11: t_huffman_noise_3_0dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x0000000D	0x00001FCE	32	0x00000002	0x00000002
1	0x0000000D	0x00001FCF	33	0x00000005	0x0000001E
2	0x0000000D	0x00001FD0	34	0x00000008	0x000000FC
3	0x0000000D	0x00001FD1	35	0x0000000A	0x000003F8
4	0x0000000D	0x00001FD2	36	0x0000000D	0x00001FCC
5	0x0000000D	0x00001FD3	37	0x0000000D	0x00001FE8
6	0x0000000D	0x00001FD4	38	0x0000000D	0x00001FE9
7	0x0000000D	0x00001FD5	39	0x0000000D	0x00001FEA
8	0x0000000D	0x00001FD6	40	0x0000000D	0x00001FEB
9	0x0000000D	0x00001FD7	41	0x0000000D	0x00001FEC
10	0x0000000D	0x00001FD8	42	0x0000000D	0x00001FCD
11	0x0000000D	0x00001FD9	43	0x0000000D	0x00001FED
12	0x0000000D	0x00001FDA	44	0x0000000D	0x00001FEE
13	0x0000000D	0x00001FDB	45	0x0000000D	0x00001FEF
14	0x0000000D	0x00001FDC	46	0x0000000D	0x00001FF0
15	0x0000000D	0x00001FDD	47	0x0000000D	0x00001FF1
16	0x0000000D	0x00001FDE	48	0x0000000D	0x00001FF2
17	0x0000000D	0x00001FDF	49	0x0000000D	0x00001FF3
18	0x0000000D	0x00001FE0	50	0x0000000D	0x00001FF4
19	0x0000000D	0x00001FE1	51	0x0000000D	0x00001FF5
20	0x0000000D	0x00001FE2	52	0x0000000D	0x00001FF6
21	0x0000000D	0x00001FE3	53	0x0000000D	0x00001FF7
22	0x0000000D	0x00001FE4	54	0x0000000D	0x00001FF8
23	0x0000000D	0x00001FE5	55	0x0000000D	0x00001FF9
24	0x0000000D	0x00001FE6	56	0x0000000D	0x00001FFA
25	0x0000000D	0x00001FE7	57	0x0000000D	0x00001FFB
26	0x0000000B	0x000007F2	58	0x0000000D	0x00001FFC
27	0x00000008	0x000000FD	59	0x0000000D	0x00001FFD
28	0x00000006	0x0000003E	60	0x0000000D	0x00001FFE
29	0x00000004	0x0000000E	61	0x0000000E	0x00003FFE
30	0x00000003	0x00000006	62	0x0000000E	0x00003FFF
31	0x00000001	0x00000000			

Table I.12: t_huffman_noise_bal_3_0dB

index	length (hexadecimal)	codeword (hexadecimal)	index	length (hexadecimal)	codeword (hexadecimal)
0	0x00000008	0x000000EC	13	0x00000003	0x00000006
1	0x00000008	0x000000ED	14	0x00000006	0x0000003A
2	0x00000008	0x000000EE	15	0x00000008	0x000000F6
3	0x00000008	0x000000EF	16	0x00000008	0x000000F7
4	0x00000008	0x000000F0	17	0x00000008	0x000000F8
5	0x00000008	0x000000F1	18	0x00000008	0x000000F9
6	0x00000008	0x000000F2	19	0x00000008	0x000000FA
7	0x00000008	0x000000F3	20	0x00000008	0x000000FB
8	0x00000008	0x000000F4	21	0x00000008	0x000000FC
9	0x00000008	0x000000F5	22	0x00000008	0x000000FD
10	0x00000005	0x0000001C	23	0x00000008	0x000000FE
11	0x00000002	0x00000002	24	0x00000008	0x000000FF
12	0x00000001	0x00000000			

I.4 SBR Noise table

Table I.13: Noise table $V\left[\varphi_{Re,noise}(i),\varphi_{Im,noise}(i)\right]$

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
0	-0,99948153278296	-0,59483417516607	256	0,99570534804836	0,45844586038111
1	0,97113454393991	-0,67528515225647	257	-0,63431466947340	0,21079116459234
2	0,14130051758487	-0,95090983575689	258	-0,07706847005931	-0,89581437101329
3	-0,47005496701697	-0,37340549728647	259	0,98590090577724	0,88241721133981
4	0,80705063769351	0,29653668284408	260	0,80099335254678	-0,36851896710853
5	-0,38981478896926	0,89572605717087	261	0,78368131392666	0,45506999802597
6	-0,01053049862020	-0,66959058036166	262	0,08707806671691	0,80938994918745
7	-0,91266367957293	-0,11522938140034	263	-0,86811883080712	0,39347308654705
8	0,54840422910309	0,75221367176302	264	-0,39466529740375	-0,66809432114456
9	0,40009252867955	-0,98929400334421	265	0,97875325649683	-0,72467840967746
10	-0,99867974711855	-0,88147068645358	266	-0,95038560288864	0,89563219587625
11	-0,95531076805040	0,90908757154593	267	0,17005239424212	0,54683053962658
12	-0,45725933317144	-0,56716323646760	268	-0,76910792026848	-0,96226617549298
13	-0,72929675029275	-0,98008272727324	269	0,99743281016846	0,42697157037567
14	0,75622801399036	0,20950329995549	270	0,95437383549973	0,97002324109952
15	0,07069442601050	-0,78247898470706	271	0,99578905365569	-0,54106826257356
16	0,74496252926055	-0,91169004445807	272	0,28058259829990	-0,85361420634036
17	-0,96440182703856	-0,94739918296622	273	0,85256524470573	-0,64567607735589
18	0,30424629369539	-0,49438267012479	274	-0,50608540105128	-0,65846015480300
19	0,66565033746925	0,64652935542491	275	-0,97210735183243	-0,23095213067791
20	0,91697008020594	0,17514097332009	276	0,95424048234441	-0,99240147091219
21	-0,70774918760427	0,52548653416543	277	-0,96926570524023	0,73775654896574
22	-0,70051415345560	-0,45340028808763	278	0,30872163214726	0,41514960556126
23	-0,99496513054797	-0,90071908066973	279	-0,24523839572639	0,63206633394807
24	0,98164490790123	-0,77463155528697	280	-0,33813265086024	-0,38661779441897
25	-0,54671580548181	-0,02570928536004	281	-0,05826828420146	-0,06940774188029
26	-0,01689629065389	0,00287506445732	282	-0,22898461455054	0,97054853316316
27	-0,86110349531986	0,42548583726477	283	-0,18509915019881	0,47565762892084
28	-0,98892980586032	-0,87881132267556	284	-0,10488238045009	-0,87769947402394
29	0,51756627678691	0,66926784710139	285	-0,71886586182037	0,78030982480538
30	-0,99635026409640	-0,58107730574765	286	0,99793873738654	0,90041310491497
31	-0,99969370862163	0,98369989360250	287	0,57563307626120	-0,91034337352097
32	0,55266258627194	0,59449057465591	288	0,28909646383717	0,96307783970534
33	0,34581177741673	0,94879421061866	289	0,42188998312520	0,48148651230437
34	0,62664209577999	-0,74402970906471	290	0,93335049681047	-0,43537023883588
35	-0,77149701404973	-0,33883658042801	291	-0,97087374418267	0,86636445711364
36	-0,91592244254432	0,03687901376713	292	0,36722871286923	0,65291654172961
37	-0,76285492357887	-0,91371867919124	293	-0,81093025665696	0,08778370229363

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
38	0,79788337195331	-0,93180971199849	294	-0,26240603062237	-0,92774095379098
39	0,54473080610200	-0,11919206037186	295	0,83996497984604	0,55839849139647
40	-0,85639281671058	0,42429854760451	296	-0,99909615720225	-0,96024605713970
41	-0,92882402971423	0,27871809078609	297	0,74649464155061	0,12144893606462
42	-0,11708371046774	-0,99800843444966	298	-0,74774595569805	-0,26898062008959
43	0,21356749817493	-0,90716295627033	299	0,95781667469567	-0,79047927052628
44	-0,76191692573909	0,99768118356265	300	0,95472308713099	-0,08588776019550
45	0,98111043100884	-0,95854459734407	301	0,48708332746299	0,99999041579432
46	-0,85913269895572	0,95766566168880	302	0,46332038247497	0,10964126185063
47	-0,93307242253692	0,49431757696466	303	-0,76497004940162	0,89210929242238
48	0,30485754879632	-0,70540034357529	304	0,57397389364339	0,35289703373760
49	0,85289650925190	0,46766131791044	305	0,75374316974495	0,96705214651335
50	0,91328082618125	-0,99839597361769	306	-0,59174397685714	-0,89405370422752
51	-0,05890199924154	0,70741827819497	307	0,75087906691890	-0,29612672982396
52	0,28398686150148	0,34633555702188	308	-0,98607857336230	0,25034911730023
53	0,95258164539612	-0,54893416026939	309	-0,40761056640505	-0,90045573444695
54	-0,78566324168507	-0,75568541079691	310	0,66929266740477	0,98629493401748
55	-0,95789495447877	-0,20423194696966	311	-0,97463695257310	-0,00190223301301
56	0,82411158711197	0,96654618432562	312	0,90145509409859	0,99781390365446
57	-0,65185446735885	-0,88734990773289	313	-0,87259289048043	0,99233587353666
58	-0,93643603134666	0,99870790442385	314	-0,91529461447692	-0,15698707534206
59	0,91427159529618	-0,98290505544444	315	-0,03305738840705	-0,37205262859764
60	-0,70395684036886	0,58796798221039	316	0,07223051368337	-0,88805001733626
61	0,00563771969365	0,61768196727244	317	0,99498012188353	0,97094358113387
62	0,89065051931895	0,52783352697585	318	-0,74904939500519	0,99985483641521
63	-0,68683707712762	0,80806944710339	319	0,04585228574211	0,99812337444082
64	0,72165342518718	-0,69259857349564	320	-0,89054954257993	-0,31791913188064
65	-0,62928247730667	0,13627037407335	321	-0,83782144651251	0,97637632547466
66	0,29938434065514	-0,46051329682246	322	0,33454804933804	-0,86231516800408
67	-0,91781958879280	-0,74012716684186	323	-0,99707579362824	0,93237990079441
68	0,99298717043688	0,40816610075661	324	-0,22827527843994	0,18874759397997
69	0,82368298622748	-0,74036047190173	325	0,67248046289143	-0,03646211390569
70	-0,98512833386833	-0,99972330709594	326	-0,05146538187944	-0,92599700120679
71	-0,95915368242257	-0,99237800466040	327	0,99947295749905	0,93625229707912
72	-0,21411126572790	-0,93424819052545	328	0,66951124390363	0,98905825623893
73	-0,68821476106884	-0,26892306315457	329	-0,99602956559179	-0,44654715757688
74	0,91851997982317	0,09358228901785	330	0,82104905483590	0,99540741724928
75	-0,96062769559127	0,36099095133739	331	0,99186510988782	0,72023001312947
76	0,51646184922287	-0,71373332873917	332	-0,65284592392918	0,52186723253637
77	0,61130721139669	0,46950141175917	333	0,93885443798188	-0,74895312615259
78	0,47336129371299	-0,27333178296162	334	0,96735248738388	0,90891816978629
79	0,90998308703519	0,96715662938132	335	-0,22225968841114	0,57124029781228
80	0,44844799194357	0,99211574628306	336	-0,44132783753414	-0,92688840659280
81	0,66614891079092	0,96590176169121	337	-0,85694974219574	0,88844532719844
82	0,74922239129237	-0,89879858826087	338	0,91783042091762	-0,46356892383970
83	-0,99571588506485	0,52785521494349	339	0,72556974415690	-0,99899555770747
84	0,97401082477563	-0,16855870075190	340	-0,99711581834508	0,58211560180426
85	0,72683747733879	-0,48060774432251	341	0,77638976371966	0,94321834873819
86	0,95432193457128	0,68849603408441	342	0,07717324253925	0,58638399856595
87	-0,72962208425191	-0,76608443420917	343	-0,56049829194163	0,82522301569036
88	-0,85359479233537	0,88738125901579	344	0,98398893639988	0,39467440420569
89	-0,81412430338535	-0,97480768049637	345	0,47546946844938	0,68613044836811
90	-0,87930772356786	0,74748307690436	346	0,65675089314631	0,18331637134880
91	-0,71573331064977	-0,98570608178923	347	0,03273375457980	-0,74933109564108
92	0,83524300028228	0,83702537075163	348	-0,38684144784738	0,51337349030406
93	-0,48086065601423	-0,98848504923531	349	-0,97346267944545	-0,96549364384098
94	0,97139128574778	0,80093621198236	350	-0,53282156061942	-0,91423265091354
95	0,51992825347895	0,80247631400510	351	0,99817310731176	0,61133572482148
96	-0,00848591195325	-0,76670128000486	352	-0,50254500772635	-0,88829338134294
97	-0,70294374303036	0,55359910445577	353	0,01995873238855	0,85223515096765
98	-0,95894428168140	-0,43265504344783	354	0,99930381973804	0,94578896296649
99	0,97079252950321	0,09325857238682	355	0,82907767600783	-0,06323442598128
100	-0,92404293670797	0,85507704027855	356	-0,58660709669728	0,96840773806582
101	-0,69506469500450	0,98633412625459	357	-0,17573736667267	-0,48166920859485

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
102	0,26559203620024	0,73314307966524	358	0,83434292401346	-0,13023450646997
103	0,28038443336943	0,14537913654427	359	0,05946491307025	0,20511047074866
104	-0,74138124825523	0,99310339807762	360	0,81505484574602	-0,94685947861369
105	-0,01752795995444	-0,82616635284178	361	-0,44976380954860	0,40894572671545
106	-0,55126773094930	-0,98898543862153	362	-0,89746474625671	0,99846578838537
107	0,97960898850996	-0,94021446752851	363	0,39677256130792	-0,74854668609359
108	-0,99196309146936	0,67019017358456	364	-0,07588948563079	0,74096214084170
109	-0,67684928085260	0,12631491649378	365	0,76343198951445	0,41746629422634
110	0,09140039465500	-0,20537731453108	366	-0,74490104699626	0,94725911744610
111	-0,71658965751996	-0,97788200391224	367	0,64880119792759	0,41336660830571
112	0,81014640078925	0,53722648362443	368	0,62319537462542	-0,93098313552599
113	0,40616991671205	-0,26469008598449	369	0,42215817594807	-0,07712787385208
114	-0,67680188682972	0,94502052337695	370	0,02704554141885	-0,05417518053666
115	0,86849774348749	-0,18333598647899	371	0,80001773566818	0,91542195141039
116	-0,99500381284851	-0,02634122068550	372	-0,79351832348816	-0,36208897989136
117	0,84329189340667	0,10406957462213	373	0,63872359151636	0,08128252493444
118	-0,09215968531446	0,69540012101253	374	0,52890520960295	0,60048872455592
119	0,99956173327206	-0,12358542001404	375	0,74238552914587	0,04491915291044
120	-0,79732779473535	-0,91582524736159	376	0,99096131449250	-0,19451182854402
121	0,96349973642406	0,96640458041000	377	-0,80412329643109	-0,88513818199457
122	-0,79942778496547	0,64323902822857	378	-0,64612616129736	0,72198674804544
123	-0,11566039853896	0,28587846253726	379	0,11657770663191	-0,83662833815041
124	-0,39922954514662	0,94129601616966	380	-0,95053182488101	-0,96939905138082
125	0,99089197565987	-0,92062625581587	381	-0,62228872928622	0,82767262846661
126	0,28631285179909	-0,91035047143603	382	0,03004475787316	-0,99738896333384
127	-0,83302725605608	-0,67330410892084	383	-0,97987214341034	0,36526129686425
128	0,95404443402072	0,49162765398743	384	-0,99986980746200	-0,36021610299715
129	-0,06449863579434	0,03250560813135	385	0,89110648599879	-0,97894250343044
130	-0,99575054486311	0,42389784469507	386	0,10407960510582	0,77357793811619
131	-0,65501142790847	0,82546114655624	387	0,95964737821728	-0,35435818285502
132	-0,81254441908887	-0,51627234660629	388	0,50843233159162	0,96107691266205
133	-0,99646369485481	0,84490533520752	389	0,17006334670615	-0,76854025314829
134	0,00287840603348	0,64768261158166	390	0,25872675063360	0,99893303933816
135	0,70176989408455	-0,20453028573322	391	-0,01115998681937	0,98496019742444
136	0,96361882270190	0,40706967140989	392	-0,79598702973261	0,97138411318894
137	-0,68883758192426	0,91338958840772	393	-0,99264708948101	-0,99542822402536
138	-0,34875585502238	0,71472290693300	394	-0,99829663752818	0,01877138824311
139	0,91980081243087	0,66507455644919	395	-0,70801016548184	0,33680685948117
140	-0,99009048343881	0,85868021604848	396	-0,70467057786826	0,93272777501857
141	0,68865791458395	0,55660316809678	397	0,99846021905254	-0,98725746254433
142	-0,99484402129368	-0,20052559254934	398	-0,63364968534650	-0,16473594423746
143	0,94214511408023	-0,99696425367461	399	-0,16258217500792	-0,95939125400802
144	-0,67414626793544	0,49548221180078	400	-0,43645594360633	-0,94805030113284
145	-0,47339353684664	-0,85904328834047	401	-0,99848471702976	0,96245166923809
146	0,14323651387360	-0,94145598222488	402	-0,16796458968998	-0,98987511890470
147	-0,29268293575672	0,05759224927952	403	-0,87979225745213	-0,71725725041680
148	0,43793861458754	-0,78904969892724	404	0,44183099021786	-0,93568974498761
149	-0,36345126374441	0,64874435357162	405	0,93310180125532	-0,99913308068246
150	-0,08750604656825	0,97686944362527	406	-0,93941931782002	-0,56409379640356
151	-0,96495267812511	-0,53960305946511	407	-0,88590003188677	0,47624600491382
152	0,55526940659947	0,78891523734774	408	0,99971463703691	-0,83889954253462
153	0,73538215752630	0,96452072373404	409	-0,75376385639978	0,00814643438625
154	-0,30889773919437	-0,80664389776860	410	0,93887685615875	-0,11284528204636
155	0,03574995626194	-0,97325616900959	411	0,85126435782309	0,52349251543547
156	0,98720684660488	0,48409133691962	412	0,39701421446381	0,81779634174316
157	-0,81689296271203	-0,90827703628298	413	-0,37024464187437	-0,87071656222959
158	0,67866860118215	0,81284503870856	414	-0,36024828242896	0,34655735648287
159	-0,15808569732583	0,85279555024382	415	-0,93388812549209	-0,84476541096429
160	0,80723395114371	-0,24717418514605	416	-0,65298804552119	-0,18439575450921
161	0,47788757329038	-0,46333147839295	417	0,11960319006843	0,99899346780168
162	0,96367554763201	0,38486749303242	418	0,94292565553160	0,83163906518293
163	-0,99143875716818	-0,24945277239809	419	0,75081145286948	-0,35533223142265
164	0,83081876925833	-0,94780851414763	420	0,56721979748394	-0,24076836414499
165	-0,58753191905341	0,01290772389163	421	0,46857766746029	-0,30140233457198

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
166	0,95538108220960	-0,85557052096538	422	0,97312313923635	-0,99548191630031
167	-0,96490920476211	-0,64020970923102	423	-0,38299976567017	0,98516909715427
168	-0,97327101028521	0,12378128133110	424	0,41025800019463	0,02116736935734
169	0,91400366022124	0,57972471346930	425	0,09638062008048	0,04411984381457
170	-0,99925837363824	0,71084847864067	426	-0,85283249275397	0,91475563922421
171	-0,86875903507313	-0,20291699203564	427	0,88866808958124	-0,99735267083226
172	-0,26240034795124	-0,68264554369108	428	-0,48202429536989	-0,96805608884164
173	-0,24664412953388	-0,87642273115183	429	0,27572582416567	0,58634753335832
174	0,02416275806869	0,27192914288905	430	-0,65889129659168	0,58835634138583
175	0,82068619590515	-0,85087787994476	431	0,98838086953732	0,99994349600236
176	0,88547373760759	-0,89636802901469	432	-0,20651349620689	0,54593044066355
177	-0,18173078152226	-0,26152145156800	433	-0,62126416356920	-0,59893681700392
178	0,09355476558534	0,54845123045604	434	0,20320105410437	-0,86879180355289
179	-0,54668414224090	0,95980774020221	435	-0,97790548600584	0,96290806999242
180	0,37050990604091	-0,59910140383171	436	0,11112534735126	0,21484763313301
181	-0,70373594262891	0,91227665827081	437	-0,41368337314182	0,28216837680365
182	-0,34600785879594	-0,99441426144200	438	0,24133038992960	0,51294362630238
183	-0,68774481731008	-0,30238837956299	439	-0,66393410674885	-0,08249679629081
184	-0,26843291251234	0,83115668004362	440	-0,53697829178752	-0,97649903936228
185	0,49072334613242	-0,45359708737775	441	-0,97224737889348	0,22081333579837
186	0,38975993093975	0,95515358099121	442	0,87392477144549	-0,12796173740361
187	-0,97757125224150	0,05305894580606	443	0,19050361015753	0,01602615387195
188	-0,17325552859616	-0,92770672250494	444	-0,46353441212724	-0,95249041539006
189	0,99948035025744	0,58285545563426	445	-0,07064096339021	-0,94479803205886
190	-0,64946246527458	0,68645507104960	446	-0,92444085484466	-0,10457590187436
191	-0,12016920576437	-0,57147322153312	447	-0,83822593578728	-0,01695043208885
192	-0,58947456517751	-0,34847132454388	448	0,75214681811150	-0,99955681042665
193	-0,41815140454465	0,16276422358861	449	-0,42102998829339	0,99720941999394
194	0,99885650204884	0,11136095490444	450	-0,72094786237696	-0,35008961934255
195	-0,56649614128386	-0,90494866361587	451	0,78843311019251	0,52851398958271
196	0,94138021032330	0,35281916733018	452	0,97394027897442	-0,26695944086561
197	-0,75725076534641	0,53650549640587	453	0,99206463477946	-0,57010120849429
198	0,20541973692630	-0,94435144369918	454	0,76789609461795	-0,76519356730966
199	0,99980371023351	0,79835913565599	455	-0,82002421836409	-0,73530179553767
200	0,29078277605775	0,35393777921520	456	0,81924990025724	0,99698425250579
201	-0,62858772103030	0,38765693387102	457	-0,26719850873357	0,68903369776193
202	0,43440904467688	-0,98546330463232	458	-0,43311260380975	0,85321815947490
203	-0,98298583762390	0,21021524625209	459	0,99194979673836	0,91876249766422
204	0,19513029146934	-0,94239832251867	460	-0,80692001248487	-0,32627540663214
205	-0,95476662400101	0,98364554179143	461	0,43080003649976	-0,21919095636638
206	0,93379635304810	-0,70881994583682	462	0,67709491937357	-0,95478075822906
207	-0,85235410573336	-0,08342347966410	463	0,56151770568316	-0,70693811747778
208	-0,86425093011245	-0,45795025029466	464	0,10831862810749	-0,08628837174592
209	0,38879779059045	0,97274429344593	465	0,91229417540436	-0,65987351408410
210	0,92045124735495	-0,62433652524220	466	-0,48972893932274	0,56289246362686
211	0,89162532251878	0,54950955570563	467	-0,89033658689697	-0,71656563987082
212	-0,36834336949252	0,96458298020975	468	0,65269447475094	0,65916004833932
213	0,93891760988045	-0,89968353740388	469	0,67439478141121	-0,81684380846796
214	0,99267657565094	-0,03757034316958	470	-0,47770832416973	-0,16789556203025
215	-0,94063471614176	0,41332338538963	471	-0,99715979260878	-0,93565784007648
216	0,99740224117019	-0,16830494996370	472	-0,90889593602546	0,62034397054380
217	-0,35899413170555	-0,46633226649613	473	-0,06618622548177	-0,23812217221359
218	0,05237237274947	-0,25640361602661	474	0,99430266919728	0,18812555317553
219	0,36703583957424	-0,38653265641875	475	0,97686402381843	-0,28664534366620
220	0,91653180367913	-0,30587628726597	476	0,94813650221268	-0,97506640027128
221	0,69000803499316	0,90952171386132	477	-0,95434497492853	-0,79607978501983
222	-0,38658751133527	0,99501571208985	478	-0,49104783137150	0,32895214359663
223	-0,29250814029851	0,37444994344615	479	0,99881175120751	0,88993983831354
224	-0,60182204677608	0,86779651036123	480	0,50449166760303	-0,85995072408434
225	-0,97418588163217	0,96468523666475	481	0,47162891065108	-0,18680204049569
226	0,88461574003963	0,57508405276414	482	-0,62081581361840	0,75000676218956
227	0,05198933055162	0,21269661669964	483	-0,43867015250812	0,99998069244322
228	-0,53499621979720	0,97241553731237	484	0,98630563232075	-0,53578899600662
229	-0,49429560226497	0,98183865291903	485	-0,61510362277374	-0,89515019899997

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
230	-0,98935142339139	-0,40249159006933	486	-0,03841517601843	-0,69888815681179
231	-0,98081380091130	-0,72856895534041	487	-0,30102157304644	-0,07667808922205
232	-0,27338148835532	0,99950922447209	488	0,41881284182683	0,02188098922282
233	0,06310802338302	-0,54539587529618	489	-0,86135454941237	0,98947480909359
234	-0,20461677199539	-0,14209977628489	490	0,67226861393788	-0,13494389011014
235	0,66223843141647	0,72528579940326	491	-0,70737398842068	-0,76547349325992
236	-0,84764345483665	0,02372316801261	492	0,94044946687963	0,09026201157416
237	-0,89039863483811	0,88866581484602	493	-0,82386352534327	0,08924768823676
238	0,95903308477986	0,76744927173873	494	-0,32070666698656	0,50143421908753
239	0,73504123909879	-0,03747203173192	495	0,57593163224487	-0,98966422921509
240	-0,31744434966056	-0,36834111883652	496	-0,36326018419965	0,07440243123228
241	-0,34110827591623	0,40211222807691	497	0,99979044674350	-0,14130287347405
242	0,47803883714199	-0,39423219786288	498	-0,92366023326932	-0,97979298068180
243	0,98299195879514	0,01989791390047	499	-0,44607178518598	-0,54233252016394
244	-0,30963073129751	-0,18076720599336	500	0,44226800932956	0,71326756742752
245	0,99992588229018	-0,26281872094289	501	0,03671907158312	0,63606389366675
246	-0,93149731080767	-0,98313162570490	502	0,52175424682195	-0,85396826735705
247	0,99923472302773	-0,80142993767554	503	-0,94701139690956	-0,01826348194255
248	-0,26024169633417	-0,75999759855752	504	-0,98759606946049	0,82288714303073
249	-0,35712514743563	0,19298963768574	505	0,87434794743625	0,89399495655433
250	-0,99899084509530	0,74645156992493	506	-0,93412041758744	0,41374052024363
251	0,86557171579452	0,55593866696299	507	0,96063943315511	0,93116709541280
252	0,33408042438752	0,86185953874709	508	0,97534253457837	0,86150930812689
253	0,99010736374716	0,04602397576623	509	0,99642466504163	0,70190043427512
254	-0,66694269691195	-0,91643611810148	510	-0,94705089665984	-0,29580042814306
255	0,64016792079480	0,15649530836856	511	0,91599807087376	-0,98147830385781

I.5 SBR low complexity stereo tables

Table I.14: LC-stereo LC_SMOOTH filter coefficients, Fs = 48 kHz

n	a_n	b_n
0	-0,98751192990731	0,00624403504634

Table I.15: LC-stereo LP filter coefficients, Fs = 48 kHz

n	a_n	b_n
0	-0,94896459579467773	0,025517702102661133

Table I.16: LC-stereo slope filter coefficients, Fs = 48 kHz

n	a_n	b_n
0	-0,61126708984375	0,34472656250000
1	-0,06530761718750	0

Annex J (informative): Numbers of input bits

Table J.1: Number of input bits L per multiplex frame for EEP SM Mode A

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, $R_{all} = 0,5$	3 757	4 248	7 878	8 857	16 374	18 336
64 QAM, $R_{all} = 0,6$	4 509	5 096	9 450	10 628	19 646	21 998
64 QAM, $R_{all} = 0,71$	5 322	6 018	11 157	12 547	23 193	25 976
64 QAM, $R_{all} = 0,78$	5 898	6 664	12 364	13 908	25 704	28 788
16 QAM, $R_{all} = 0,5$	2 505	2 832	5 250	5 904	10 914	12 222
16 QAM, $R_{all} = 0,62$	3 131	3 540	6 565	7 381	13 645	15 280

Table J.2: Number of input bits L per multiplex frame for EEP SM Mode B

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, $R_{all} = 0,5$	2 880	3 312	6 133	6 991	12 727	14 304
64 QAM, $R_{all} = 0,6$	3 456	3 972	7 361	8 390	15 272	17 162
64 QAM, $R_{all} = 0,71$	4 080	4 692	8 688	9 900	18 026	20 264
64 QAM, $R_{all} = 0,78$	4 520	5 196	9 630	10 980	19 980	22 456
16 QAM, $R_{all} = 0,5$	1 920	2 208	4 089	4 662	8 484	9 534
16 QAM, $R_{all} = 0,62$	2 400	2 760	5 111	5 826	10 606	11 920

Table J.3: Number of input bits L per multiplex frame for EEP SM Mode C

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, $R_{all} = 0,5$	Not used			5 514	Not used	11 581
64 QAM, $R_{all} = 0,6$				6 615		13 898
64 QAM, $R_{all} = 0,71$				7 808		16 406
64 QAM, $R_{all} = 0,78$				8 654		18 188
16 QAM, $R_{all} = 0,5$				3 675		7 722
16 QAM, $R_{all} = 0,62$				4 595		9 651

Table J.4: Number of input bits L per multiplex frame for EEP SM Mode D

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, $R_{all} = 0,5$	Not used			3 660	Not used	7 800
64 QAM, $R_{all} = 0,6$				4 391		9 359
64 QAM, $R_{all} = 0,71$				5 185		11 050
64 QAM, $R_{all} = 0,78$				5 746		12 242
16 QAM, $R_{all} = 0,5$				2 439		5 199
16 QAM, $R_{all} = 0,62$				3 050		6 500

Table J.5: Number of input bits L per hierarchical frame for EEP HMsym VSPP Mode A

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used		2 626	2 953	5 458	6 112
64 QAM, R_0 = 0,57			3 000	3 372	6 236	6 984
64 QAM, R_0 = 0,6			3 150	3 543	6 549	7 332
64 QAM, R_0 = 0,66			3 500	3 936	7 276	8 148

Table J.6: Number of input bits L per hierarchical frame for EEP HMsym VSPP Mode B

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used		2 045	2 331	4 243	4 768
64 QAM, R_0 = 0,57			2 336	2 664	4 848	5 448
64 QAM, R_0 = 0,6			2 454	2 796	5 091	5 721
64 QAM, R_0 = 0,66			2 726	3 108	5 656	6 356

Table J.7: Number of input bits L per hierarchical frame for EEP HMsym VSPP Mode C

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used			1 838	Not used	3 861
64 QAM, R_0 = 0,57				2 100		4 412
64 QAM, R_0 = 0,6				2 205		4 632
64 QAM, R_0 = 0,66				2 450		5 148

Table J.8: Number of input bits L per hierarchical frame for EEP HMsym VSPP Mode D

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used			1 220	Not used	2 600
64 QAM, R_0 = 0,57				1 392		2 968
64 QAM, R_0 = 0,6				1 464		3 120
64 QAM, R_0 = 0,66				1 626		3 466

Table J.9: Number of input bits L per multiplex frame for EEP HMsym SPP Mode A

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,45	Not used		4 725	5 313	9 822	10 998
64 QAM, R_{all} = 0,55			5 724	6 432	11 904	13 332
64 QAM, R_{all} = 0,72			7 592	8 538	15 784	17 680
64 QAM, R_{all} = 0,78			8 164	9 184	16 972	19 012

Table J.10: Number of input bits L per multiplex frame for EEP HMsym SPP Mode B

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,45	Not used		3 681	4 194	7 635	8 580
64 QAM, R_{all} = 0,55			4 452	5 076	9 252	10 392
64 QAM, R_{all} = 0,72			5 913	6 738	12 268	13 792
64 QAM, R_{all} = 0,78			6 358	7 252	13 192	14 828

Table J.11: Number of input bits L per multiplex frame for EEP HMsym SPP Mode C

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,45	Not used			3 306	Not used	6 948
64 QAM, R_{all} = 0,55				4 008		8 424
64 QAM, R_{all} = 0,72				5 313		11 167
64 QAM, R_{all} = 0,78				5 714		12 012

Table J.12: Number of input bits L per multiplex frame for EEP HMsym SPP Mode D

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,45	Not used			2 196	Not used	4 680
64 QAM, R_{all} = 0,55				2 652		5 664
64 QAM, R_{all} = 0,72				3 527		7 518
64 QAM, R_{all} = 0,78				3 794		8 082

Table J.13: Number of input bits L per hierarchical frame for EEP HMmix VSPP Mode A

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used		1 310	1 473	2 726	3 053
64 QAM, R_0 = 0,57			1 496	1 684	3 112	3 488
64 QAM, R_0 = 0,6			1 572	1 767	3 270	3 663
64 QAM, R_0 = 0,66			1 746	1 964	3 634	4 070

Table J.14: Number of input bits L per hierarchical frame for EEP HMmix VSPP Mode B

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used		1 019	1 162	2 118	2 381
64 QAM, R_0 = 0,57			1 164	1 328	2 420	2 720
64 QAM, R_0 = 0,6			1 221	1 395	2 541	2 856
64 QAM, R_0 = 0,66			1 358	1 550	2 824	3 174

Table J.15: Number of input bits L per hierarchical frame for EEP HMmix VSPP Mode C

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used			916	Not used	1 927
64 QAM, R_0 = 0,57				1 044		2 200
64 QAM, R_0 = 0,6				1 098		2 313
64 QAM, R_0 = 0,66				1 220		2 570

Table J.16: Number of input bits L per hierarchical frame for EEP HMmix VSPP Mode D

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_0 = 0,5	Not used			607	Not used	1 297
64 QAM, R_0 = 0,57				692		1 480
64 QAM, R_0 = 0,6				726		1 554
64 QAM, R_0 = 0,66				808		1 728

Table J.17: Number of input bits L per multiplex frame for EEP HMmix SPP Mode A

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,48	Not used		6 288	7 066	13 083	14 650
64 QAM, R_{all} = 0,58			7 571	8 506	15 751	17 649
64 QAM, R_{all} = 0,71			9 349	10 517	19 461	21 801
64 QAM, R_{all} = 0,78			10 244	11 516	21 308	23 872

Table J.18: Number of input bits L per multiplex frame for EEP HMmix SPP Mode B

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,48	Not used		4 885	5 577	10 164	11 425
64 QAM, R_{all} = 0,58			5 885	6 717	12 244	13 753
64 QAM, R_{all} = 0,71			7 266	8 293	15 121	17 001
64 QAM, R_{all} = 0,78			7 960	9 088	16 556	18 620

Table J.19: Number of input bits L per multiplex frame for EEP HMmix SPP Mode C

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,48	Not used			4 395	Not used	9 247
64 QAM, R_{all} = 0,58				5 286		11 139
64 QAM, R_{all} = 0,71				6 540		13 750
64 QAM, R_{all} = 0,78				7 152		15 072

Table J.20: Number of input bits L per multiplex frame for EEP HMmix SPP Mode D

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
64 QAM, R_{all} = 0,48	Not used			2 908	Not used	6 220
64 QAM, R_{all} = 0,58				3 500		7 484
64 QAM, R_{all} = 0,71				4 322		9 257
64 QAM, R_{all} = 0,78				4 728		10 136

Table J.21: Number of input bits L per SDC block for Mode A

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
16 QAM, R_{all} = 0,5	321	366	705	798	1 494	1 680
4 QAM, R_{all} = 0,5	161	184	353	399	748	840

Table J.22: Number of input bits L per SDC block for Mode B

Parameters	Spectrum occupancy					
	0	1	2	3	4	5
16 QAM, R_{all} = 0,5	246	288	552	630	1 164	1 311
4 QAM, R_{all} = 0,5	124	144	276	316	582	656

Table J.23: Number of input bits L per SDC block for Mode C

Parameters	Spectrum occupancy					
	0	**1**	**2**	**3**	**4**	**5**
16 QAM, R_{all} = 0,5	Not used			564	Not used	1 200
4 QAM, R_{all} = 0,5				282		601

Table J.24: Number of input bits L per SDC block for Mode D

Parameters	Spectrum occupancy					
	0	**1**	**2**	**3**	**4**	**5**
16 QAM, R_{all} = 0,5	Not used			291	Not used	651
4 QAM, R_{all} = 0,5				146		326

Annex K (informative): Simulcast transmission

The DRM signal is designed to work in the same broadcast bands as AM signals. Simulcast transmission of services using DRM and AM can be performed by the juxtaposition of the analogue AM signal (DSB or VSB or SSB) and a DRM digital signal.

The following figures illustrate some solutions for transmitting the AM and DRM signals from a single transmitter. They can equally be produced by two separate transmitters.

Figure K.1 gives some possibilities for the case where the DRM reference frequency, f_R, is one channel or two channels (i.e. ±9 kHz, ±10 kHz, -18 kHz or -20 kHz) from the AM carrier frequency, f_C, and figure K.2 gives some possibilities for the case where the DRM reference frequency, f_R, is nominally half a channel from the AM carrier frequency, f_C. Due to the requirement to position the DRM reference frequency on an integer multiple of 1 kHz, the DRM reference frequency and the AM carrier frequency shall be either 4 kHz or 5 kHz apart.

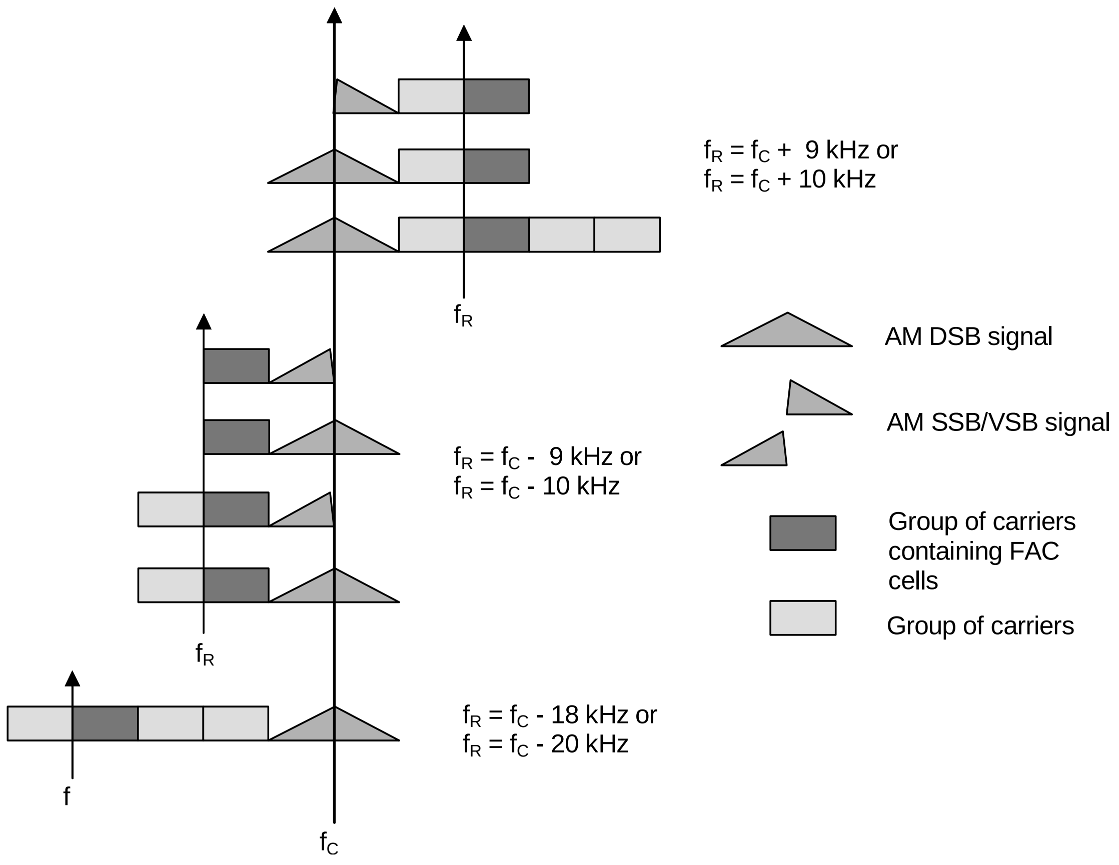


Figure K.1: Example simulcast modes for whole channel offsets

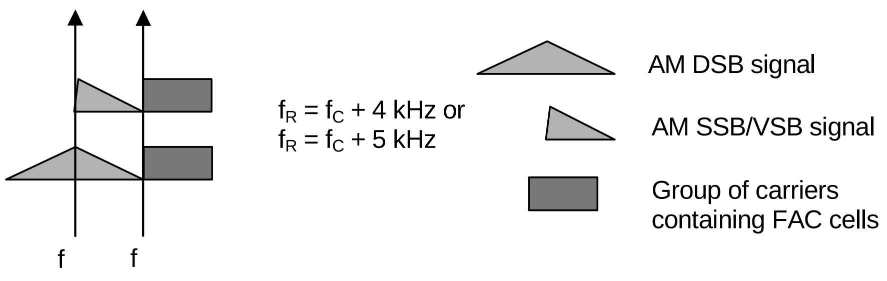


Figure K.2: Example simulcast modes for half channel offsets

Annex L (informative): Pilot reference illustrations

The figures below show the position of the gain reference cells (character "O") for nominal channel bandwidths of up to 10 kHz (spectrum occupancy parameter = 0, 1, 2 or 3). The patterns continue to the right for the 18 kHz and 20 kHz nominal channel bandwidth options. (Spectrum occupancy parameter = 4 or 5).

Mode A:

```
+-> carriers (k)
|
v
symbols (s)                                    DC
                   negative frequencies         :          positive frequencies
    1..1                                        :                                        1..1
    1..0..33333333322222222221111111111000000000:00000000011111111112222222222333333333..0..1
    4..2..87654321098765432109876543210987654321:12345678901234567890123456789012345678..2..4
                                                :
 0  .  .  O...................O................:::O...................O................  O  .
 1  O  .  ....O...................O............:::....O...................O............  .  .
 2  .  .  ........O...................O........:::........O...................O........  .  .
 3  .  .  ............O...................O....:::............O...................O....  .  O
 4  .  O  ................O...................O:::................O...................O  O  .
 5  .  .  O...................O................:::O...................O................  .  .
 6  O  .  ....O...................O............:::....O...................O............  .  .
 7  .  .  ........O...................O........:::........O...................O........  .  .
 8  .  .  ............O...................O....:::............O...................O....  .  O
 9  .  O  ................O...................O:::................O...................O  O  .
10  .  .  O...................O................:::O...................O................  .  .
11  O  .  ....O...................O............:::....O...................O............  .  .
12  .  .  ........O...................O........:::........O...................O........  .  .
13  .  .  ............O...................O....:::............O...................O....  .  O
14  .  O  ................O...................O:::................O...................O  O  .
```

Mode B:

```
+-> carriers (k)
|
v
symbols (s)                                    DC
                   negative frequencies         :          positive frequencies
    1                                           :                                           1
    0..9..33333333322222222221111111111000000000:00000000011111111112222222222333333333..9..0
    3..1..87654321098765432109876543210987654321:12345678901234567890123456789012345678..1..3
                                                :
 0  .  .  ...O.....O.....O.....O.....O.....O....:O.....O.....O.....O.....O.....O.....O.  O  O
 1  .  .  .....O.....O.....O.....O.....O.....O..:..O.....O.....O.....O.....O.....O.....  .  .
 2  O  O  .O.....O.....O.....O.....O.....O.....O:....O.....O.....O.....O.....O.....O...  .  .
 3  .  .  ...O.....O.....O.....O.....O.....O....:O.....O.....O.....O.....O.....O.....O.  O  O
 4  .  .  .....O.....O.....O.....O.....O.....O..:..O.....O.....O.....O.....O.....O.....  .  .
 5  O  O  .O.....O.....O.....O.....O.....O.....O:....O.....O.....O.....O.....O.....O...  .  .
 6  .  .  ...O.....O.....O.....O.....O.....O....:O.....O.....O.....O.....O.....O.....O.  O  O
 7  .  .  .....O.....O.....O.....O.....O.....O..:..O.....O.....O.....O.....O.....O.....  .  .
 8  O  O  .O.....O.....O.....O.....O.....O.....O:....O.....O.....O.....O.....O.....O...  .  .
 9  .  .  ...O.....O.....O.....O.....O.....O....:O.....O.....O.....O.....O.....O.....O.  O  O
10  .  .  .....O.....O.....O.....O.....O.....O..:..O.....O.....O.....O.....O.....O.....  .  .
11  O  O  .O.....O.....O.....O.....O.....O.....O:....O.....O.....O.....O.....O.....O...  .  .
12  .  .  ...O.....O.....O.....O.....O.....O....:O.....O.....O.....O.....O.....O.....O.  O  O
13  .  .  .....O.....O.....O.....O.....O.....O..:..O.....O.....O.....O.....O.....O.....  .  .
14  O  O  .O.....O.....O.....O.....O.....O.....O:....O.....O.....O.....O.....O.....O...  .  .
```

Mode C:

```
+-> carriers (k)
|
v
symbols (s)                                    DC
                   negative frequencies         :          positive frequencies
                                                :
    6.....333333333222222222211111111110000000000:000000000111111111122222222223333333333.....6
    9.....87654321098765432109876543210987654321:12345678901234567890123456789012345678.....9
                                                :
 0  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
 1  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
 2  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
 3  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
 4  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
 5  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
 6  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
 7  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
 8  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
 9  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
10  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
11  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
12  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
13  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
14  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
15  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
16  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
17  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
18  .     ...0...0...0...0...0...0...0...0...0..:0...0...0...0...0...0...0...0...0...0.     0
19  0     .0...0...0...0...0...0...0...0...0...0:..0...0...0...0...0...0...0...0...0...     .
```

Mode D:

```
+-> carriers (k)
|
v
symbols (s)                                    DC
                   negative frequencies         :          positive frequencies
:
    44444333333333322222222221111111111000000000:00000000011111111112222222222333333333344444
    43210987654321098765432109876543210987654321:12345678901234567890123456789012345678901234
                                                :
 0  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
 1  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
 2  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
 3  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
 4  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
 5  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
 6  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
 7  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
 8  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
 9  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
10  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
11  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
12  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
13  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
14  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
15  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
16  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
17  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
18  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
19  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
20  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
21  0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.:0..0..0..0..0..0..0..0..0..0..0..0..0..0..0.
22  .0..0..0..0..0..0..0..0..0..0..0..0..0..0..0:.0..0..0..0..0..0..0..0..0..0..0..0..0..0..0
23  ..0..0..0..0..0..0..0..0..0..0..0..0..0..0..:..0..0..0..0..0..0..0..0..0..0..0..0..0..0..
```

The figures below show the position of the gain reference cells (character 'O'), the frequency reference cells (character 'f') and FAC cells (character 'x') for the DC to 4,5 kHz (nominal).

Mode A: positions for pilot cells

```
+-> carriers (k)
|
v
symbols (s)

DC (not used)
     :                                                                                                   111
     :000000000111111111122222222223333333333444444444455555555556666666666777777777788888888889999999999000
     :123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
     :
 0   ::0...............f...0...................0...........f.......0.........f.........0...................0
 1   ::....0...........f.......0...................0.......f...........0.....f.............0................
 2   ::........0.......f.......x...0...............x...0...f...........x...0.f.............x...0............
 3   ::........x...0...f...........x...0...............x...f...............x.f.0...............x...0........
 4   ::............x...f...x...........x...0...............f...0...x.........f.x...0...............x...0....
 5   ::0...............f...0...x...........x...0...........f...x...0...x.....f.....x...0...................0
 6   ::....0...........f...x...0...x...........x...0.......f.......x...0...x.f.........x...0................
 7   ::........0.......f.......x...0...x...........x...0...f...........x...0.f.x...........x...0............
 8   ::........x...0...f...........x...0...x...........x...f...x...........x.f.0...x...........x...0........
 9   ::............x...f...x...........x...0...x...........f...0...x.........f.x...0...x...........x...0....
10   ::0...............f...0...x...........x...0...x.......f.......0...x.....f.........0...x...............0
11   ::....0...x.......f.......0...x...............0...x...f...........0...x.f.............0...x............
12   ::........0...x...f...........0...x...............0...f...............0.f.x...............0...x........
13   ::............0...f...............0...x...............f...x.............f.0...x...............0........
14   ::................f...................0...............f...0.............f.....0...................0....
```

Mode B: positions for pilot cells

```
+-> carriers (k)
|
v
symbols (s)

DC (not used)
     :
     :0000000001111111111222222222233333333334444444444555555555566666666667777777777888888888899
     :1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901
     :
 0   :0.....0.....0..f..0.....0.....0.....0.....0....f0.....0.....0..f..0.....0.....0.....0.....0
 1   :..0.....0.....0f....0.....0.....0.....0.....0..f..0.....0.....0f....0.....0.....0.....0....
 2   :....0.....0.x..f0.....0.x...0.....0.....0.x...0f....0.x...0....f0.x...0.....0.....0.....0..
 3   :0.....0.....0.xf..0.....0.x...0.....0.....0.x..f0.....0.x...0..f..0.x...0.....0.....0.....0
 4   :..0.....0.....0fx...0.....0.x...0.....0.....0.xf..0.....0.x...0f....0.x...0.....0.....0....
 5   :....0.....0....f0.x...0.....0.x...0.....0.....0fx...0.....0.x..f0.....0.x...0.....0.....0..
 6   :0.....0.x...0..f..0.x...0.....0.x...0.....0....f0.x...0.....0.xf..0.....0.x...0.....0.....0
 7   :..0.....0.x...0f....0.x...0.....0.x...0.....0..f..0.x...0.....0fx...0.....0.x...0.....0....
 8   :....0.....0.x..f0.....0.x...0.....0.x...0.....0f....0.x...0....f0.x...0.....0.x...0.....0..
 9   :0.....0.....0.xf..0.....0.x...0.....0.x...0....f0.....0.x...0..f..0.x...0.....0.x...0.....0
10   :..0.....0.....0fx...0.....0.x...0.....0.x...0..f..0.....0.x...0f....0.x...0.....0.x...0....
11   :....0.....0....f0.x...0.....0.x...0.....0.x...0f....0.....0.x..f0.....0.x...0.....0.....0..
12   :0.....0.....0..f..0.x...0.....0.x...0.....0.x..f0.....0.....0.xf..0.....0.x...0.....0.....0
13   :..0.....0.....0f....0.x...0.....0.x...0.....0.xf..0.....0.....0fx...0.....0.x...0.....0....
14   :....0.....0....f0.....0.....0.....0.....0.....0f....0.....0....f0.....0.....0.....0.....0..
```

Mode C: positions for pilot cells

```
+-> carriers (k)
|
v
symbols (s)

DC (not used)
     :
     :000000000111111111122222222223333333333444444444455555555556666666666
     :123456789012345678901234567890123456789012345678901234567890123456789
     :
 0   :0...0...0.f.0...0...0...0...0...f...0...0..f0...0...0...0...0...0...0
 1   :..0...0...f...0...0...0...0...0.f.0...0...0f..0...0...0...0...0...0..
 2   :0...0...0.f.0...0...0...0...0...f...0...0..f0...0...0...0...0...0...0
 3   :..0...0.x.f...0...0.x.0...0...0.f.0...0...0fx.0...0...0.x.0...0...0..
 4   :0...0...0.f.0...0...0.x.0...0...f.x.0...0..f0.x.0...0...0...0...0...0
 5   :..0...0...f.x.0...0...0.x.0...0.f.0.x.0...0f..0.x.0...0...0...0...0..
 6   :0...0...0.f.0.x.0...0...0.x.0...f...0.x.0..f0...0.x.0...0...0...0...0
 7   :..0.x.0...f...0.x.0...0...0.x.0.f.0...0.x.0f..0...0.x.0...0...0...0..
 8   :0...0.x.0.f.0...0.x.0...0...0.x.f...0...0.xf0...0...0.x.0...0...0...0
 9   :..0...0.x.f...0...0.x.0...0...0.f.0...0...0fx.0...0...0.x.0...0...0..
10   :0...0...0.f.0...0...0.x.0...0...f.x.0...0..f0x..0...0...0...0...0...0
11   :..0...0...f.x.0...0...0.x.0...0.f.0.x.0...0f..0.x.0...0...0...0...0..
```

```
12   :0...0...0.f.0.x.0...0...0.x.0...f...0.x.0..f0...0.x.0...0...0...0...0
13   :..0.x.0...f...0.x.0...0...0.x.0.f.0...0.x.0f..0...0.x.0...0...0...0..
14   :0...0.x.0.f.0...0.x.0...0...0.x.f...0...0.xf0...0...0.x.0...0...0...0
15   :..0...0.x.f...0...0.x.0...0...0.f.0...0...0fx.0...0...0.x.0...0...0..
16   :0...0...0.f.0...0...0.x.0...0...f.x.0...0..f0.x.0...0...0...0...0...0
17   :..0...0...f.x.0...0...0.x.0...0.f.0.x.0...0f..0.x.0...0...0...0...0..
18   :0...0...0.f.0.x.0...0...0.x.0...f...0.x.0..f0...0.x.0...0...0...0...0
19   :..0...0...f...0...0...0...0...0.f.0...0...0f..0...0...0...0...0...0..
```

Mode D: positions for pilot cells

```
+-> carriers (k)
|
v
symbols (s)

DC (not used)
     :
     :0000000001111111111222222222233333333334444
     :1234567890123456789012345678901234567890123 4
     :
 0   :0..0..f..0..0..0..0.f0..0..f..0..0..0..0..0.
 1   :.0..0.f0..0..0..0..0f.0..0.f0..0..0..0..0..0
 2   :..0..0f.0..0..0..0..f..0..0f.0..0..0..0..0..
 3   :0..0..f.x0..0..0.x0.f0..0.xf..0..0..0..0..0.
 4   :.0..0.f0.x0..0..0.x0f.0..0.f0..0..0..0..0..0
 5   :..0..0f.0.x0..0..0.xf..0..0fx0..0..0..0..0..
 6   :0..0..f..0.x0..0..0.f0..0..f.x0..0..0..0..0.
 7   :.0..0.f0..0.x0..0..0fx0..0.f0.x0..0..0..0..0
 8   :..0.x0f.0..0.x0..0..f.x0..0f.0.x0..0..0..0..
 9   :0..0.xf..0..0.x0..0.f0.x0..f..0.x0..0..0..0.
10   :.0..0.f0..0..0.x0..0f.0.x0.f0..0.x0..0..0..0
11   :..0..0fx0..0..0.x0..f..0.x0f.0..0.x0..0..0..
12   :0..0..f.x0..0..0.x0.f0..0.xf..0..0.x0..0..0.
13   :.0..0.f0.x0..0..0.x0f.0..0.f0..0..0.x0..0..0
14   :..0..0f.0.x0..0..0.xf..0..0fx0..0..0..0..0..
15   :0..0..f..0.x0..0..0.f0..0..f.x0..0..0..0..0.
16   :.0..0.f0..0.x0..0..0fx0..0.f0.x0..0..0..0..0
17   :..0.x0f.0..0.x0..0..f.x0..0f.0.x0..0..0..0..
18   :0..0.xf..0..0.x0..0.f0.x0..f..0.x0..0..0..0.
19   :.0..0.f0..0..0.x0..0f.0.x0.f0..0.x0..0..0..0
20   :..0..0fx0..0..0.x0..f..0.x0f.0..0.x0..0..0..
21   :0..0..f.x0..0..0.x0.f0..0.xf..0..0.x0..0..0.
22   :.0..0.f0.x0..0..0.x0f.0..0.f0..0..0.x0..0..0
23   :..0..0f.0..0..0..0..f..0..0f.0..0..0..0..0..
```

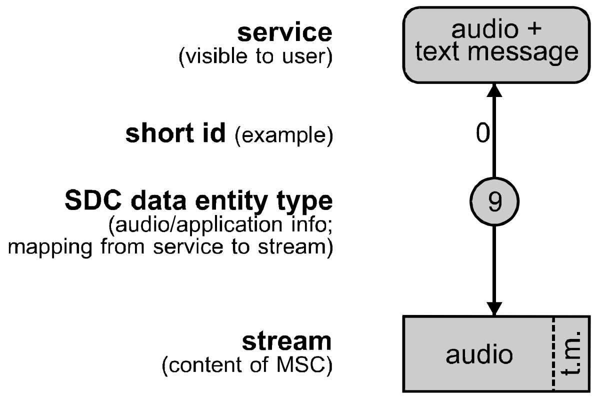
Annex M (informative): MSC configuration examples

The examples below demonstrate some possibilities for configuring the MSC. Especially the mapping from services to audio or data streams is covered together with some limitations which must be respected when assembling the DRM multiplex.

General Preface:

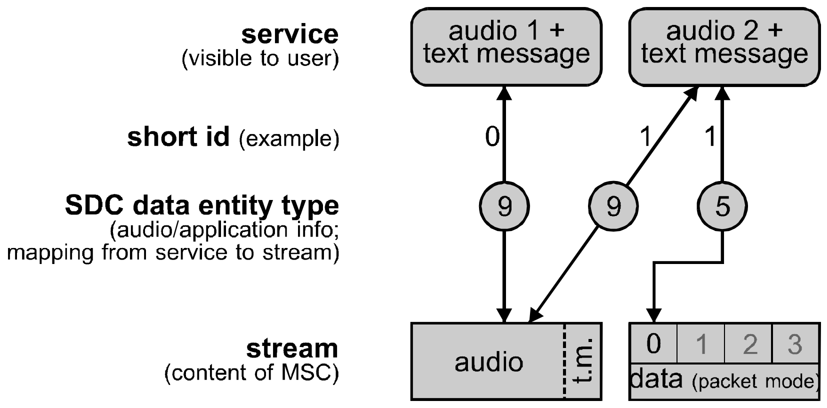
- The DRM multiplex may contain up to four streams in the MSC, each carrying audio or data information.
- An audio stream is described by SDC data entity type 9. A packet mode data stream consists of 1 - 4 "sub-streams" (distinguished by their packet id), each described by SDC data entity type 5.
- 1 - 4 services can be signalled to the user. A data service points to one data (sub-)stream. An audio service points to one audio stream plus optionally to text message information (contained in the audio stream) and/or one data (sub-)stream.
- Audio services are mapped to audio streams by SDC data entity type 9. Data (and audio) services are mapped to data streams by SDC data entity type 5.
- If several services point to the same stream, the stream configuration in SDC data entity type 5 or 9 must be identical.

EXAMPLE 1: A very simple DRM multiplex consists of just one single audio service pointing to the one and only audio stream. The audio stream may contain text messages.



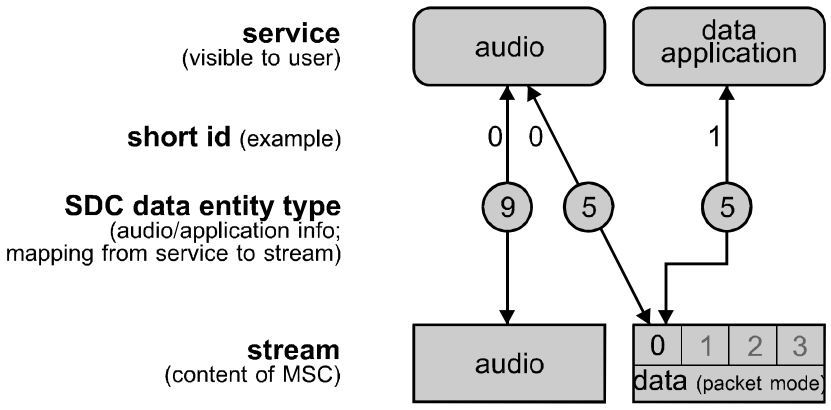
EXAMPLE 2: There are two audio services. Both point to the same audio stream. One of these services points to an additional packet mode data sub-stream carrying a multimedia data application. In total there are one audio stream and one data stream using packet mode.

Note that if multiple services point to the same stream the configuration for that stream (carried in SDC data entities type 9 for audio information or in SDC data entities type 5 for data application information) must be the same. So in this example both or none of the audio services can comprise the text message service.



EXAMPLE 3: There is one audio service and one data service signalled to the user. The DRM multiplex consists of one audio stream and one data stream in packet mode containing one sub-stream. The data service points to the data application carried in the one sub-stream of the data stream. The audio service points to the audio stream and additionally also to the one sub-stream of the data stream.

Note that in this case the data application description in the two SDC data entities type 5 must be identical for both services referencing the same packet mode sub-stream.



EXAMPLE 4: To make maximum use of the possibilities of a DRM multiplex, the following scenario is possible:

There are three audio services each pointing to its own audio stream. In addition to audio, each of these three audio streams carries text messages. Every audio service also points to its own data application carried as a sub-stream of a packet mode data stream (being the fourth stream in the DRM multiplex). Since a data stream can carry up to four sub-streams in packet mode, there can be an additional data service pointing to the fourth sub-stream of the data stream.

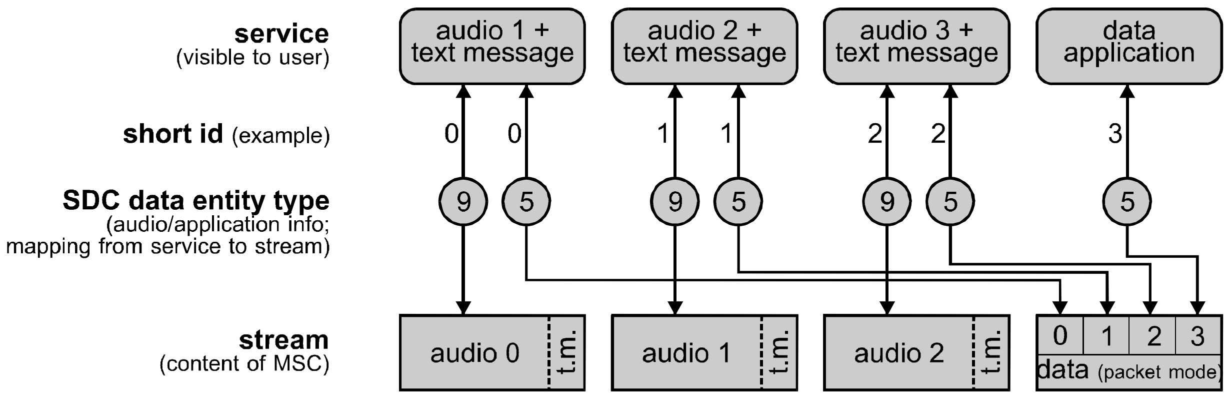
In total there are:

- three audio services;
- one data service.

These four services point to 10 different "logical channels":

- three different audio streams with their own text messages;
- one data stream in packet mode with four sub-streams.

NOTE: The packet mode configuration parameters (e.g. the packet length) of all four SDC data entities type 5 (describing the four sub-streams of the one packet mode data stream) must be identical!


service
(visible to user)
short id (example)
SDC data entity type
(audio/application info;
mapping from service to stream)
stream
(content of MSC)
audio 1 + text message
audio 2 + text message
audio 3 + text message
data application
0 0 1 1 2 2 3
9 5 9 5 9 5 5
audio 0
t.m.
audio 1
t.m.
audio 2
t.m.
0 1 2 3
data (packet mode)

Annex N (informative): HVXC parameters

Table N.1: HVXC source coder parameters

Name	Meaning	Number	Fixed Rate	
		of bits	2,0 kbit/s	4,0 kbit/s
LSP1	LSP index 1	5	X	X
LSP2	LSP index 2	7	X	X
LSP3	LSP index 3	5	X	X
LSP4	LSP index 4	1	X	X
VUV	voiced/unvoiced flag	2	X	X
Pitch	pitch parameter	7	X	X
SE_shape1	spectrum index 0	4	X	X
SE_shape2	spectrum index 1	4	X	X
SE_gain	spectrum gain index	5	X	X
VX_shape1[0]	stochastic codebook index 0	6	X	X
VX_shape1[1]	stochastic codebook index 1	6	X	X
VX_gain1[0]	gain codebook index 0	4	X	X
VX_gain1[1]	gain codebook index 1	4	X	X
LSP5	LSP index 5	8		X
SE_shape3	4k spectrum index 0	7		X
SE_shape4	4k spectrum index 1	10		X
SE_shape5	4k spectrum index 2	9		X
SE_shape6	4k spectrum index 3	6		X
VX_shape2[0]	4k stochastic codebook index 0	5		X
VX_shape2[1]	4k stochastic codebook index 1	5		X
VX_shape2[2]	4k stochastic codebook index 2	5		X
VX_shape2[3]	4k stochastic codebook index 3	5		X
VX_gain2[0]	4k gain codebook index 0	3		X
VX_gain2[1]	4k gain codebook index 1	3		X
VX_gain2[2]	4k gain codebook index 2	3		X
VX_gain2[3]	4k gain codebook index 3	3		X
NOTE: X indicates that the parameter is used in the corresponding mode.				

Table N.2: Bit allocations of 2,0/4,0 kbit/s HVXC coder (fixed rate)

Parameter	2,0 kbit/s (fixed rate)		4,0 kbit/s (fixed rate)	
	Voiced	Unvoiced	Voiced	Unvoiced
LSP	18 bits/20 ms	18 bits/20 ms	18 bits/20 ms	18 bits/20 ms
LSP(enh)			8 bits/20 ms	8 bits/20 ms
V/UV	2 bits/20 ms	2 bits/20 ms	2 bits/20 ms	2 bits/20 ms
Pitch	7 bits/20 ms		7 bits/20 ms	
spectral shape	4 + 4 bits/20 ms		4 + 4 bits/20 ms	
spectral gain	5 bits/20 ms		5 bits/20 ms	
spectral shape(enh)			32 bits/20 ms	
VXC shape		6 bits/10 ms		6 bits/10 ms
VXC gain		4 bits/10 ms		4 bits/10 ms
VXC shape(enh)				5 bits/5 ms
VXC gain(enh)				3 bits/5 ms
Total - 2,0 kbit/s	40 bits/20 ms	40 bits/20 ms		
Total - 4,0 kbit/s			80 bits/20 ms	80 bits/20 ms

Annex O (informative): Bibliography

DRM Output Requirements, November 1999.

ITU-R Circular letter LCCE/39: "Submission of candidate systems for digital sound broadcasting at frequencies below 30 MHz, September 1999".

ITU-R Recommendation BS.1348: "Service requirements for digital sound broadcasting to vehicular, portable and fixed receivers using terrestrial transmitters in the LF, MF and HF bands".

ITU-R Recommendation BS.1349: "Implementation of digital sound broadcasting to vehicular, portable and fixed receivers using terrestrial transmitters in the LF, MF and HF bands".

History

Document history		
V1.1.1	September 2001	Publication as TS 101 980
V1.2.1	July 2002	Membership Approval Procedure MV 20020927: 2002-07-30 to 2002-09-27
V1.2.2	April 2003	Publication