ETSI TS 126 290 V8.0.0 (2009-01)

Technical Specification

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Audio codec processing functions;
Extended Adaptive Multi-Rate - Wideband (AMR-WB+) codec;
Transcoding functions
(3GPP TS 26.290 version 8.0.0 Release 8)

Reference
RTS/TSGS-0426290v800

Keywords
GSM, LTE, UMTS

ETSI

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

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

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

Intellectual Property Rights

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

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

Foreword

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

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

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

Contents

Intellectual Property Rights 2
Foreword 2
Foreword 6
1 Scope 7
2 References 7
3 Definitions and abbreviations 7
3.1 Definitions 7
3.2 Abbreviations 9
4 Outline description 9
4.1 Functional description of audio parts 9
4.2 Preparation of input samples 10
4.3 Principles of the extended adaptive multi-rate wideband codec 10
4.3.1 Encoding and decoding structure 11
4.3.2 LP analysis and synthesis in low-frequency band 13
4.3.3 ACELP and TCX coding 13
4.3.4 Coding of high-frequency band 13
4.3.5 Stereo coding 13
4.3.6 Low complexity operation 13
4.3.7 Frame erasure concealment 13
4.3.8 Bit allocation 14
5 Functional description of the encoder 16
5.1 Input signal pre-processing 16
5.1.1 High Pass Filtering 16
5.1.2 Stereo Signal Downmixing/Bandsplitting 16
5.2 Principle of the hybrid ACELP/TCX core encoding 17
5.2.1 Timing chart of the ACELP and TCX modes 17
5.2.2 ACELP/TCX mode combinations and mode encoding 18
5.2.3 ACELP/TCX closed-loop mode selection 19
5.2.4 ACELP/TCX open-loop mode selection 20
5.3 Hybrid ACELP/TCX core encoding description 24
5.3.1 Pre-emphasis 24
5.3.2 LP analysis and interpolation 24
5.3.2.1 Windowing and auto-correlation computation 24
5.3.2.2 Levinson-Durbin algorithm 24
5.3.2.3 LP to ISP conversion 24
5.3.2.4 ISP to LP conversion 24
5.3.2.5 Quantization of the ISP coefficient 25
5.3.2.6 Interpolation of the ISPs 25
5.3.3 Perceptual weighting 25
5.3.4 ACELP Excitation encoder 25
5.3.4.1 Open-loop pitch analysis 25
5.3.4.2 Impulse response computation 25
5.3.4.3 Target signal computation 26
5.3.4.4 Adaptive codebook 26
5.3.4.5 Algebraic codebook 26
5.3.4.5.1 Codebook structure 26
5.3.4.5.2 Pulse indexing 26
5.3.4.5.3 Codebook search 26
5.3.4.6 Quantization of the adaptive and fixed codebook gains 26
5.3.5 TCX Excitation encoder 27
5.3.5.1 TCX encoder block diagram 27
5.3.5.2 Computation of the target signal for transform coding 30
5.3.5.3 Zero-input response subtraction 30

5.3.5.4 Windowing of target signal 31
5.3.5.5 Transform 32
5.3.5.6 Spectrum pre-shaping 32
5.3.5.7 Split multi-rate lattice VQ 33
5.3.5.8 Spectrum de-shaping 38
5.3.5.9 Inverse transform 38
5.3.5.10 Gain optimization and quantization 38
5.3.5.11 Windowing for overlap-and-add 39
5.3.5.12 Memory update 39
5.3.5.13 Excitation signal computation 39
5.4 Mono Signal High-Band encoding (BWE) 39
5.5 Stereo signal encoding 42
5.5.1 Stereo Signal Low-Band Encoding 42
5.5.1.1 Principle 43
5.5.1.2 Signal Windowing 44
5.5.1.3 Pre-echo mode 44
5.5.1.4 Redundancy reduction 44
5.5.2 Stereo Signal Mid-Band Processing 44
5.5.2.1 Principle 44
5.5.2.2 Residual computation 45
5.5.2.3 Filter computation, smoothing and quantization 45
5.5.2.4 Channel energy matching 45
5.5.3 Stereo Signal High-Band Processing 46
5.6 Packetization 46
5.6.1 Packetization of TCX encoded parameters 46
5.6.1.1 Multiplexing principle for a single binary table 47
5.6.1.2 Multiplexing in case of multiple binary tables 48
5.6.2 Packetization procedure for all parameters 50
5.6.3 TCX gain multiplexing 52
5.6.4 Stereo Packetization 53

6 Functional description of the decoder 53
6.1 Mono Signal Low-Band synthesis 53
6.1.1 ACELP mode decoding and signal synthesis 54
6.1.2 TCX mode decoding and signal synthesis 54
6.1.3 Post-processing of Mono Low-Band signal 57
6.2 Mono Signal High-Band synthesis 59
6.3 Stereo Signal synthesis 62
6.3.1 Stereo signal low-band synthesis 63
6.3.2 Stereo Signal Mid-Band synthesis 64
6.3.3 Stereo Signal High-Band synthesis 65
6.3.4 Stereo output signal generation 65
6.4 Stereo to mono conversion 65
6.4.1 Low-Band synthesis 65
6.4.2 High-Band synthesis 65
6.5 Bad frame concealment 66
6.5.1 Mono 66
6.5.1.1 Mode decoding and extrapolation 66
6.5.1.2 TCX bad frame concealment 68
6.5.1.2.1 Spectrum de-shaping 68
6.5.1.2.2 Spectrum Extrapolation 68
6.5.1.2.3 Amplitude Extrapolation 69
6.5.1.2.4 Phase Extrapolation 69
6.5.2 Stereo 70
6.5.2.1 Low-band 70
6.5.2.2 Mid-band 71
6.6 Output signal generation 71

7 Detailed bit allocation of the Extended AMR-WB codec 72

8 Storage and Transport Interface formats 78
8.1 Available Modes and Bitrates 78
8.2 AMR-WB+ Transport Interface Format 81

8.3 AMR-WB+ File Storage Format 83

Annex A (informative): Change history 85

History 86

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP). This document describes the Extended Adaptive Multi-Rate Wideband (AMR-WB+) coder within the 3GPP system.

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

Version x.y.z

where:

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

1 Scope

This Telecommunication Standard (TS) describes the detailed mapping from input blocks of monophonic or stereophonic audio samples in 16 bit uniform PCM format to encoded blocks and from encoded blocks to output blocks of reconstructed monophonic or stereophonic audio samples. The coding scheme is an extension of the AMR-WB coding scheme [2] and is referred to as extended AMR-WB or AMR-WB+ codec. It comprises all AMR-WB speech codec modes including VAD/DTX [3] as well as extended functionality for encoding general audio signals such as music, speech, mixed, and other signals.

In the case of discrepancy between the requirements described in the present document and the ANSI-C code computational description of these requirements contained in [4], [5], the description in [4], [5], respectively, will prevail. The ANSI-C code is not described in the present document, see [4], [5] for a description of the floating-point or, respectively, fixed-point ANSI-C code.

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] GSM 43.050: " Digital cellular telecommunications system (Phase 2); Transmission planning aspects of the speech service in the GSM Public Land Mobile Network (PLMN) system"

[2] 3GPP TS 26.194: "AMR wideband speech codec; Voice Activity Detection (VAD)".

[3] 3GPP TS 26.190: " AMR Wideband speech codec; Transcoding functions ".

[4] 3GPP TS 26.304: "ANSI-C code for the floating point Extended AMR Wideband codec".

[5] 3GPP TS 26.273: "ANSI-C code for the fixed point Extended AMR Wideband codec".

[6] M. Xie and J.-P. Adoul, "Embedded algebraic vector quantization (EAVQ) with application to wideband audio coding," IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Atlanta, GA, U.S.A, vol. 1, pp. 240-243, 1996.

[7] J.H. Conway and N.J.A. Sloane, "A fast encoding method for lattice codes and quantizers," *IEEE Trans. Inform. Theory*, vol. IT-29, no. 6, pp. 820-824, Nov. 1983

[8] 3GPP TS 26.193: "AMR Wideband speech codec; Source controlled rate operation".

[9] 3GPP TS 26.244: "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)"

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and apply.

adaptive codebook: The adaptive codebook contains excitation vectors that are adapted for every subframe. The adaptive codebook is derived from the long-term filter state. The lag value can be viewed as an index into the adaptive codebook.

algebraic codebook: A fixed codebook where algebraic code is used to populate the excitation vectors (innovation vectors). The excitation contains a small number of nonzero pulses with predefined interlaced sets of potential positions. The amplitudes and positions of the pulses of the k^{th} excitation codevector can be derived from its index k through a rule requiring no or minimal physical storage, in contrast with stochastic codebooks whereby the path from the index to the associated codevector involves look-up tables.

anti-sparseness processing: An adaptive post-processing procedure applied to the fixed codebook vector in order to reduce perceptual artifacts from a sparse fixed codebook vector.

closed-loop pitch analysis: This is the adaptive codebook search, i.e., a process of estimating the pitch (lag) value from the weighted input speech and the long term filter state. In the closed-loop search, the lag is searched using error minimization loop (analysis-by-synthesis). In the adaptive multi-rate wideband codec, closed-loop pitch search is performed for every subframe.

direct form coefficients: One of the formats for storing the short term filter parameters. In the adaptive multi-rate wideband codec, all filters which are used to modify speech samples use direct form coefficients.

fixed codebook: The fixed codebook contains excitation vectors for speech synthesis filters. The contents of the codebook are non-adaptive (i.e., fixed). In the adaptive multi-rate wideband codec, the fixed codebook is implemented using an algebraic codebook.

fractional lags: A set of lag values having sub-sample resolution. In the adaptive multi-rate wideband codec a sub-sample resolution of 1/4th or 1/2nd of a sample is used.

super frame: A time interval equal to 1024 samples (80ms at a 12.8 kHz sampling rate).

frame: A time interval equal to 256 samples (20ms at a 12.8 kHz sampling rate).

Immittance Spectral Frequencies: (see Immittance Spectral Pair)

Immittance Spectral Pair: Transformation of LPC parameters. Immittance Spectral Pairs are obtained by decomposing the inverse filter transfer function A(z) to a set of two transfer functions, one having even symmetry and the other having odd symmetry. The Immittance Spectral Pairs (also called as Immittance Spectral Frequencies) are the roots of these polynomials on the z-unit circle.

integer lags: A set of lag values having whole sample resolution.

interpolating filter: An FIR filter used to produce an estimate of sub-sample resolution samples, given an input sampled with integer sample resolution. In this implementation, the interpolating filter has low pass filter characteristics. Thus the adaptive codebook consists of the low-pass filtered interpolated past excitation.

inverse filter: This filter removes the short term correlation from the speech signal. The filter models an inverse frequency response of the vocal tract.

lag: The long term filter delay. This is typically the true pitch period, or its multiple or sub-multiple.

LP analysis window: For each frame, the short term filter coefficients are computed using the high pass filtered speech samples within the analysis window. In the adaptive multi-rate wideband codec, the length of the analysis window is always 384 samples. For all the modes, a single asymmetric window is used to generate a single set of LP coefficients. The 5 ms look-ahead is used in the analysis.

LP coefficients: Linear Prediction (LP) coefficients (also referred as Linear Predictive Coding (LPC) coefficients) is a generic descriptive term for the short term filter coefficients.

open-loop pitch search: A process of estimating the near optimal lag directly from the weighted speech input. This is done to simplify the pitch analysis and confine the closed-loop pitch search to a small number of lags around the open-loop estimated lags. In the adaptive multi-rate wideband codec, an open-loop pitch search is performed in every other subframe.

residual: The output signal resulting from an inverse filtering operation.

short term synthesis filter: This filter introduces, into the excitation signal, short term correlation which models the impulse response of the vocal tract.

perceptual weighting filter: This filter is employed in the analysis-by-synthesis search of the codebooks. The filter exploits the noise masking properties of the formants (vocal tract resonances) by weighting the error less in regions near the formant frequencies and more in regions away from them.

subframe: A time interval equal to 64 samples (5ms at 12.8 kHz sampling rate).

vector quantization: A method of grouping several parameters into a vector and quantizing them simultaneously.

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

zero state response: The output of a filter due to the present input, given that no past inputs have been applied, i.e., given that the state information in the filter is all zeroes.

3.2 Abbreviations

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

TCX	Transform coded excitation
ACELP	Algebraic Code Excited Linear Prediction
AGC	Adaptive Gain Control
AMR	Adaptive Multi-Rate
AMR-WB	Adaptive Multi-Rate Wideband
AMR-WB+	Extended Adaptive Multi-Rate Wideband
CELP	Code Excited Linear Prediction
FIR	Finite Impulse Response
ISF	Immittance Spectral Frequency
ISP	Immittance Spectral Pair
ISPP	Interleaved Single-Pulse Permutation
LP	Linear Prediction
LPC	Linear Predictive Coding
LTP	Long Term Predictor (or Long Term Prediction)
MA	Moving Average
MRWB-ACELP	Wideband Multi-Rate ACELP
S-MSVQ	Split-MultiStage Vector Quantization
WB	Wideband

4 Outline description

This TS is structured as follows:

Section 4.1 contains a functional description of the audio parts including the A/D and D/A functions. Section 4.2 describes input format for the AMR-WB+ encoder and the output format for the AMR-WB+ decoder. Section 4.3 presents a simplified description of the principles of the AMR-WB codec. In subclause 4.4, the sequence and subjective importance of encoded parameters are given.

Section 5 presents the functional description of the encoding functions of the AMR-WB+ extension modes, whereas clause 6 describes the decoding procedures for the extension modes. In section 7, the detailed bit allocation of the AMR-WB+ codec extension modes is tabulated. The AMR-WB speech modes are functionally unchanged as well as their bit allocation. Detailed information on them is found in [1].

4.1 Functional description of audio parts

The analogue-to-digital and digital-to-analogue conversion will in principle comprise the elements given below. In case of stereo codec operation, the given principles will be applied to the 2 available audio channels.

1) Analogue to uniform digital PCM

- microphone;
- input level adjustment device;
- input anti-aliasing filter;
- sample-hold device sampling at 16/24/32/48 kHz;
- analogue-to-uniform digital conversion to 16-bit representation.

The uniform format shall be represented in two's complement.

2) Uniform digital PCM to analogue

- conversion from 16-bit uniform PCM sampled at 16/24/32/48 kHz to analogue;
- a hold device;
- reconstruction filter including x/sin(x) correction;
- output level adjustment device;
- earphone or loudspeaker.

In the terminal equipment, the A/D function may be achieved

- by direct conversion to 14-bit uniform PCM format;

For the D/A operation, the inverse operations take place.

4.2 Preparation of input samples

The encoder is fed with data from one/two input channels comprising of samples with a resolution of 16 bits in a 16-bit word. The decoder outputs data in the same format and number of output channels. Though, mono output of decoded stereo signals is supported.

4.3 Principles of the extended adaptive multi-rate wideband codec

The AMR-WB+ audio codec contains all the AMR-WB speech codec modes 1-9 and AMR-WB VAD and DTX. AMR-WB+ extends the AMR-WB codec by adding TCX, bandwidth extension, and stereo.

The AMR-WB+ audio codec processes input frames equal to 2048 samples at an internal sampling frequency F_s . The internal sampling frequency is limited to the range 12800-38400 Hz, see section 8 for more details. The 2048-sample frames are split into two critically sampled equal frequency bands. This results in two superframes of a 1024 samples corresponding to the low frequency (LF) and high frequency (HF) band. Each superframe is divided into four 256-samples frames.

Sampling at the internal sampling rate is obtained by using a variable sampling conversion scheme, which re-samples the input signal.

The LF and HF signals are then encoded using two different approaches: the LF is encoded and decoded using the "core" encoder/decoder, based on switched ACELP and transform coded excitation (TCX). In ACELP mode, the standard AMR-WB codec is used. The HF signal is encoded with relatively few bits (16 bits/frame) using a bandwidth extension (BWE) method.

The basic set of rates are built based on AMR-WB rates in addition to bandwidth extension. The basic set of mono rates are shown in Table 1.

Table 1: Basic set of mono rates

Mono rate(incl. BWE) (bits/frame)	Corresponding AMR-WB mode
208	NA
240	NA
272	12.65
304	14.25
336	15.85
384	18.25
416	19.85
480	23.05

Note that in ACELP mode of operation, compared to AMR-WB, the VAD bit is removed, two bits per frame are added for gain prediction, and 2 bits are added for signaling frame encoding type. This adds 3 bits per frame. Note also that 16 bits/frame is always used for bandwidth extension (to encode the HF band). The first two basic mono rates are similar to other rates except that they use a fixed codebook with 20 bits or 28 bits, respectively.

For stereo coding, the set of stereo extension rates given in Table 2 are used.

Table 2: Basic set of stereo rates

Stereo extension rates (incl. BWE) (Bits/frame)	
40	104
48	112
56	120
64	128
72	136
80	144
88	152
96	160

Note that the bandwidth extension is applied to both channels which requires additional 16 bits/frame for the stereo extension.

A certain mode of operation is obtained by choosing a rate from Table 1, in case of mono operation, or by combining a rate from Table 1 with a stereo extension rate from Table 2, in case of stereo operation. The resulting coding bitrate is (mono rate + stereo rate) Fs / 512.

Examples:

- For an internal sampling frequency of 32 kHz by choosing mono rate equal to 384 bits/frame and without stereo, we can obtain a bit-rate equal to 24 kbps and the frame length would be of a 16 ms duration.
- For an internal sampling frequency of 25.6 kHz by choosing mono rate equal to 272 bits/frame and stereo rate equal to 88 bits/frame, we can obtain a bit-rate equal to 18 kbps and the frame length would be of a 20 ms duration.

Note. The documentation of the AMR-WB+ floating-point C-code in [4] contains further information on how to use the executables compiled from this source code to exercise the various possible uses, in the codec, of mono bit rate, stereo bit rate and internal sampling frequency, and the resulting total bit rates.

4.3.1 Encoding and decoding structure

Figure 1 presents the AMR-WB+ encoder structure. The input signal is separated in two bands. The first band is the low-frequency (LF) signal, which is critically sampled at Fs/2 . The second band is the high-frequency (HF) signal, which is also downsampled to obtain a critically sampled signal. The LF and HF signals are then encoded using two different approaches: the LF signal is encoded and decoded using the "core" encoder/decoder, based on switched ACELP and transform coded excitation (TCX). In ACELP mode, the standard AMR-WB codec is used. The HF signal is encoded with relatively few bits using a bandwidth extension (BWE) method.

The parameters transmitted from encoder to decoder are the mode selection bits, the LF parameters and the HF parameters. The parameters for each 1024-sample super-frame are decomposed into four packets of identical size.

When the input signal is stereo, the Left and right channels are combined into mono signal for ACELP/TCX encoding, whereas the stereo encoding receives both input channels.

Figure 2 presents the AMR-WB+ decoder structure. The LF and HF bands are decoded separately after which they are combined in a synthesis filterbank. If the output is restricted to mono only, the stereo parameters are omitted and the decoder operates in mono mode.

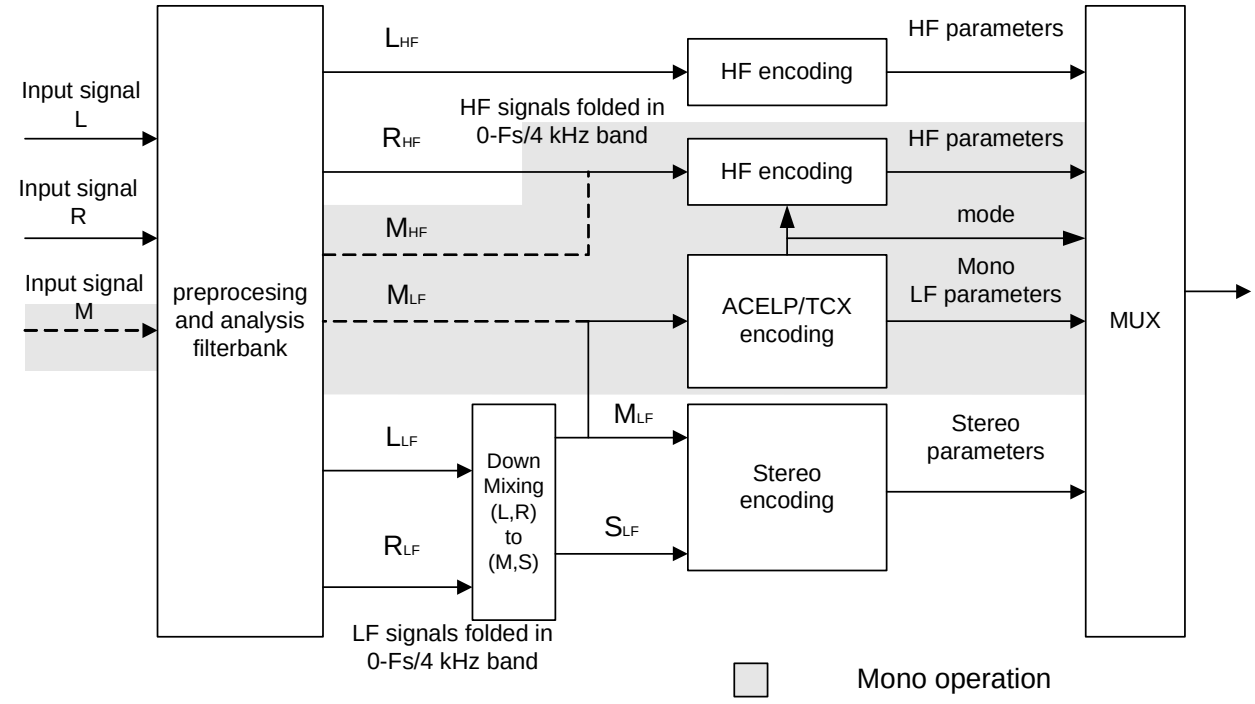


Figure 1: High-level structure of AMR-WB+ encoder

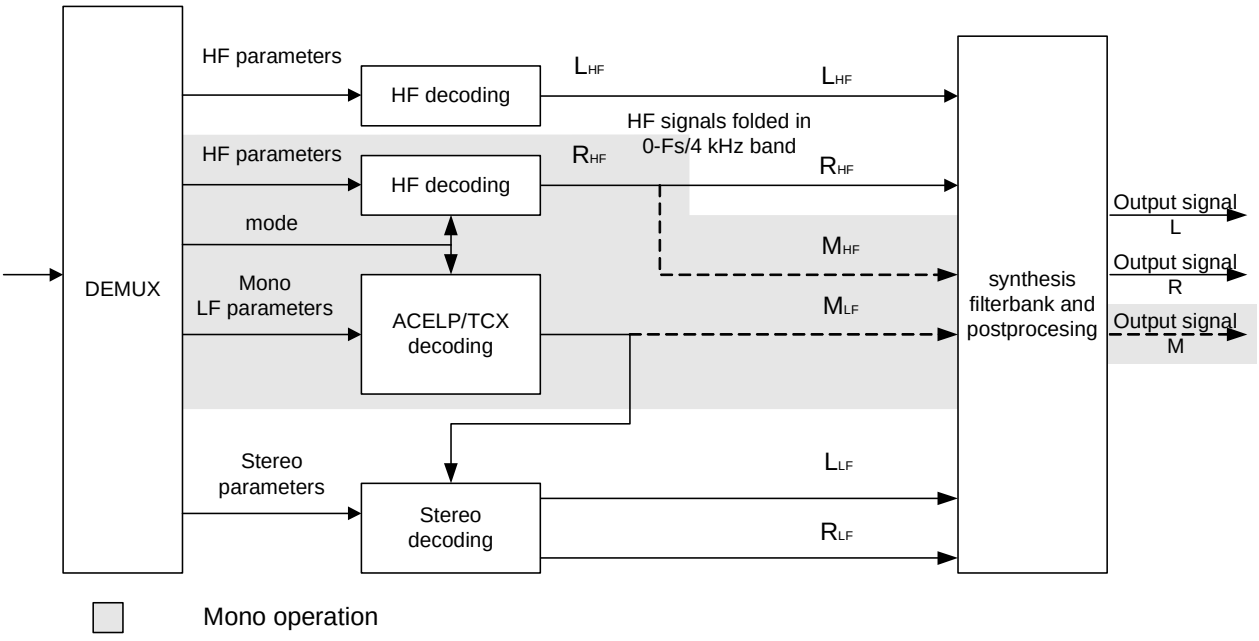


Figure 2: High-level structure of AMR-WB+ decoder

4.3.2 LP analysis and synthesis in low-frequency band

The AMR-WB+ codec applies LP analysis for both the ACELP and TCX modes when encoding the LF signal. The LP coefficients are interpolated linearly at every 64-sample sub-frame. The LP analysis window is a half-cosine of length 384 samples.

4.3.3 ACELP and TCX coding

To encode the core mono signal (0-Fs/4 kHz band), the AMR-WB+ codec utilises either ACELP or TCX coding for each frame. The coding mode is selected based on closed-loop analysis-by-synthesis method. Only 256-sample frames are considered for ACELP frames (as in AMR-WB), whereas frames of 256, 512 or 1024 samples are possible in TCX mode.

ACELP encoding and decoding are similar to standard AMR-WB speech codec. The ACELP coding consists of LTP analysis and synthesis and algebraic codebook excitation. The ACELP coding mode is used in AMR-WB operation within AMR-WB+ codec.

In TCX mode the perceptually weighted signal is processed in the transform domain. The Fourier transformed weighted signal is quantised using split multi-rate lattice quantisation (algebraic VQ). Transform is calculated in 1024, 512 or 256 samples windows. The excitation signal is recovered by inverse filtering the quantised weighted signal through the inverse weighting filter (same weighting filter as in AMR-WB).

4.3.4 Coding of high-frequency band

Whereas the LF signal (0-Fs/4 kHz band) is encoded using the previously described switched ACELP/TCX encoding approach, the HF signal is encoded using a low-rate parametric bandwidth extension (BWE) approach. Only gains and spectral envelope information are transmitted in the BWE approach used to encode the HF signal.

The bandwidth extension is done separately for left and right channel in stereo operation.

4.3.5 Stereo coding

In the case of stereo coding, a similar band decomposition as in the mono case is used. The two channels L and R are decomposed into LF and HF signals. The LF signals of the two channels are down-mixed to form an LF mono signal, (0-Fs/4 kHz band). This mono signal is encoded separately by the core codec.

The LF part of the two channels is further decomposed into two bands (0-5Fs/128 kHz band) and (5Fs/128 kHz- Fs/4 kHz band). The very low frequency (VLF) band is critically down-sampled, and the side signal is computed. The resulting signal is semi-parametrically encoded in the frequency domain using the algebraic VQ. The frequency domain encoding is performed in closed loop by choosing among 40-, 80- and 160-sample frame lengths.

The high frequency part of the LF signals (Midband) are parametrically encoded. In the decoder, the parametric model is applied on the mono signal excitation in order to restore the high frequency part of the original LF part of the two channels.

The HF part of the two channels are encoded by using parametric BWE described below.

4.3.6 Low complexity operation

In the low complexity operation (use case B) the decision on the usage of ACELP and TCX mode is done in an open-loop manner. This approach introduces computational savings in the encoder.

4.3.7 Frame erasure concealment

When missing packets occur at the receiver, the decoder applies concealment. The concealment algorithm depends on the mode of the correctly received packets preceding and following the missing packet. Concealment uses either time-domain coefficient extrapolation, as in AMR-WB, or frequency-domain interpolation for some of the TCX modes.

4.3.8 Bit allocation

The bit allocation for the different parameters in the low-frequency band coding (Core) (0-Fs/4 kHz band) is shown in Tables 3, 4, 5, and 6. Note that there are two mode bits sent in each 256-sample packet. These mode bits are not shown in the bit allocation tables. The bit allocations for the stereo part is shown in Tables 7, 8, and 9. Note that there are also two additional mode bits for the VLF stereo encoder, which are not shown in the bit allocation. The bit allocation for the stereo HF part is by definition that of the bandwidth extension, as presented in Tables 7,8 and 9.

Tables 2 and 3 show the total bits per 256-sample packet, including mode bits.

Table 3: Bit allocations for ACELP core rates including BWE (per frame)

Parameter	Number of bits							
Mode bits	2							
ISF Parameters	46							
Mean Energy	2							
Pitch Lag	30							
Pitch Filter	4 × 1							
Fixed-codebook Indices	4 × 20	4 × 28	4 × 36	4 × 44	4 × 52	4 × 64	4 × 72	4 × 88
Codebook Gains	4 × 7							
HF ISF Parameters	9							
HF gain	7							
Total in bits	208	240	272	304	336	384	416	480

Table 4: Bit allocations for 256-sample TCX window (Core)

Parameter	Number of bits							
Mode bits	2							
ISF Parameters	46							
Noise factor	3							
Global Gain	7							
Algebraic VQ	134	166	198	230	262	310	342	406
HF ISF Parameters	9							
HF gain	7							
Total in bits	208	240	272	304	336	384	416	480

Table 5: Bit allocations for 512-sample TCX window (Core)

Parameter	Number of bits							
Mode bits	2+2							
ISF Parameters	46							
Noise factor	3							
Global Gain	7							
Gain redundancy	6							
Algebraic VQ	318	382	446	510	574	670	734	862
HF ISF Parameters	9							
HF gain	7							
HF Gain correction	8 × 2							
Total in bits	416	480	544	608	672	768	832	960

Table 6: Bit allocations for 1024-sample TCX window (Core)

Parameter	Number of bits							
Mode bits	2+2+2+2							
ISF Parameters	46							
Noise factor	3							
Global Gain	7							
Gain redundancy	3+3+3							
Algebraic VQ	695	823	951	1079	1207	1399	1527	1783
HF ISF Parameters	9							
HF gain	7							
HF Gain correction	16 × 3							
Total in bits	832	960	1088	1216	1344	1536	1664	1920

Table 7 Bit allocations for stereo encoder for 256-sample window

Parameter	Number of bits															
Mode bits	2															
Global Gain	7															
Gain	7															
Unused bits	1															
Midband	6							12								
Algebraic VQ	1	9	17	25	33	41	49	51	59	67	75	83	91	99	107	115
HF ISF Parameters	9															
HF gain	7															
Total in bits	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160

Table 8 Bit allocations for stereo encoder for 512-sample window

Parameter	Number of bits															
Mode bits	2+2															
Global Gain	7															
Gain	7															
Unused bits	1+1															
Midband	6×2							12×2								
Algebraic VQ	16	32	48	64	80	96	112	116	132	148	164	180	196	212	228	244
HF ISF Parameters	9															
HF gain	7															
HF Gain correction	8 × 2															
Total in bits	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320

Table 9 Bit allocations for stereo encoder for 1024-sample window

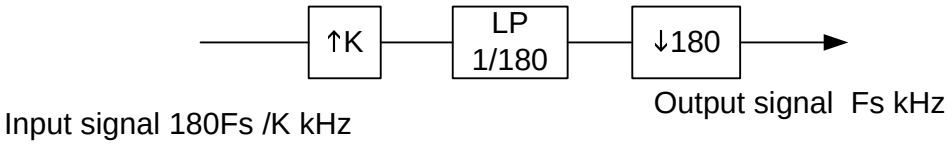
Parameter	Number of bits															
Mode bits	2+2+2+2															
Global Gain	7															
Gain	7															
Unused bits	1+1+1+1															
Midband	6×4							12×4								
Algebraic VQ	46	78	110	142	174	206	238	246	278	310	342	374	406	438	470	502
HF ISF Parameters	9															
HF gain	7															
HF Gain correction	16 × 3															
Total in bits	160	192	224	256	288	320	352	384	416	448	480	512	544	576	608	640

5 Functional description of the encoder

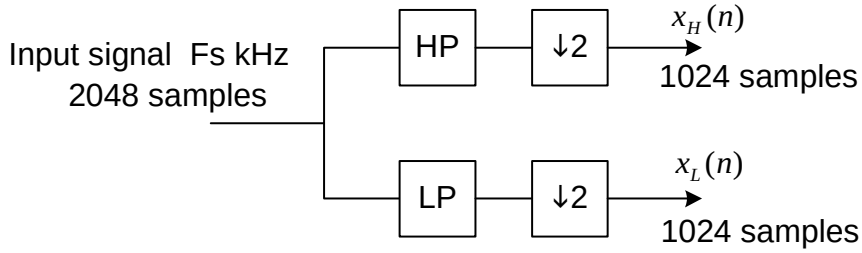
In this clause, the different functions of the encoder extension modes represented in Figure 1 are described. Input signals are understood as internal, i.e. sampled at the internal sampling frequency *Fs*.

5.1 Input signal pre-processing

Input signals are pre-processed in order to bring them to the internal sampling frequency of the encoder *Fs kHz*. The signal is upsampled by a factor *K* (related to the desired internal sampling frequency), filtered by a a low pass filter and then downsampled by a factor 180. This operation is efficiently implemented by a polyphase filter implementation.



The resulting signals are further decomposed into two equal critically sampled bands as shown in the following figure:



At an internal sampling rate of *Fs kHz*, the lower band signals are obtained by first low-pass filtering to *Fs*/4 kHz critically downsampling the low-pass filtered signal to Fs/2 kHz. The higher band signals are obtained by band-pass filtering the input signals to frequencies above Fs/4 kHz, and critically downsampling the high-pass filtered signal to Fs/2kHz sampling frequency.

5.1.1 High Pass Filtering

The lower band signals are high pass filtered. The high-pass filter serves as a precaution against undesired low frequency components. A high pass filter is used, and it is given by

$$H_{h1}(z) = \frac{\mathrm{b}_0 - \mathrm{b}_1 z^{-1} + \mathrm{b}_2 z^{-2}}{1 - \mathrm{a}_1 z^{-1} + \mathrm{a}_2 z^{-2}}.$$

where the filter parameters are dependent on the internal sampling rate.

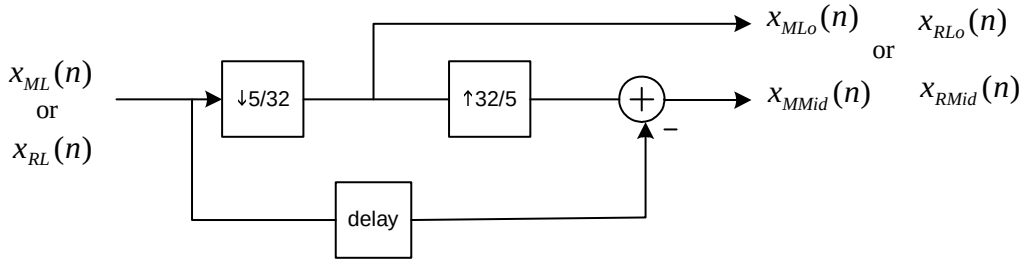
5.1.2 Stereo Signal Downmixing/Bandsplitting

When the input audio signal is stereo, the lower band mono signal is obtained by downmixing the left and right channels according to the following

$$x_{ML}(n) = 0.5\left(x_{LL}(n) + x_{RL}(n)\right)$$

where $x_{LL}(n)$, resp. $x_{RL}(n)$, is the lower band signal from the left, resp. right, channels. The lower band mono signal is supplied to the core low band encoder for TCX/ACELP encoding.

For stereo encoding, the obtained downmixed mono signal $x_{ML}(n)$ and the right channel signal $x_{RL}(n)$ are further split into two bands: a critically sampled low frequency band and a residual high frequency band according to the following diagram



The critically sampled low band output signals, $x_{MLo}(n)$ and $x_{RLo}(n)$ are fed to the stereo low band encoder, while the signals $x_{MMid}(n)$ and $x_{RMid}(n)$ to the stereo mid band encoder.

5.2 Principle of the hybrid ACELP/TCX core encoding

The encoding algorithm at the core of the AMR-WB+ codec is based on a hybrid ACELP/TCX model. For every block of input signal, the encoder decides (either in open-loop or closed-loop) which encoding model (ACELP or TCX) is best. The ACELP model is a time-domain, predictive encoder, best suited for speech and transient signals. The AMR-WB encoder is used in ACELP modes. Alternatively, the TCX model is a transform-based encoder, and is more appropriate for typical music samples. Frame lengths of variable sizes are possible in TCX mode, as will be explained in Section 5.2.1.

In Sections 5.2.1 to 5.2.4, the general principles of the hybrid ACELP/TCX core encoder will be presented. Then Section 5.3 and its subsections will give the details of the ACELP and TCX encoding modes.

5.2.1 Timing chart of the ACELP and TCX modes

The ACELP/TCX core encoder takes a mono signal as input, at a sampling frequency of Fs/2 kHz. This signal is processed in super-frames of 1024 samples in duration. Within each 1024-sample super-frame, several encoding modes are possible, depending on the signal structure. These modes are: 256-sample ACELP, 256-sample TCX, 512-sample TCX and 1024-sample TCX. These encoding modes will be described further, but first we look at the different possible mode combinations, described by a timing chart.

Figure 4 shows the timing chart of all possible modes within an 1024-sample superframe. As the figure shows, each 256-sample frame within a super-frame can be into one of four possible modes, which we call ACELP, TCX256, TCX512 and TCX1024. When in ACELP mode, the corresponding 256-sample frame is encoded with AMR-WB. In TCX256 mode, the frame is encoded using TCX with a 256-sample support, plus 32 samples of look-ahead used for overlap-and add since TCX is a transform coding approach. The TCX512 mode means that two consecutive 256-sample frames are grouped to be encoded as a single 512-sample block, using TCX with a 512-sample support plus 64 samples look-ahead. Note that the TCX512 mode is only allowed by grouping either the first two 256-sample frames of the super-frame, or the last two 256-sample frames. Finally, the TCX1024 mode indicates that all 256-sample frames within the super-frame are grouped together to be encoded in a single block using TCX with an 1024-sample support plus 128 samples look-ahead.

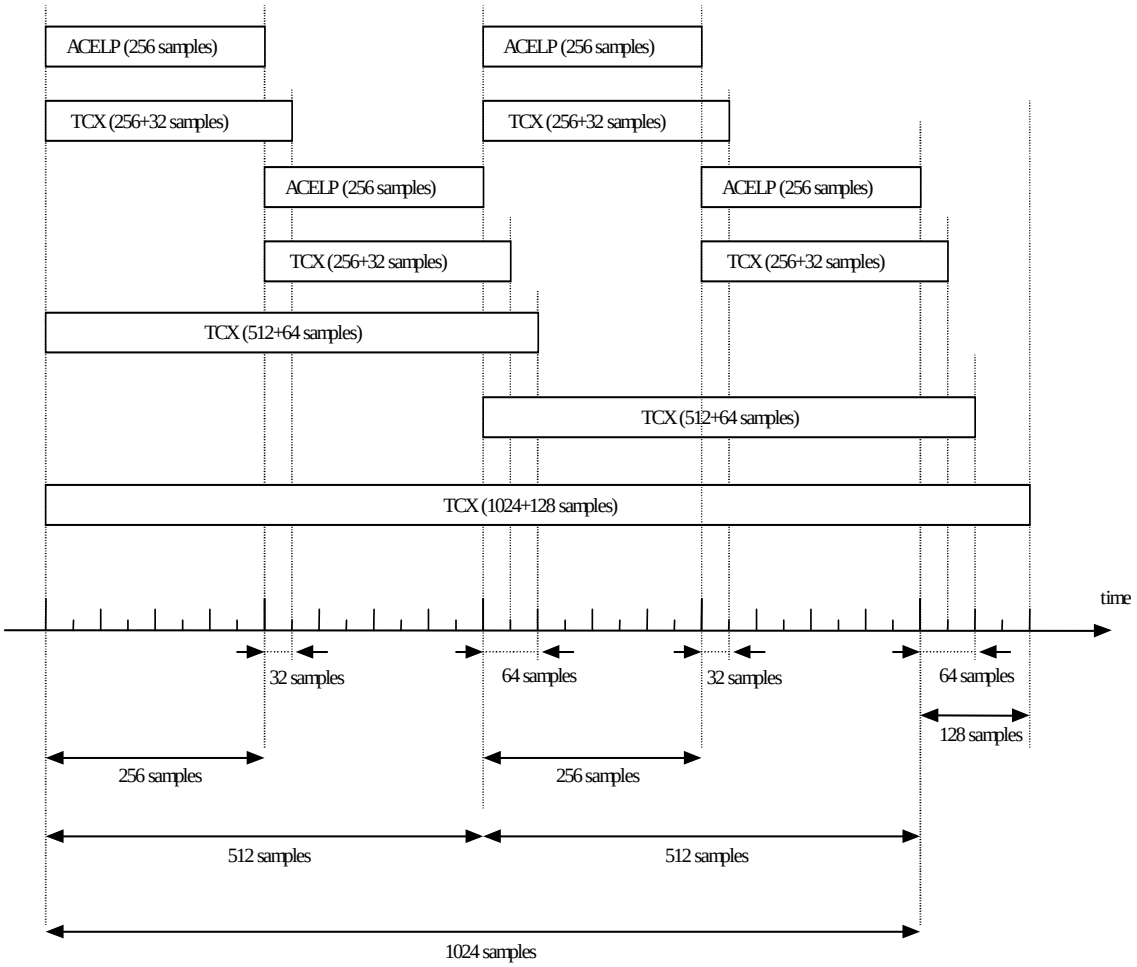


Figure 4: Timing chart of the frame types

5.2.2 ACELP/TCX mode combinations and mode encoding

From Figure 4, there are exactly 26 different ACELP/TCX mode combinations within an 1024-sample superframe. These are shown in Table10.

Table 10: Possible mode combinations in an 1024-sample super-frame

(0, 0, 0, 0)	(0, 0, 0, 1)	(2, 2, 0, 0)	
(1, 0, 0, 0)	(1, 0, 0, 1)	(2, 2, 1, 0)	
(0, 1, 0, 0)	(0, 1, 0, 1)	(2, 2, 0, 1)	
(1, 1, 0, 0)	(1, 1, 0, 1)	(2, 2, 1, 1)	
(0, 0, 1, 0)	(0, 0, 1, 1)	(0, 0, 2, 2)	
(1, 0, 1, 0)	(1, 0, 1, 1)	(1, 0, 2, 2)	
(0, 1, 1, 0)	(0, 1, 1, 1)	(0, 1, 2, 2)	(2, 2, 2, 2)
(1, 1, 1, 0)	(1, 1, 1, 1)	(1, 1, 2, 2)	(3, 3, 3, 3)

We interpret each quadruplet of numbers (m_0, m_1, m_2, m_3) in Table 10 as follows: m_k is the mode indication for the k^{th} 256-sample frame in the 1024-sample super-frame, where m_k can take the following values:

- m_k = 0 means the mode for frame k is 256-sample ACELP
- m_k = 1 means the mode for frame k is 256-sample TCX
- m_k = 2 means the mode for frame k is 512-sample TCX
- m_k = 3 means the mode for frame k is 1024-sample TCX

Obviously, when the first 256-sample frame is in mode "2" (512-sample TCX), the second 256-sample frame must also be in mode 2. Similarly, when the third 256-sample frame is in mode "2" (512-sample TCX), the fourth 256-sample frame must also be in mode 2. And there is only one possible mode configuration including the value "3" (1024-sample TCX), namely all four 256-sample frames are in the same mode ($m_k = 3$ for $k = 0, 1, 2$ and 3). This rigid frame structure can be exploited to aid in frame erasure concealment.

As discussed above, the parameters for each 1024-sample super-frame are actually decomposed into four frames of identical size. To increase robustness, the mode bits are actually sent as two bits (the values of m_k) in each transmitted frame. For example, if the superframe is encoded in a full 1024-sample TCX frame, which is then decomposed into four packets of equal size, then each of these four packets will contain the binary value "11" (mode $m_k = 3$) as mode indicator.

5.2.3 ACELP/TCX closed-loop mode selection

The best mode combination out of the 26 possible combinations of Table 10 is determined in closed-loop. This means that the signal in each 256-sample frame within an 1024-sample super-frame has to be encoded in several modes before selecting the best combination. This closed-loop approach is explained in Figure 5.

The left portion of Figure 5 (Trials) shows what encoding mode is applied to each 256-sample frame in 11 successive trials. Fr0 to Fr3 refer to Frame 0 to Frame 3 in the super-frame. The trial number (1 to 11) indicates a step in the closed-loop mode-selection process. Note that each 256-sample frame is involved in only four of the 11 encoding trials. When more than 1 frame is involved in a trial (lines 5, 10 and 11 of Figure 5), then TCX of the corresponding length is applied (TCX512 or TCX1024). The right portion of Figure 5 gives an example of mode selection, where the final decision (after Trial 11) is 1024-sample TCX. This would result in sending a value of 3 for the mode in all four packets for this super-frame. Bold numbers in the example at the right of Figure 5 show at what point a mode decision is taken in the intermediate steps of the mode selection process. The final mode decision is only known after Trial 11.

The mode selection process shown in Figure 5 proceeds as follows. First, in trials 1 and 2, ACELP (AMR-WB) then 256-sample TCX encoding are tried in the first 256-sample frame (Fr0). Then, a mode selection is made for Fr0 between these two modes. The selection criterion is the average segmental SNR between the weighted speech $x_w(n)$ and the synthesized weighted speech $\hat{x}_w(n)$. The segmental SNR in subframe i is defined as

$$segSNR_i = 20\log_{10}\left(\frac{\sum_{n=0}^{N-1} x_w^2(n)}{\sum_{n=0}^{N-1}\left(x_w(n)-\hat{x}_w(n)\right)^2}\right)$$

where N is the length of the subframe (equivalent to a 64-sample sub-frame in the encoder). Then, the average segmental SNR is defined as

$$\overline{segSNR} = \frac{1}{N_{SF}}\sum_{i=0}^{N_{SF}-1} segSNR_i$$

where N_{SF} is the number of subframes in the frame. Since a frame can be either 256, 512 or 1024 samples in length, N_{SF} can be either 4, 8 or 16. In the example of Figure 5, we assume that, according to the $\overline{segSNR}$ decision criterion, mode ACELP was retained over TCX. Then, in trials 3 and 4, the same mode comparison is made for Fr1 between ACELP and 256-sample TCX. Here, we assume that 256-sample TCX was better than ACELP, based again on the segmental SNR measure described above. This choice is indicated in bold on line 4 of the example at the right of Figure 5. Then, in trial 5, Fr0 and Fr1 are grouped together to form a 512-sample frame which is encoded using 512-sample TCX. The algorithm now has to choose between 512-sample TCX for the first 2 frames, compared to ACELP in the first frame and TCX256 in the second frame. In this example, on line 5 in bold, the sequence ACELP-TCX256 was selected over TCX-512, according to the segmental SNR criterion.

	TRIALS (11)				Example of selection (in bold = comparison is made)			
	Fr 0	**Fr 1**	**Fr 2**	**Fr 3**	**Fr 0**	**Fr 1**	**Fr 2**	**Fr 3**
1	ACELP				ACELP			
2	TCX256				**ACELP**			
3		ACELP			ACELP	ACELP		
4		TCX256			ACELP	**TCX256**		
5	TCX512	TCX512			**ACELP**	**TCX256**		
6			ACELP		ACELP	TCX256	ACELP	
7			TCX256		ACELP	TCX256	**TCX256**	
8				ACELP	ACELP	TCX256	TCX256	ACELP
9				TCX256	ACELP	TCX256	TCX256	**TCX256**
10			TCX512	TCX512	ACELP	TCX256	**TCX512**	**TCX512**
11	TCX1024	TCX1024	TCX1024	TCX1024	**TCX1024**	**TCX1024x**	**TCX1024**	**TCX1024**

Figure 5: Closed-loop selection of ACELP/TCX mode combination

The same procedure as trials 1 to 5 is then applied to the third and fourth frames (Fr2 and Fr3), in trials 6 to 10. After trial 10, in the example of Figure 5, the four 256-sample frames are classified as: ACELP for F0, then TCX256 for F1, then TCX512 for F2 and F3 grouped together. A last trial (line 11) is then performed where all four 256-sample frames (the whole super-frame) are encoded with 1024-sample TCX. Using the segmental SNR criterion, again with 64-sample segments, this is compared with the signal encoded using the mode selection in trial 10. In this example, the final mode decision is 1024-sample TCX for the whole frame. The mode bits for each 256-sample frame would then be (3, 3, 3, 3) as discussed in Table10.

5.2.4 ACELP/TCX open-loop mode selection

The alternative method for ACELP/TCX mode selection is the low complexity open-loop method. The open-loop mode selection is divided into three selection stages: Excitation classification (EC), excitation classification refinement (ECR) and TCX selection (TCXS). The mode selection is done purely open-loop manner in EC and ECR. The usage of TCXS algorithm depends on EC and ECR and it is closed loop TCX mode selection.

1. stage

The first stage excitation classification is done before LP analysis. The EC algorithm is based on the frequency content of the input signal using the VAD algorithm filter bank.

AMR-WB VAD produces signal energy $\boldsymbol{E(n)}$ in the 12 non-uniform bands over the frequency range from 0 to Fs/4 kHz for every 256-sample frame. Then energy levels of each band are normalised by dividing the energy level $\boldsymbol{E(n)}$ from each band by the width of that band in Hz producing normalised $\boldsymbol{E_N(n)}$ energy levels of each band where n is the band number from 0 to 11. Index 0 refers to the lowest sub band.

For each of the 12 bands, the standard deviation of the energy levels is calculated using two windows: a short window $\boldsymbol{std_{short}(n)}$ and a long window $\boldsymbol{std_{long}(n)}$. The length of the short and long window is 4 and 16 frames, respectively. In these calculations, the 12 energy levels from the current frame together with past 3 or 15 frames are used to derive two $\boldsymbol{stda_{short}}$ and $\boldsymbol{stda_{long}}$ standard deviation values. The standard deviation calculation is performed only when VAD indicates active signal.

The relation between lower frequency bands and higher frequency bands are calculated in each frame. The energy of lower frequency bands ***LevL*** from 1 to 7 are normalised by dividing it by the length of these bands in Hz. The higher frequency bands 8 to 11 are normalised respectively to create ***LevH***. Note that the lowest band 0 is not used in these calculations because it usually contains so much energy that it will distort the calculations and make the contributions from other bands too small. From these measurements the relation ***LPH = LevL / LevH*** is defined. In addition, for each frame a moving average ***LPHa*** is calculated using the current and 3 past ***LPH*** values. The final measurement of the low

and high frequency relation ***LPHaF*** for the current frame is calculated by using weighted sum of the current and 7 past ***LPHa*** values by setting slightly more weighting for the latest values.

The average level (***AVL***) in the current frame is calculated by subtracting the estimated level of background noise from each filter bank level after which the filter bank levels are normalised to balance the high frequency bands containing relatively less energy than the lower bands. In addition, total energy of the current frame, $TotE_0$, is derived from all the filter banks subtracted by background noise estimate of the each filter bank. Total energy of previous frame is therefore $TotE_{-1}$.

After calculating these measurements, a choice between ACELP and TCX excitation is made by using the following pseudo-code:

```
if (stda_long < 0.4)
  SET TCX_MODE
else if (LPHaF > 280)
  SET TCX_MODE
else if (stda_long >= 0.4)
  if ((5+(1/( stda_long -0.4))) > LPHaF)
     SET TCX_MODE
  else if ((-90* stda_long +120) < LPHaF)
     SET ACELP_MODE
  else
     SET UNCERTAIN_MODE

if (ACELP_MODE or UNCERTAIN_MODE) and (AVL > 2000)
  SET TCX_MODE
if (UNCERTAIN_MODE)
  if (stda_short < 0.2)
     SET TCX_MODE
  else if (stda_short >= 0.2)
     if ((2.5+(1/( stda_short -0.2))) > LPHaF)
        SET TCX_MODE
     else if ((-90* stda_short +140) < LPHaF)
        SET ACELP_MODE
     else
        SET UNCERTAIN_MODE
if (UNCERTAIN_MODE)
  if ((TotE_0 / TotE_-1 )>25)
     SET ACELP_MODE

if (TCX_MODE || UNCERTAIN_MODE))
  if (AVL > 2000 and TotE_0 < 60)
     SET ACELP_MODE
```

2. stage

ECR is done after open-loop LTP anlysis.

If VAD flag is set and mode has been classified in EC algorithm as uncertain mode (defined as TCX_OR_ACELP), the is mode is selected as follows:

```
if (SD_n > 0.2)
  Mode = ACELP_MODE;
else
  if (LagDif_buf < 2 )
     if (Lag_n == HIGH LIMIT or Lag_n == LOW LIMIT){
        if (Gain_n-NormCorr_n<0.1 and NormCorr_n>0.9)
```

```
            Mode = ACELP_MODE
          else
            Mode = TCX_MODE
      else if (Gain_n - NormCorr_n < 0.1 and NormCorr_n > 0.88)
        Mode = ACELP_MODE
      else if (Gain_n – NormCorr_n > 0.2)
        Mode = TCX_MODE
      else
        NoMtcx = NoMtcx +1
    if (MaxEnergy_buf < 60 )
      if (SD_n > 0.15)
        Mode = ACELP_MODE;
      else
        NoMtcx = NoMtcx +1.
```

Where spectral distance, SD_n, of the frame *n* is calculated from ISP parameters as follows:

$$SD(n) = \sum_{i=0}^{N} \left| ISP_n(i) - ISP_{n-1}(i) \right|,$$

where ISP_n is the ISP coefficients vector of the frame *n* and $ISP_n(i)$ is *i*th element of it.

$LagDif_{buf}$ is the buffer containing open loop lag values of previous ten frames (256 samples).

Lag_n contains two open loop lag values of the current frame *n*.

$Gain_n$ contains two LTP gain values of the current frame *n*.

$NormCorr_n$ contains two normalised correlation values of the current frame *n*.

$MaxEnergy_{buf}$ is the maximum value of the buffer containing energy values. The energy buffer contains last six values of current and previous frames (256 samples).

lph_n indicates the spectral tilt.

If VAD flag is set and mode has been classified in EC algorithm as ACELP mode, the mode decision is verified according to following algorithm where mode can be switched to TCX mode.

```
    if (LagDif_buf < 2)
      if (NormCorr_n < 0.80 and SD_n < 0.1)
        Mode = TCX_MODE;
    if (lph_n > 200 and SD_n < 0.1)
      Mode = TCX_MODE
```

If VAD flag is set in current frame and VAD flag has set to zero at least one of frames in previous super-frame and the mode has been selected as TCX mode, the usage of TCX1024 is disabled (the flag ***NoMtcx*** is set).

```
    if (vadFlag_old == 0 and vadFlag == 1 and Mode == TCX_MODE))
      NoMtcx = NoMtcx +1
```

If VAD flag is set and mode has been classified as uncertain mode (TCX_OR_ACELP) or TCX mode, the mode decision is verified according to following algorithm.

```
    if (Gain_n - NormCorr_n < 0.006 and NormCorr_n > 0.92 and Lag_n > 21)
      DFTSum = 0;
      for (i=1; i<40; i++)
        DFTSum = DFTSum  + mag[i];
      if (DFTSum > 95 and mag[0] < 5)
        Mode = TCX_MODE;
      else
```

Mode = ACELP_MODE;
NoMtcx = ***NoMtcx*** +1

$vadFlag_{old}$ is the VAD flag of the previous frame and ***vadFlag*** is the VAD flag of the current frame.

NoMtcx is the flag indicating to avoid TCX transformation with long frame length (1024 samples), if TCX coding model is selected.

Mag is a discete Fourier transformed (DFT) spectral envelope created from LP filter coefficients, ***Ap,*** of the current frame. ***DFTSum*** is the sum of first 40 elements of the vector ***mag*** , excluding the first element (***mag(0)***) of the vector ***mag***.

If VAD flag is set and the mode, ***Mode(Index)***, of the ***Index***th frame of current superframe has still been classified as uncertain mode (TCX_OR_ACELP), the mode is decided based on selected modes in the previous and current superframes. The counter, ***TCXCount***, gives the number of selected long TCX frames (TCX512 and TCX1024) in previous superframe (1024 samples). The counter, ***ACELPCount***, gives the number of ACELP frames (256 samples) in previous and current superframes.

if ((***prevMode(i)*** == TCX1024 or ***prevMode(i)*** == TCX512) and $vadFlag_{old}(i)$== 1 and $TotE_i$ > 60)
  TCXCount = ***TCXCount*** + 1
if (***prevMode(i)*** == ACELP_MODE)
  ACELPCount = ***ACELPCount*** + 1
if (***Index*** != ***i***)
  if (***Mode(i)*** == ACELP_MODE)
    ACELPCount = ***ACELPCount*** + 1

Where ***prevMode(i)*** is the ***i****th* frame (256 samples) in the previous superframe, ***Mode(i)*** is the ***i****th* frame in the current superframe. ***i*** is the frame (256 samples) number in superframe (1, 2, 3, 4), The mode, ***Mode(Index)***, is selected based on the counters ***TCXCount*** and ***ACELPCount*** as follows

if (***TCXCount*** > 3)
  Mode(Index) = TCX_MODE;
else if (***ACELPCount*** > 1)
  Mode(Index) = ACELP_MODE
else
  Mode(Index) = TCX_MODE

3. stage: TCXS is done only if the number of ACELP modes selected in EC and ECR is less than three (ACELP<3) within an 1024-sample super-frame. The Table11 shows the possible mode combination which can be selected in TCXS. TCX mode is selected according to segmental SNR described in Chapter 5.2.3 (ACELP/TCX closed-loop mode selection).

Table 11: Possible mode combination selected in TCXS

Selected mode combination after open-loop mode selection (TCX = 1 and ACELP = 0)	Possible mode combination after TCXS (ACELP = 0, TCX256 = 1, TCX512 = 2 and TCX1024 = 3)		
			NoMTcx
(0, 1, 1, 1)	(0, 1, 1, 1)	(0, 1, 2, 2)	
(1, 0, 1, 1)	(1, 0, 1, 1)	(1, 0, 2, 2)	
(1, 1, 0, 1)	(1, 1, 0, 1)	(2, 2, 0, 1)	
(1, 1, 1, 0)	(1, 1, 1, 0)	(2, 2, 1, 0)	
(1, 1, 0, 0)	(1, 1, 0, 0)	(2, 2, 0, 0)	
(0, 0, 1, 1)	(0, 0, 1, 1)	(0, 0, 2, 2)	
(1, 1, 1, 1)	(1, 1, 1, 1)	(2, 2, 2, 2)	1
(1, 1, 1, 1)	(2, 2, 2, 2)	(3, 3, 3, 3)	0

5.3 Hybrid ACELP/TCX core encoding description

5.3.1 Pre-emphasis

The input (mono) signal to the core ACELP/TCX encoder is first pre-processed through a high-pass filter. Then, a first-order, fixed pre-emphasis filter is applied with transfer function:

$$P(z) = 1 - 0.68z^{-1}$$

This pre-emphasis filter reduces the signal energy at low frequency and increases the signal energy at high frequency. The result is a signal with a less spectral dynamics, which enhances the resolution of LPC analysis. This is particularly important in fixed-point implementations.

5.3.2 LP analysis and interpolation

The principles are similar to 3GPP TS 26.190 (Section 5.2) except for the window shape and position, and the interpolation factors.

LPC analysis is performed every 256 samples in the superframe. The analysis window has 448 samples in duration, and has the shape of a half-sine cycle. It is symmetrical, and centred, for each LP analysis, at the middle of the first 64-sample sub-frame following each 256-sample frame. Hence, the lookahead required for LP analysis is half the analysis window length plus 32 samples, for a total of 256 samples.

The autocorrelations of windowed speech are converted to the LP coefficients using the Levinson-Durbin algorithm. Then the LP coefficients are transformed to the ISP domain for quantization and interpolation purposes. The interpolated quantized and unquantized filters are converted back to the LP filter coefficients (to construct the synthesis and weighting filters at each subframe).

The LP coefficients are quantized in the ISF domain, and interpolated every 64 samples. Depending on the coding mode selected (ACELP or TCX), the LP coefficients are transmitted at different update rates. For ACELP, which is based on the AMR-WB encoder, the LP coefficients are transmitted every 256 samples. For TCX, the LPC coefficients are transmitted every 256, 512 or 1024 samples, depending on the TCX frame length.

The LPC analysis is identical for ACELP or TCX.

5.3.2.1 Windowing and auto-correlation computation

A 448-sample symmetrical sine window is used. The window is given by

$$w(n) = \sin\left(\frac{\pi}{L}(i + 0.5)\right), \qquad i = 0,..., L-1$$

where L=488 is the window length.

Autocorrelation computation and lag windowing are similar to 3GPP TS 26.190 (Section 5.2.1) with the exception that the window length is 488 samples and centred at the middle of the first subframe of the next 256-sample frame.

5.3.2.2 Levinson-Durbin algorithm

Same as 3GPP TS 26.190.

5.3.2.3 LP to ISP conversion

Same as 3GPP TS 26.190.

5.3.2.4 ISP to LP conversion

Same as 3GPP TS 26.190.

5.3.2.5 Quantization of the ISP coefficient

Same as 3GPP TS 26.190. The difference is that the in case of TCX, the LP parameters are quantized and transmitted once per TCX frame (256, 512, or 1024 samples).

5.3.2.6 Interpolation of the ISPs

Since the LP analysis window shape and position is changed compared to AMR-WB, the interpolation factors have changed.

The interpolation is performed on the ISPs in the $\mathbf{q}$ domain (cosine domaine). Let $\mathbf{q}^{(n)}$ be the ISP vector from LP analysis at frame *n* (centred at the 1st subframe of the frame *n*+1), and $\mathbf{q}^{(n-1)}$ the ISP vector from LP analysis at frame *n*-1 (centred at the 1st subframe of the frame n). The interpolated ISP vectors in each subframe are given by

$$\mathbf{q}_i^{(n)} = (1-\frac{i}{4})\mathbf{q}^{(n-1)} + \frac{i}{4}\mathbf{q}^{(n)}, \qquad i = 0,...,3.$$

The interpolated ISP vectors are used to compute a different LP filter at each subframe using the ISP to LP conversion method.

The interpolation is performed for both unquantized and quantized parameters. In case TCX frames, the LP parameters are quantized once per 256-sample, 512-sample, or 1024-sample TCX frame, depending on the selected TCX mode. Thus the interpolation of quantized ISPs is performed as follows. In case of ACELP or 256-sample TCX frames, the interpolated quantized parameters are given by

$$\hat{\mathbf{q}}_i^{(n)} = (1-\frac{i}{4})\hat{\mathbf{q}}^{(n-1)} + \frac{i}{4}\hat{\mathbf{q}}^{(n)}, \qquad i = 0,...,3.$$

In case of 512-sample TCX frames, comprised of 256-sample frames *n* and *n*+1, the interpolated quantized parameters are given by

$$\hat{\mathbf{q}}_i^{(n,n+1)} = (1-\frac{i}{8})\hat{\mathbf{q}}^{(n-1)} + \frac{i}{8}\hat{\mathbf{q}}^{(n+1)}, \qquad i = 0,...,7.$$

In case of 1024-sample TCX frames, comprised of 256-sample frames *n*, *n*+1, n+2, and n+3, the interpolated quantized parameters are given by

$$\hat{\mathbf{q}}_i^{(n,n+3)} = (1-\frac{i}{16})\hat{\mathbf{q}}^{(n-1)} + \frac{i}{16}\hat{\mathbf{q}}^{(n+3)}, \qquad i = 0,...,15.$$

5.3.3 Perceptual weighting

Same as 3GPP TS 26.190, using a filter of the form $W(z) = A(z/\gamma_1)H_{de-emph}(z)$. Note that in case of TCX coding, quantized filter coefficients are used at the numerator, so that the filter is given by $W(z) = \hat{A}(z/\gamma_1)H_{de-emph}(z)$.

5.3.4 ACELP Excitation encoder

5.3.4.1 Open-loop pitch analysis

Same as 3GPP TS 26.190 but with the minimum and maximum pitch lags as functions of the internal sampling rate.

5.3.4.2 Impulse response computation

Same as 3GPP TS 26.190.

5.3.4.3 Target signal computation

Same as 3GPP TS 26.190

5.3.4.4 Adaptive codebook

Same as 3GPP TS 26.190 but with the minimum and maximum pitch lags as functions of the internal sampling rate.

5.3.4.5 Algebraic codebook

Same as 3GPP TS 26.190. The codebooks used in some basic core rates are the same as used in AMR-WB at 12.65, 14.25, 15.85, 18.25, 19.85, and 23.05, respectively. The codebook used in mono rate 208 bits/frame is the same as the one used in AMR-WB at 8.85 kbps (20 bit codebook). The codebook used in mono rate 240 bits/frame is a 28 bit codebook where two tracks contain one pulse each and two tracks contain two pulses each according to the following table:

Table 12: Potential positions of individual pulses in the 28-bit algebraic codebook

Track	Pulse	Positions
1	i_0, i_4	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	i_1,	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	i_2, i_5	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	i_3,	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

5.3.4.5.1 Codebook structure

Same as 3GPP TS 26.190.

5.3.4.5.2 Pulse indexing

Same as 3GPP TS 26.190.

5.3.4.5.3 Codebook search

Same as 3GPP TS 26.190.

5.3.4.6 Quantization of the adaptive and fixed codebook gains

The adaptive codebook gain (pitch gain) and the fixed (algebraic) codebook gain are vector quantized using the same 7-bit codebook used in AMR-WB for modes 2 to 8. However, instead of using MA prediction to obtain the predicted gain g'_c, it is found by directly quantizing the average innovation energy in the whole frame.

Let $E_s(n)$ be the innovation energy (in dB) at subframe n, and given by

$$E_s(n) = 10\log\left(\frac{1}{N} g_c^2 \sum_{i=0}^{N-1} c^2(i)\right) = 20\log(g_c) + E_i$$

where N=64 is the subframe size, $c(i)$ is the fixed codebook excitation, and E_i is the un-scaled innovation energy given by

$$E_i = \frac{1}{N}\sum_{i-0}^{N-1} c^2(i)$$

An estimated innovation energy $\overline{E}_s$ is computed and quantized, and used to find the estimated gain g'_c. That is,

$$g'_c = 10^{0.05(\overline{E}_s - E_i)}$$

which is derived from the relation $\overline{E}_s = 20\log(g'_c) + E_i$.

A correction factor between the gain g_c and the estimated one g'_c is given by

$$\gamma = g_c / g'_c.$$

The pitch gain, g_p, and correction factor γ are jointly vector quantized using the same 7-bit codebook used in AMR-WB, and using the same error minimization procedure. That is, the gain codebook search is performed by minimizing the mean-square of the weighted error between original and reconstructed signal.

The estimated innovation energy is computed and quantized as follows. First, the LP residual energy is computed in each subframe *n* by

$$E_{res}(n) = 10\log_{10}\left(\frac{1}{N}\sum_{i=0}^{N-1} r^2(j)\right)$$

then the average residual energy per subframe is found by

$$\overline{E}_{res} = \frac{1}{4}\sum_{n=0}^{3} E_{res}(n)$$

The innovation energy is estimated from the residual energy by removing an estimate of the adaptive codebook contribution. This is done by removing an energy related to the average normalized correlation obtained from the two open-loop pitch analyses performed in the frame. That is

$$E_s = \overline{E}_{res} - 10\overline{R}$$

where $\overline{R}$ is the average of the normalized pitch correlations obtain for each half-frame from the open-loop pitch analysis.

The estimated innovation energy is quantized once per frame using 2 bits, with the quantization levels: 18, 30, 42, and 54. Further, the quantized estimated innovation energy $\overline{E}_s$ is constrained to be larger than E_{max}-37, where E_{max} is the maximum value of $E_{res}(n)$ from the 4 subframes. This is done by incrementing $\overline{E}_s$ by 12 (and the quantization index by 1) until $\overline{E}_s > E_{\max} - 37$ or $\overline{E}_s = 54$.

The quantized estimated innovation energy is then used to compute the estimated gain in each subframe is explained above.

5.3.5 TCX Excitation encoder

This section presents the details of the TCX encoder, which is one of the possible modes to encode the mono, low-frequency signal in the 0-Fs/4 kHz band. Section 5.3.5.1 first presents the block diagram of the TCX encoder. Then, the details of each module are given in sections 5.3.5.2 to 5.3.5.13.

5.3.5.1 TCX encoder block diagram

Figure 6 shows a block diagram of the TCX encoding mode. The TCX encoding principle is similar for TCX frames of 256, 512 and 1024 samples, with a few differences mostly involving the windowing and filter interpolation. The input audio signal is first filtered through a time-varying weighting filter (same perceptual filter as in AMR-WB) to obtain a weighted signal ***x***. The weighting filter coefficients are interpolated in the ISP domain as in Section 5.3.2.6. The interpolation is linear, and the beginning and end of the interpolation depend on the refresh rate of the LPC filter. The LPC filter is transmitted only once per TCX frame. For longer frames (TCX512 and TCX1024) the interpolated LPC filters will be farther apart that in the case of TCX256 or ACELP frames.

Continuing in Figure 6, if the past frame was an ACELP frame, the zero-input response (ZIR) of the weighting filter is removed from the weighted signal, using the filter state at the end of the previous (ACELP) frame. The signal is then windowed (the window shape will be described in section 5.3.5.4) and a transform is applied to the windowed signal. In

the transform domain, the signal is first pre-shaped, to minimize coding noise artefact in the low-frequencies, and then quantized using a specific lattice quantizer. Specifically, an 8-dimensional multi-rate lattice quantizer is used, based on an extension of the Gosset lattice.

After quantization, the inverse pre-shaping function is applied to the spectrum which is then inverse transformed to provide a quantized time-domain signal. The gain for that frame is then rescaled to optimize the correlation with the original weighted signal. After gain rescaling, a window is again applied to the quantized signal to minimize the block effects due to quantizing in the transform domain. Overlap-and-add is used with the previous frame if it was also in TCX mode. Finally, the excitation signal is found through inverse filtering with proper filter memory updating. This TCX excitation is in the same "domain" as the ACELP (AMR-WB) excitation.

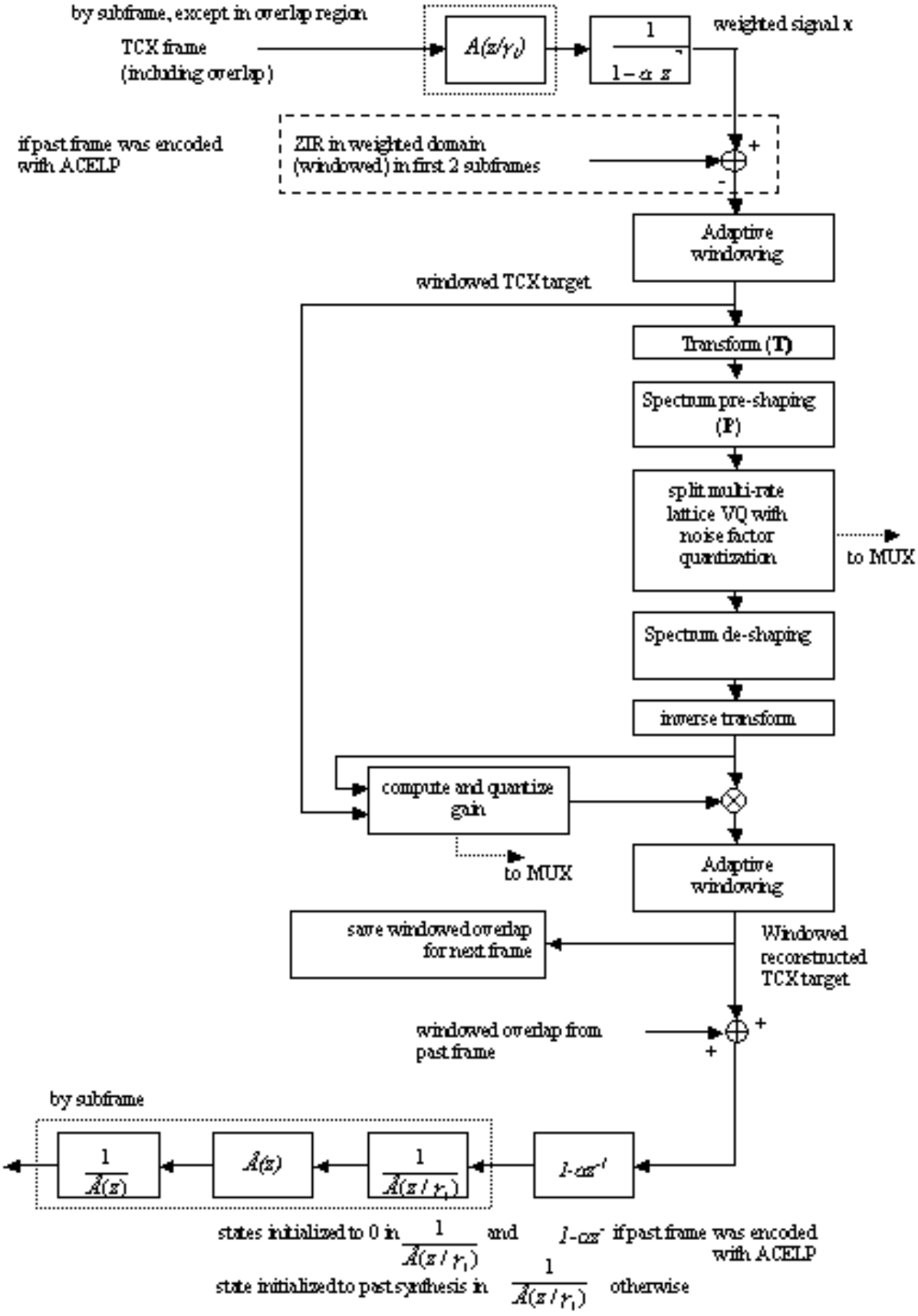


Figure 6: Principle of TCX encoding

Each module of Figure 6 will now be detailed in the following subsections.

5.3.5.2 Computation of the target signal for transform coding

To obtain the weighted signal, the input frame of audio samples is filtered with a perceptual filter having the following transfer function:

$$W(z) = \frac{\hat{A}(z/0.92)}{1-0.68z^{-1}}$$

Here, $\hat{A}(z)$ is the quantized LP filter, interpolated at every 64-sample sub-frame in the ISP domain as in Section 5.3.2.6, and $\hat{A}(z/0.92)$ is the weighted version of that filter. The denominator of $W(z)$ is a constant polynomial of order 1, which is equal to the numerator of the pre-emphasis filter in Section 5.3.1.

5.3.5.3 Zero-input response subtraction

If the previous encoded frame was ACELP, then the zero-input response (ZIR) of the combination of the weighting filter and synthesis filter is removed from the weighted signal. The ZIR is truncated to 128 samples and windowed in such a way that its amplitude monotonically decreases to zero at after 128 samples. The truncated ZIR is computed through the following steps:

Using the filter states at the end of the previous frame, compute the ZIR of the following transfer function over 2 consecutive subframes (128 samples duration):

$$H(z) = \frac{1}{\hat{A}(z)} \frac{\hat{A}(z/0.92)}{1-0.68z^{-1}}$$

where $\hat{A}(z)$ and $\hat{A}(z/0.92)$ are as defined in Section 5.3.5.2.

Then, calling $z(n)$ the truncated ZIR of $H(z)$ (truncated to the first $2N$ samples, where N=64 is the subframe length), compute $z_w(n)$, the windowed ZIR such that it is always forced to zero at the last sample:

$$z_w(n) = z(n)*w(n) \quad \text{for } n = 0 \text{ to } 2*N-1$$

where

$$w(n) = 1 \quad \text{for } n = 0 \text{ to } N-1$$

and $$w(n) = (2*N-n) / N \quad \text{for } n = N \text{ to } 2*N-1$$

The shape of $w(n)$ is shown in Figure 7 below, for a value of N= 64.

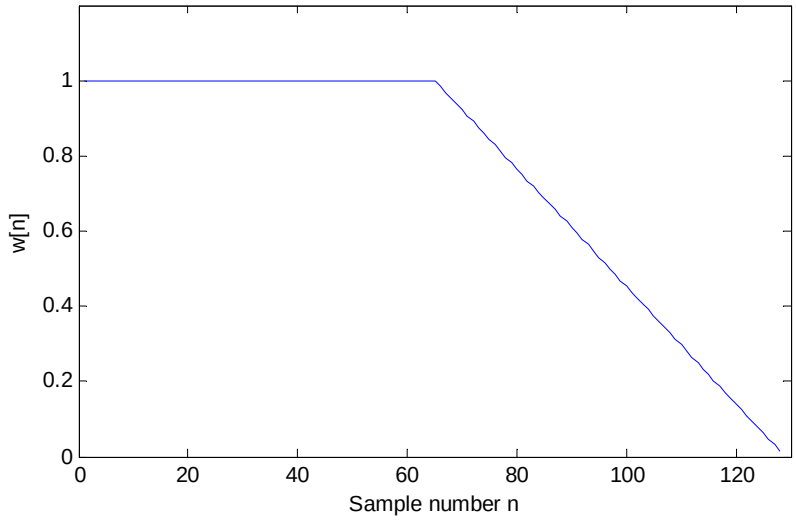


Figure 7: Shape of window to truncate the ZIR

After computing $z_w(n)$, it is removed from the first $2*N$ samples of the weighted signal $x(n)$. This removal of the ZIR from the past frame is performed only when the past frame was in ACELP mode.

5.3.5.4 Windowing of target signal

In TCX mode, windowing in applied prior to the transform, and after the inverse transform, in order to apply overlap-and-add to minimize the framing effects due to quantization.

To smooth the transition between ACELP and TCX modes, proper care has to be given to windowing and overlap of successive frames. Figure 8 shows the window shapes depending on the TCX frame length and the type of the previous frame (ACELP of TCX).

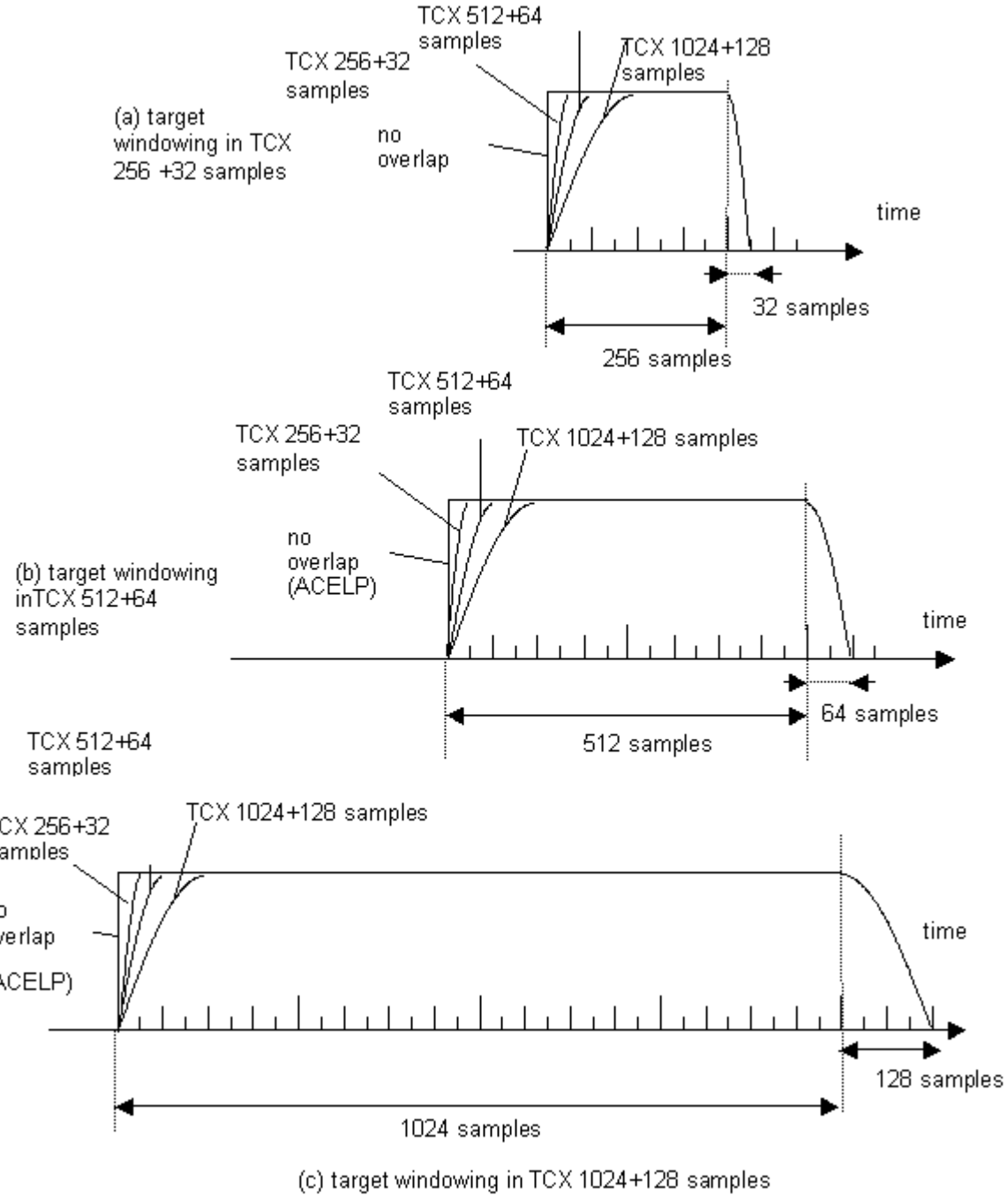


Figure 8: Target signal windowing in TCX coding

The window is defined as the concatenation of the following three sub-windows:

$$w_1(n) = \sin(2 \pi n / (4 L_1)) \qquad \text{for } n = 0, \ldots, L_1\text{-}1$$

$w_2(n)$ = 1 for n = 0, ..., L- L_1-1

$w_3(n)$ = $\sin(2 \pi n / (4 L2))$ for $n = L_2, \ldots, 2L_2$-1

The constants L_1, L_2 and L are defined as follows.

$L_1 = 0$ when the previous frame is a 256-sample ACELP frame

$L_1 = 32$ when the previous frame is a 256-sample TCX frame

$L_1 = 64$ when the previous frame is a 512-sample TCX frame

$L_1 = 128$ when the previous frame is an 1024-sample TCX frame

Additionally:

For 256-sample **TCX**: $L = 256$ and $L_2 = 32$

For 512-sample **TCX**: $L = 512$, and $L_2 = 64$

For 1024-sample **TCX**: $L = 1024$, and $L_2 = 128$ and

We note again that all these window types are applied to the weighted signal, only when the present frame is a TCX frame. Frames of type ACELP are encoded as in AMR-WB encoding (i.e. through analysis-by-synthesis encoding of the excitation signal, so as to minimize the error in the target signal – the target signal is essentially the weighted signal from which the zero-input response of the weighting filter is removed).

5.3.5.5 Transform

After windowing, the signal is mapped to the frequency domain through a Discrete Fourier Transform (DFT), defined as:

$$X(k) = \frac{1}{L_{TOT}} \sum_{n=0}^{LTOT-1} x(n) e^{-j\frac{2\pi}{L_{TOT}}nk}$$

where L_{TOT} is the number of samples in the DFT. L_{TOT} depends on the frame length (256, 512 or 1024 samples, plus the lookahead which is a function of the frame length).

An FFT is used to accelerate the computation of the Fourier coefficients. A radix-9 FFT is used to adapt to the frame length which is not a power of 2. Including the overlap in the windowing described in Section 5.3.5.4, the number of samples at the input of the FFT is, respectively, L_{TOT} = 288 for 256-sample TCX frames (256 samples in the frame plus 32 samples in the look-ahead), L_{TOT} =576 for 512-sample TCX (512 samples in the frame plus 64 samples in the lookahead), and L_{TOT} =1152 samples for 1024-sample TCX (1024 samples in the frame plus 128 samples in the lookahead).

5.3.5.6 Spectrum pre-shaping

Once the Fourier spectrum (FFT) is computed, an adaptive low-frequency emphasis module is applied to the spectrum, to minimize the perceived distortion in the lower frequencies. The inverse low-frequency emphasis will be applied at the decoder, as well as in the encoder to allow obtaining the excitation signal necessary to encode the next frames. The adaptive low-frequency emphasis is applied only on the first quarter of the spectrum, as follows.

First, we call X the transformed signal at the output of the transform (FFT) in Figure 6. The Fourier coefficient at Nyquist frequency is systematically set to 0. Then, if L_{TOT} is the number of samples in the FFT (L_{TOT} is thus the window length), the K= L_{TOT} /2 complex-valued Fourier coefficients are grouped in blocks of four consecutive coefficients, forming 8-dimensional real-valued blocks. This block size of 8 is chosen to coincide with the 8-dimensional lattice quantizer used for spectral quantization. The energy of each block is computed, up to the first quarter of the spectrum. The energy E_{max} and position index I of the block with maximum energy are stored. Then, we calculate a factor for each 8-dimensional block with position index m smaller than I, as follows:

- calculate the energy E_m of the 8-dimensional block at position index m
- compute the ratio $R_m = E_{max} / E_m$

- compute the value $(R_m)^{1/4}$
- if $R_m > 10$, then set $R_m = 10$ (maximum gain of 20 dB)
- also, if $R_m > R_{m-1}$ then $R_m = R_{m-1}$

This last condition ensures that the ratio function R_m decreases monotonically. Further, limiting the ratio R_m to be smaller or equal to 10 means that no spectral components in the low-frequency emphasis function will be modified by more than 20 dB.

After computing the ratio $R_m = (E_{max} / E_m)^{1/4}$ for all blocks with position index smaller that I (and with the limiting conditions described above), we then apply these ratios as a gain for each corresponding block. This has the effect of increasing the energy of blocks with relatively low energy compared to the block with maximum energy E_{max}. Applying this procedure prior to quantization has the effect of shaping the coding noise in the lower band, such that low energy components before the first spectral peak will be better encoded.

5.3.5.7 Split multi-rate lattice VQ

To quantize the pre-shaped spectrum X of the weighted signal in TCX mode, a method based on lattice quantizers is used. Specifically, the spectrum is quantized in 8-dimensional blocks using vector codebooks composed of subsets of the Gosset lattice, referred to as the RE_8 lattice (see [6]). All points of a given lattice can be generated from the so-called *generator matrix* G of the lattice, as $c = k\,G$, where k is a line vector with integer values and c is the generated lattice point. To form a vector codebook at a given rate, only lattice points inside a sphere (in 8 dimensions) of a given radius are taken. Multi-rate codebooks can thus be formed by taking subsets of different radii.

In lattice quantization, the operation of finding the nearest neighbour of an input vector x among all codebook points is reduced to a few simple operations, involving rounding the components of a vector and verifying a few constraints. Hence, no exhaustive search is carried out as in stochastic quantization, which uses stored tables. Once the best lattice codebook point is determined, further calculations are also necessary to compute the binary index that will be sent to the decoder. The larger the components of the input vector x, the more bits will be required to encode the index of its nearest neighbour in the lattice codebook. Hence, to remain within a pre-defined bit budget, a gain-shape approach has to be used, where the input vector is first scaled down, i.e. divided by a gain which has to be estimated, then quantized in the lattice, then scaled up again to produce the quantization result. To reduce computation complexity, the binary indices will actually only be calculated if a given TCX mode is retained as the best mode for a frame.

For simplicity, we let N be the length of the DFT. Since the transform used to obtain X is a Discrete Fourier Transform, there are $N/2+1$ Fourier coefficients including $X(N/2)$ at Nyquist frequency. In the quantization process, coefficient $X(N/2)$ is always set to 0, so there are exactly $N/2$ Fourier coefficients to quantize. Then, all coefficients of X are complex, except $X(0)$ which is real.

To be quantized using the RE_8 lattice codebooks, the pre-shaped spectrum X is split into consecutive blocks of 8 real values (4 consecutive complex coefficients). There are $K=N/8$ such blocks in the whole spectrum. We call B_k the k^{th} block, with $k = 0, 1, \ldots, K-1$. To remain within the total bit budget, the spectrum X will have to be divided by a global gain g prior to quantization, and multiplied by the quantized global gain after each block B_k is encoded using the RE_8 lattice. We call $X'=X/g$ the scaled spectrum and $B'_k = B_k / g$ the k^{th} scaled block. Thus, the parameters sent to the decoder to encode the TCX spectrum X are the global gain g and the index of the nearest neighbour of each block B_k within the lattice codebook.

The index of the nearest neighbour in the lattice is actually composed of three parts: 1) a codebook index, which essentially represents the bit allocation for each 8-dimensional vector; 2) a vector index, which uniquely identifies a lattice vector in a so-called *base codebook* C; and 3) an extention index k, which is used to extend the base codebook when the selected point in the lattice is not in the base codebook C. The extension used, called the *Voronoi* extension, will be described in Step 5 below.

These parameters are encoded using the 5 Steps described below.

Step 1 Find the energy E_k of each block B_k:

$$E_k = \max(2, \sum_{m=0}^{7} B_k[m] B_k[m])$$

and obtain from E_k a first estimate of the bit budget using the starting assumption that the global gain g equals 1 (i.e. that the spectrum X is quantized without scaling first):

$$R_k(1)=5\log_2(\frac{E_k}{2})$$

The formula for $R_k(1)$ is based on the properties of the underlying RE_8 lattice, and the method used for encoding the index of a lattice point selected by the quantizer. These properties and encoding method will be described in Steps 3 and 5.

Unless the energy of the frame is very small, the block energies E_k will be too large to ensure that the total bit consumption (sum of all $R_k(1)$) remains within the total bit budget for the frame. Hence, it is necessary to estimate a gain g so that the quantization of $X'=X/g$ in the RE_8 lattice will produce a set of indices that stay within the bit budget. This gain estimation is performed in Step 2.

Step 2 The estimation of the global gain g for the TCX frame is performed in an iteration, as follows.

Initialisation: Set *fac* = 128, *offset* = 0 and *nbits_max* = 0.95*(NB_BITS_ - *K*)

Iteration: Do the following block of operations *NITER* times (here, *NITER* = 10).

1- offset = offset + fac

2- $nbits=\sum_{k=1}^{K}\max(0,R_k(1)-offset)$

3- if nbits <= nbits_max, then offset = offset – fac

fac = fac / 2

After the iteration, the global gain is estimated as

$$g=10^{\frac{offset*\log_{10}(2)}{10}}$$

The scaled spectrum can then be obtained as X'=X/g. The input to the lattice quantizer described in Step 3 are the scaled blocks $B'_k = B_k / g$, each an 8-dimensional vector of real components. The assumption if that the total number of bits used to quantize B'_k into the lattice codebook will be close to the bit budget.

Step 3 In this step, each 8-dimensional block B'_k of the scaled spectrum $X'=X/g$ is rounded as a point in the RE_8 lattice, to produce its quantized version, $\hat{X}'$. Before looking at the quantization procedure, it is worthwhile to look at the properties of this lattice. RE_8 is defined as follows:

$$RE_8=2D_8\cup\{2D_8+(1,\cdots,1)\}$$

that is as the union of the $2D_8$ lattice and a version of $2D_8$ shifted by the vector (1,1,1,1,1,1,1,1). Therefore, searching for the nearest neighbour in the lattice RE_8 is equivalent to searching for the nearest neighbour in the lattice $2D_8$, then searching for the nearest neighbour in the lattice $2D_8$ + (1,1,1,1,1,1,1,1), and finally selecting the best of those two lattice points. The lattice $2D_8$ is just the D_8 lattice scaled by a factor of 2, with the D_8 lattice defined as:

$$D_8=\{(x_1,\cdots,x_8)\in Z^8 \mid x_1+\cdots+x_8 \quad is \quad even\}$$

That is, the lattice points in D_8 are all integers, with the constraint that the sum of all components is even. This also implies that the sum of the components of a lattice point in $2D_8$ is an integer multiple of 4.

From this definition of RE_8, it is straightforward to develop a fast algorithm to search for the nearest neighbour of an 8-dimensional block B'_k among all lattice points in RE_8. This is done by applying the following operations. We note that the components of B'_k are floating point values. The result of the quantization, $\hat{B}'_k$, will be a vector of integers.

1. $z_k = 0.5 * B'_k$

2. Round each component of z_k to the nearest integer, to generate $\bar{z}_k$

3. $y1_k = 2\,\bar{z}_k$

4. calculate S as the sum of the components of $y1_k$

5. If S is not an integer multiple of 4 (negative values are possible), then modify one of its components as follows:

   - find the position I where abs($z_k(i)$- $y1_k(i)$) is the highest
   - if $z_k(I)$- $y1_k(I)$ < 0, then $y1_k(I) = y1_k(I) - 2$
   - if $z_k(I)$- $y1_k(I)$ > 0, then $y1_k(I) = y1_k(I) + 2$

6. $z_k = 0.5 * (B'_k$ - **1.0**) where **1.0** denotes a vector with all 1's

7. Round each component of z_k to the nearest integer, to generate $\bar{z}_k$

8. $y2_k = 2\,\bar{z}_k$

9. calculate S as the sum of the components of $y2_k$

10. If S is not an integer multiple of 4 (negative values are possible), then modify one of its components as follows:

    - find the position I where abs($z_k(i)$- $y2_k(i)$) is the highest
    - if $z_k(I)$- $y2_k(I)$ < 0, then $y2_k(I) = y2_k(I) - 2$
    - if $z_k(I)$- $y2_k(I)$ > 0, then $y2_k(I) = y2_k(I) + 2$

11. $y2_k = y2_k$ + **1.0**

12. Compute $e1_k = (B'_k - y1_k)^2$ and $e2_k = (B'_k - y2_k)^2$

13. If $e1_k > e2_k$, then the best lattice point (nearest neighbour in the lattice) is $y1_k$

    otherwise the best lattice point is $y2_k$.

    This is noted as $\hat{B}'_k = c_k$ where c_k is the best lattice point as selected above.

Through this quantization procedure, the scaling gain g, estimated in Step 2, is left unquantized. The gain will be quantized only after being recomputed as in Section 5.3.5.10, to obtain $\hat{g}$. The quantized spectrum will then be obtained as $\hat{X} = \hat{g}\hat{X}'$.

We note that after this lattice quantization step, the indices of the selected lattice points are not known. The indices will only be computed if a particular TCX mode is selected instead of an ACELP mode. (See Step 5 for the lattice index computation)

Step 4 A last step in the quantization procedure is the determination and quantization of a comfort noise factor. Comfort noise enhances the perceived quality in transform-based coders, which is the case for the TCX modes. Comfort noise will be added only to unquantized spectral components in the upper-half of the spectrum (Fs/8 kHz and above). Taking again K as the total number of 8-dimensional blocks in the spectrum, the comfort noise factor is calculated as follows:

Initialisation: Set *nbits* = 0, n = 1 and take the *offset* value at the end of the iteration in Step 2 above.

Iteration : For $k = K / 2$ to K-1, do

1 *tmp* = $R_k(1)$ - *offset* (with $R_k(1)$ as calculated in Step 1)

2. if (*tmp* < 5), then *nbits* = *nbits* + *tmp* and $n = n + 1$

Noise factor calculation:

Set *nbits* = *nbits*/ n, and evaluate the noise factor as

$$Noise\_factor = 10^{\frac{(nbits-5)*\log_{10}(2)}{10}}$$

The noise factor will be comprised between 0 and 1.

Noise level quantization

The comfort noise factor is quantized using 3 bits in the range from 0.8 to 1.0.

Step 5 In Step 3, each scaled block B'_k was rounded as a point in the RE_8 lattice. The result is $c_k = \hat{B}'_k$, the quantized version of B'_k. If the corresponding TCX mode is actually retained as the best encoding mode for that frame (in open-loop fashion as in Section 5.2.4 or in closed-loop fashion as in Section 5.2.3), then an index has to be computed for each c_k for transmission to the decoder. The computation of these indices is described in this final Step.

The calculation of an index for a given point in the RE_8 lattice is based on two basic principles:

1- All points in the lattice RE_8 lie on concentric spheres of radius $\sqrt{8m}$ with m = 0, 1, 2, 3, etc., and each lattice point on a given sphere can be generated by permuting the coordinates of reference points called *leaders*. There are very few leaders on a sphere, compared to the total number of lattice points which lie on the sphere. Codebooks of different bit rates can be constructed by including only spheres up to a given number m. See reference [6] for more details, where codebooks Q_0, Q_1, Q_2, Q_3, Q_4, and Q_5 are constructed with respectively 0, 4, 8, 12, 16 and 20 bits. Hence, codebook Q_n requires $4n$ bits to index any point in that codebook.

2- From a base codebook C (i.e. a codebook containing all lattice points from a given set of spheres up to a number m), an extended codebook can be generated by multiplying the elements of C by a factor M, and adding a second-stage codebook called the *Voronoi extension*. This construction is given by $y = M z + v$, where M is the scale factor, z is a point in the base codebook and v is the Voronoi extension. The extension is computed in such a way that any point $y = M z + v$ is also a lattice point in RE_8. The extended codebook includes lattice points that extend further out from the origin than the base codebook.

The base codebook C in the present TCX modes can be either codebook Q_0, Q_2, Q_3 or Q_4 from reference [6]. When a given lattice point c_k is not included in these base codebooks, the Voronoi extension is applied, using this time only the Q_3 or Q_4 part of the base codebook. Note that here, $Q_2 \subset Q_3$ but $Q_3 \not\subset Q_4$.

Then, the calculation of the index for each lattice point c_k (quantization result in Step 3) is done according to the following operations.

Verify if c_k is in the base codebook *C*. Here, this implies verifying if c_k is an element of Q_0, Q_2, Q_3 or Q_4 from [6]. If y is in C, the index used to encode c_k is thus the codebook number n_k plus the index I_k of codevector c_k in Q_{nk}. The codebook number n_k is encoded as a unary code, as follows:

$Q_0 \rightarrow$ unary code for n_k is 0

$Q_2 \rightarrow$ unary code for n_k is 10

$Q_3 \rightarrow$ unary code for n_k is 110

$Q_4 \rightarrow$ unary code for n_k is 1110

The terminating "0" in this unary code will indicate to the decoder the separation between the successive blocks c_k in the bit stream.

The index I_k indicates the rank of c_k, i.e. the permutation to be applied to a specific leader to obtain c_k (see [6]). Note that if $n_k = 0$, then I_k uses no bits. Otherwise, the index I_k uses $4n_k$ bits. Hence, a total of $5n_k$ bits are required to index any lattice point in the base codebook: n_k bits for the unary code specifying the codebook number (with the exception that 1 bit is required for Q_0), and $4n_k$ bits to index the lattice point c_k in that codebook.

If c_k is not in the base codebook, then apply the Voronoi extension through the following sub-steps, using this time only Q_3 or Q_4 as the base codebook.

V0 Set the extension order $r = 1$ and the scale factor $M = 2^r = 2$.

V1 Compute the Voronoi index k of the lattice point c_k. The Voronoi index k depends on the extension order r and the scale factor M. The Voronoi index is computed via modulo operations such that k depends only on the relative position of c_k in a scaled and translated Voronoi region:

$$k = \mathrm{mod}_M (c_k G^{-1})$$

Here, G is the generator matrix and $\mathrm{mod}_m(\cdot)$ is the component-wise modulo-M operation. Hence, the Voronoi index k is a vector of integers with each component comprised in the interval 0 to $M-1$.

V2 Compute the Voronoi codevector v from the Voronoi index k. This can be implemented using an algorithm described in [7].

V3 Compute the difference vector $w = c_k - v$. This difference vector w always belongs to the scaled lattice $m\Lambda$, where Λ is the lattice RE_8. Compute $z = w/M$, i.e., apply the inverse scaling to the difference vector w. The codevector z belongs to the lattice Λ, since w belongs to $M\Lambda$.

V4 Verify if z is in the base codebook C (i.e. in Q_3 or Q_4)

If z is not in C, increment the extension order r by 1, multiply the scale factor M by 2, and go back to sub-step V1.

Otherwise, if z is in C, then we have found an extension order r and a scaling factor $M = 2^r$ sufficiently large to encode the index of c_k. The index is formed of three parts: 1) the codebook index n_k as a unary code defined below; 2) the rank I_k of z in the corresponding base codebook (either Q_3 or Q_4); and 3) the 8 indices of the Voronoi index vector k calculated in sub-step V1, where each index requires exactly r bits (r is the Voronoi extension order set in sub-step V0).

The codebook index n_k is encoded in unary code as follows:

n_k = 11110	when the base codebook is Q_3 and the Voronoi extension order is $r = 1$
n_k = 111110	when the base codebook is Q_4 and the Voronoi extension order is $r = 1$
n_k = 1111110	when the base codebook is Q_3 and the Voronoi extension order is $r = 2$
n_k = 11111110	when the base codebook is Q_5 and the Voronoi extension order is $r = 2$

etc.

The lattice point c_k is then described as

$$c_k = M\ z + v$$

The packetisation of these indices into transmission packets will be described in Section 5.6.1.

5.3.5.8 Spectrum de-shaping

Spectrum de-shaping is applied to the quantized spectrum prior to applying the inverse FFT. The de-shaping is done according to the following steps:

- calculate the energy E_m of the 8-dimensional block at position index m
- compute the ratio $R_m = E_{max} / E_m$
- compute the value $(R_m)^{1/2}$
- if $R_m > 10$, then set $R_m = 10$ (maximum gain of 20 dB)
- also, if $R_m > R_{m-1}$ then $R_m = R_{m-1}$

After computing the ratio $R_m = (E_{max} / E_m)^{1/2}$ for all blocks with position index smaller that I (and with the limiting conditions described above), we then divide each block by the corresponding ratio. Note that if we neglect the effects of quantization, this de-shaping is the inverse of the pre-shaping function as applied in Section 5.3.5.6.

5.3.5.9 Inverse transform

The quantized spectrum $\hat{X}(k)$ is inverse transformed to obtain the time-domain quantized signal $\hat{x}(n)$. The Inverse DFT is applied, as defined by:

$$\hat{x}(n) = \sum_{n=0}^{L_{TOT}-1} \hat{X}(k) e^{j\frac{2\pi}{L_{TOT}}nk}$$

where L_{TOT} is the number of samples in the TCX frame, as defined in Section 5.3.5.5. An Inverse FFT is used to optimize the computation time of the inverse DFT.

5.3.5.10 Gain optimization and quantization

The global gain estimated in Section 5.3.5.7 to scale the spectrum prior to the multi-rate lattice quantization is not guaranteed to maximize the correlation between the original weighted signal x and the quantized weighted signal $\hat{x}$. Thus, after the inverse transform of the quantized spectrum (Section 5.3.5.9), the optimal gain between x and $\hat{x}$ is computed as follows:

$$g^* = \frac{\sum_{n=0}^{L_{TOT}-1} x(n)\hat{x}(n)}{\sum_{n=0}^{L_{TOT}-1} \hat{x}(n)\hat{x}(n)}$$

with L_{TOT} as defined previously. Then, the gain g^* is quantized on a logarithmic scale to a 7-bit index, using the following procedure. The procedure is purely algebraic and does not require storing a gain codebook.

1. Calculate the energy of the quantized weighted signal: $E = \sum_{n=0}^{L_{TOT}-1} \hat{x}(n)^2$
2. Compute the RMS value: $rms = 4\sqrt{\frac{E}{L_{TOT}}}$ (known also at the decoder)

3. Set $G = g^* \times rms$ (normalization step)
4. Calculate the index as $index = \lfloor 28\log_{10}(G) + 0.5 \rfloor$ where $\lfloor x \rfloor$ denotes removing the fractional part of x (rounding towards 0).
5. If $index < 0$, then set $index = 0$, and if $index > 127$, then set $index = 127$.

The quantized gain $\hat{g}^*$ can be calculated as follows, both as the encoder and decoder since the decoder can calculate locally the value of *rms*:

$$\hat{g}^* = 10^{\frac{index}{28*rms}}$$

5.3.5.11 Windowing for overlap-and-add

After gain scaling, the quantized weighted signal is windowed again, according to the TCX frame length and the mode of the previous frame. The window shapes are as shown in Figure 8 and defined in Section 5.3.5.4.

To reconstruct the complete quantized weighted signal, overlap-and-add is applied between the memory of the past frame and the beginning of the present frame corresponding to the non-flat portion of the window. Recall that if the past frame was in ACELP mode, the memory of the past frame corresponds to the windowed, truncated ZIR of the perceptual filter, as calculated in Section 5.3.5.3.

5.3.5.12 Memory update

The samples in the lookahead (windowed portion to the right of the TCX frames in Figure 8) are kept in memory for the overlap-and-app procedure in the next TCX frame.

5.3.5.13 Excitation signal computation

The excitation signal is finally computed by filtering the quantized weighted signal through the inverse weighting filter with zero-memory. The excitation is needed at the encoder in particular to update the long-term predictor memory.

5.4 Mono Signal High-Band encoding (BWE)

The encoding of the HF signal is detailed in Figure 9. The HF signal is composed of the frequency components above Fs/4 kHz in the input signal. The bandwidth of this HF signal depends on the input signal sampling rate. To encode the HF signal at a low rate, a bandwidth extension (BWE) approach is employed. In BWE, energy information is sent to the decoder in the form of spectral envelope and frame energy, but the fine structure of the signal is extrapolated at the decoder from the received (decoded) excitation signal in the LF signal.

The down-sampled HF signal is called $s_{HF}(n)$ in Figure 9. The spectrum of this signal can be seen as a folded version of the high-frequency band prior to down-sampling. An LP analysis is performed on $s_{HF}(n)$ to obtain a set of coefficients which model the spectral envelope of this signal. Typically, fewer parameters are necessary than in the LF signal. Here, a filter of order 8 is used. The LP coefficients are then transformed into ISP representation and quantized for transmission. The number of LP analysis in an 1024-sample super-frame depends on the frame lengths in the super-frame.

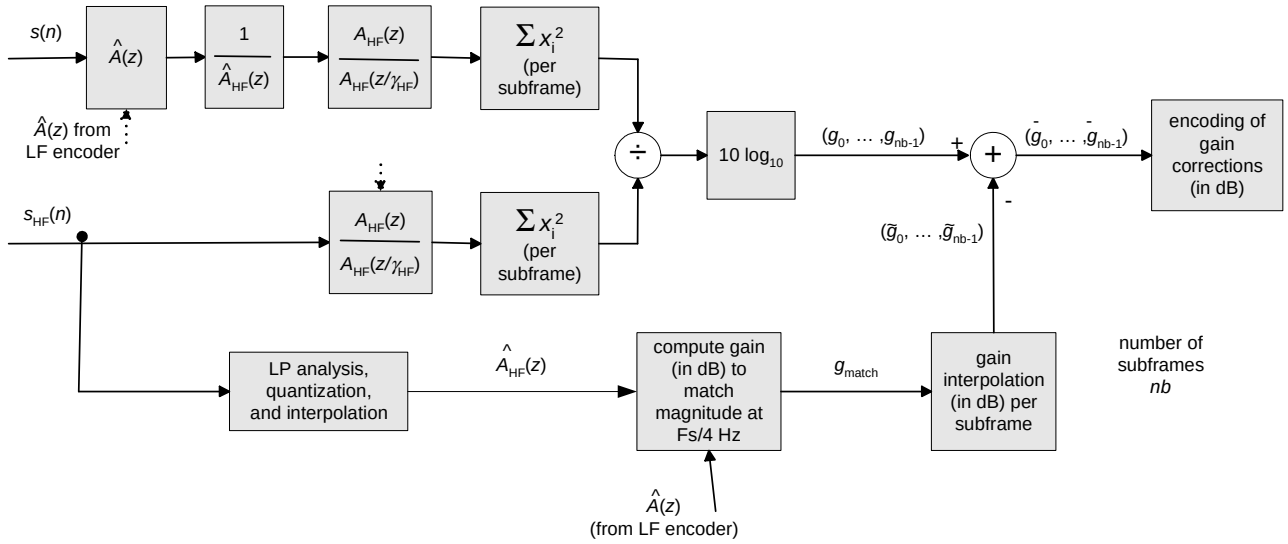


Figure 9: High frequency encoding

The LP filter for the HF signal is denoted by $A_{HF}(z)$, and its quantized version is denoted by $\widehat{A}_{HF}(z)$. From the LF signal ($s(n)$ in Figure 9), a residual signal is first obtained by filtering $s(n)$ through the inverse filter $\widehat{A}(z)$. Then, this residual is filtered through the quantized HF synthesis filter, $1/\widehat{A}_{HF}(z)$. Up to a gain factor, this produces a good approximation of the HF signal, but in a spectrally folded version. The actual HF synthesis signal will be recovered when up-sampling is applied to this signal

Since the excitation is taken from the LF signal, an important step is to compute the proper gain for the HF signal. This is done by comparing the energy of the reference HF signal ($s_{HF}(n)$) with the energy of the synthesized HF signal. The energy is computed once per 64-sample subframe, with energy match ensured at the Fs/4 kHz subband boundary. Specifically, the synthesized HF signal and the reference HF signal are filtered through a perceptual filter derived from $A_{HF}(z)$. The ratio of the energy of these two filtered signals is computed every 64 samples, and expressed in dB. There are 4 such gains in a 256-sample frame (one for every 64-sample subframe). This 4-gain vector represents the gain that should be applied to the HF signal to properly match the HF signal energy. Instead of transmitting this gain directly, an estimated gain ratio is first computed by comparing the gains of filters $\widehat{A}(z)$ from the lower band and $\widehat{A}_{HF}(z)$ from the higher band. This gain ratio estimation is detailed in Figure 10 and will be explained below. The gain ratio estimation is interpolated every 64 samples, expressed in dB and subtracted from the measured gain ratio. The resulting gain differences or gain corrections, noted $\overline{g}_0$ to $\overline{g}_{nb-1}$ in Figure 9, are quantized as 4-dimensional vectors, i.e. 4 values per 256-sample frame.

The gain estimation computed from filters $\widehat{A}(z)$ and $\widehat{A}_{HF}(z)$ is detailed in Figure 12. These two filters are available at the decoder side. The first 64 samples of a decaying sinusoid at Nyquist frequency π radians per sample is first computed by filtering a unit impulse through a one-pole filter. The Nyquist frequency is used since the goal is to match the filter gains at around Fs/4 kHz, i.e. at the junction frequency between the LF and HF signals. Note the 64-sample length of this reference signal is the sub-frame length (64 samples). The decaying sinusoid is then filtered first through $\widehat{A}(z)$, to obtain a low-frequency residual, then through $1/\widehat{A}_{HF}(z)$ to obtain a synthesis signal from the HF synthesis filter. We note that if filters $\widehat{A}(z)$ and $\widehat{A}_{HF}(z)$ have identical gains at the normalized frequency of π radians per sample, the energy of the output of $1/\widehat{A}_{HF}(z)$ would be equivalent to the energy of the input of $\widehat{A}(z)$ (the decaying sinusoid). If the gains differ, then this gain difference is taken into account in the energy of the signal at the output, noted $x(n)$. The correction gain should actually increased as the energy of $x(n)$ decreases. Hence, the gain correction is computed as the multiplicative inverse of the energy of signal $x(n)$, in the logarithmic domain (i.e. in dB). To get a true energy ratio, the energy of the decaying sinusoid, in dB, should be removed from the output. However, since this energy offset is a constant, it will simply be taken into account in the gain correction encoder.

At the decoder, the gain of the HF signal can be recovered by adding $\tilde{g}_0$ to $\tilde{g}_{nb-1}$ (known at the decoder) to the decoded gain corrections.

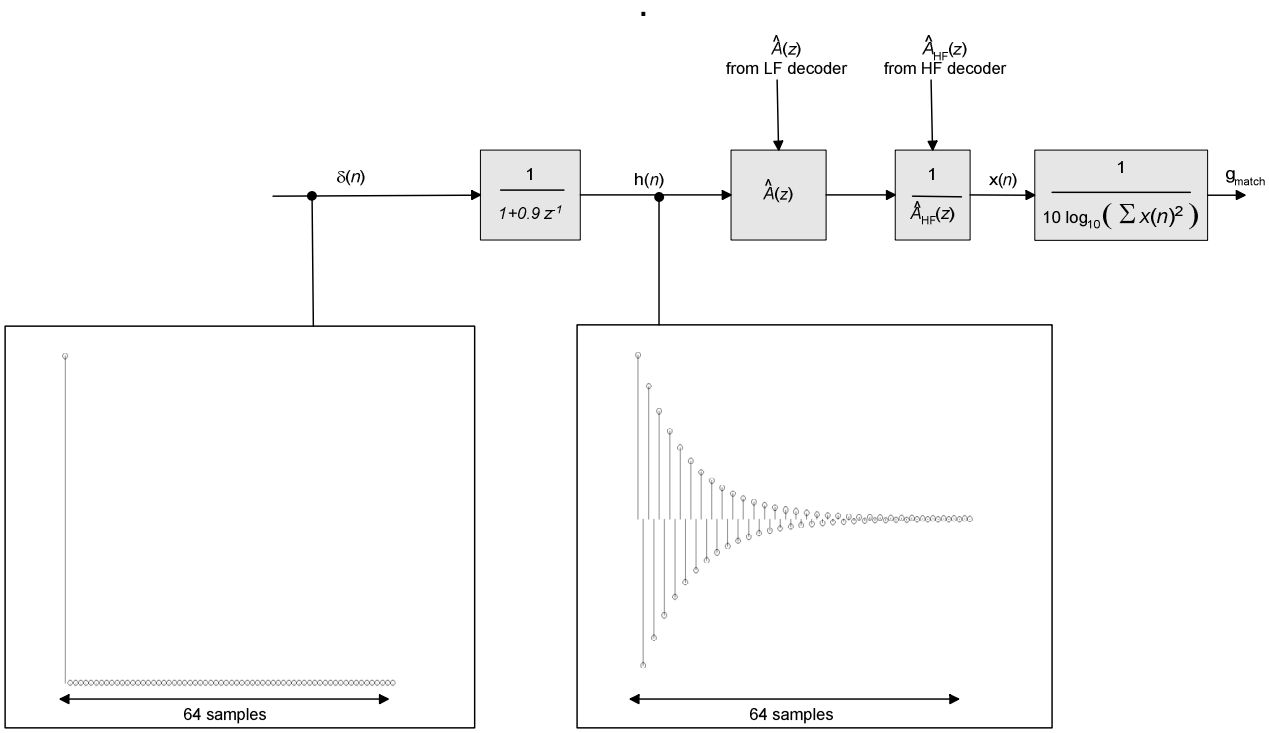
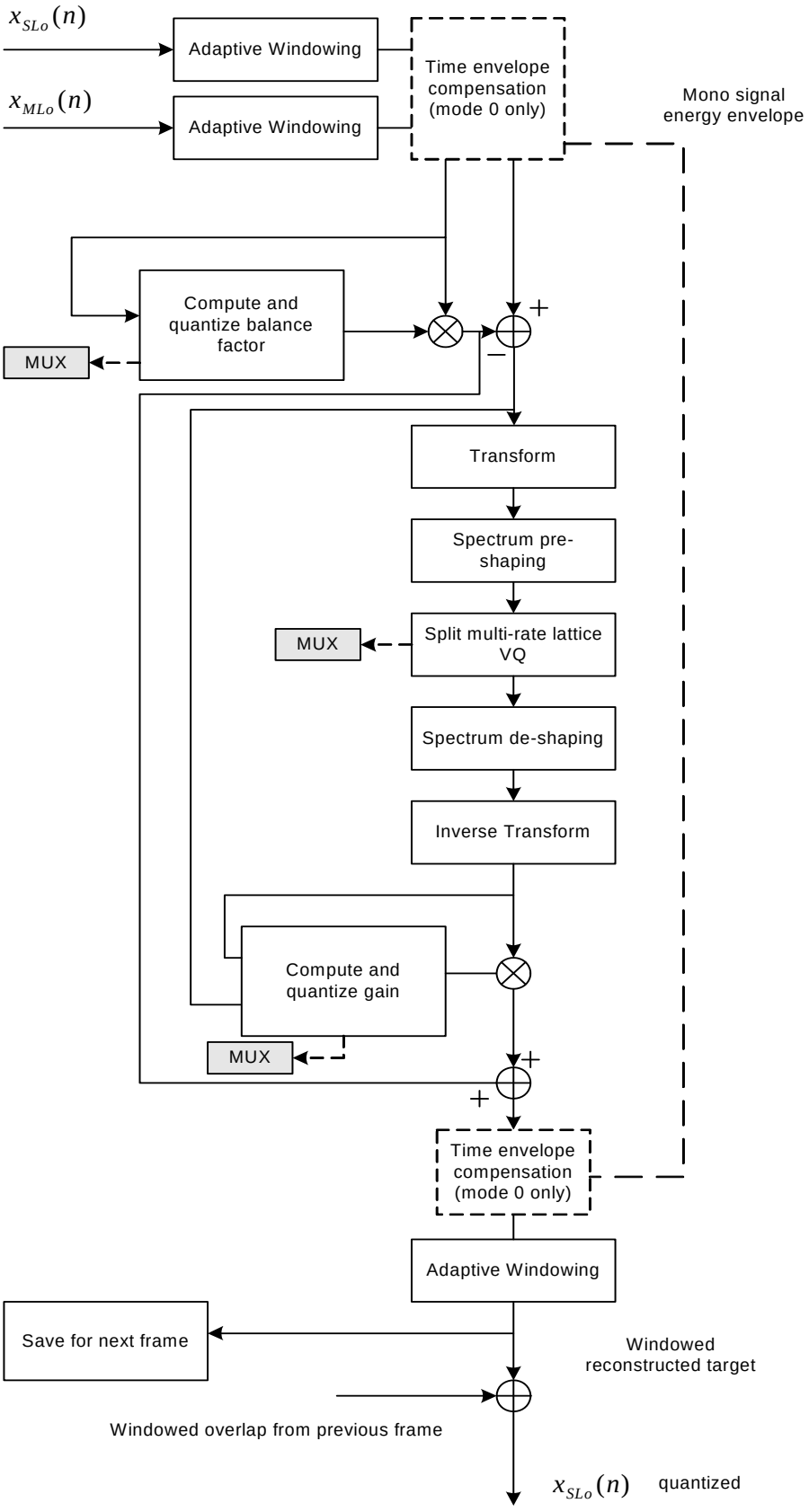


Figure 10: Gain matching between low and high frequency envelope

5.5 Stereo signal encoding

5.5.1 Stereo Signal Low-Band Encoding



5.5.1.1 Principle

The stereo Low band encoder receives the signals $x_{MLo}(n)$ and $x_{RLo}(n)$ for encoding. The Low band encoder is based on fidelity optimized encoding of the low band side signal. The Lo side signal is obtained by computing the difference

$$x_{SLo}(n) = x_{MLo}(n) - x_{RLo}(n),$$

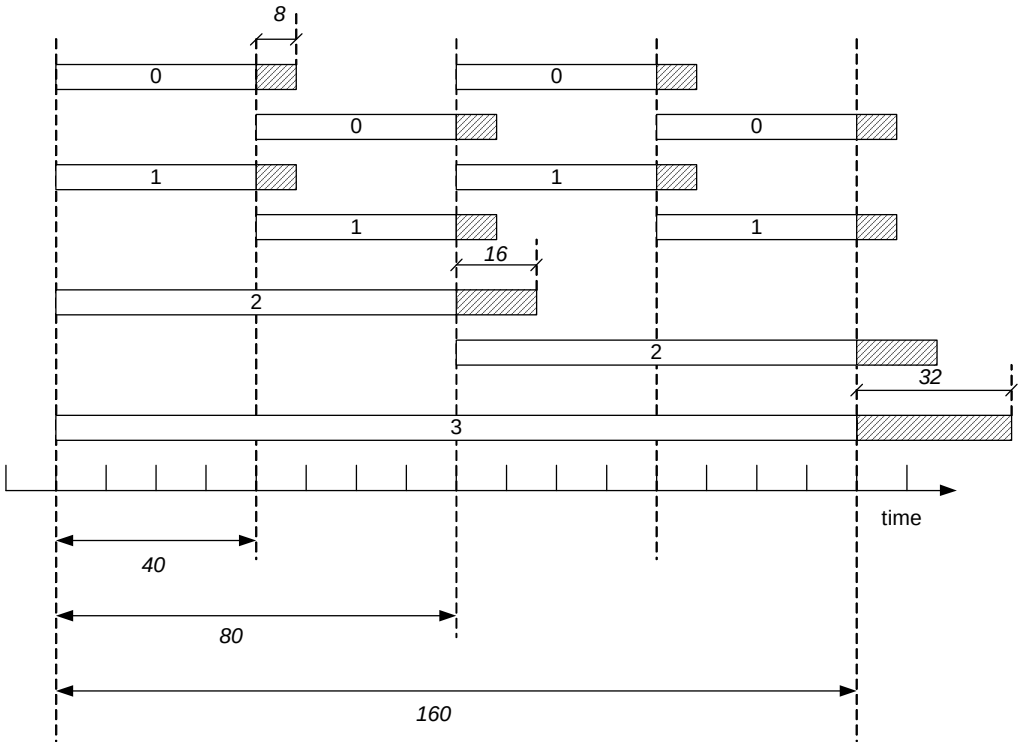
The encoding of the side signal is performed following a similar approach to that of the core encoder, except that the ACELP mode is not used. For each input signal block, the encoder decides in closed loop which encoding models to use. A signal to noise ratio fidelity criterion is used.

All 4 encoding models are based on encoding in the frequency domain a redundancy reduced side signal. In order to account for transients, there is an encoding model that uses pre-echo reduction. Encoding in the frequency domain uses the same split multi-rate lattice VQ.

Within each super-frame, the different encoding modes are:

mode	Duration (length + overlap)	Encoding
0	40 + 8	RR + Pre-echo + AVEQ
1	40 + 8	RR + AVEQ
2	80 + 16	RR + AVEQ
3	160 + 32	RR + AVEQ

The timing chart as well as the possible mode combinations is similar to that of the core encoder and is described in the following figure:



Selection of the encoding mode is done by closed loop search identical to the one used for the core encoder.

5.5.1.2 Signal Windowing

The two signals $x_{SLo}(n)$ and $x_{MLo}(n)$ are windowed prior to redundancy removal and frequency transformation. This is necessary in order to apply overlap-add to minimize the framing effects due to quantization. The window shape is adaptive depending on the previous coding mode and is similar to that described in section 5.3.5.4. The windowed signals are denoted by $\tilde{x}_{SLo}(n)$ and $\tilde{x}_{MLo}(n)$.

5.5.1.3 Pre-echo mode

In order to encode transients more efficiently, a pre-echo mode is used. It is often the case in transients that the energy envelope of the mono signal is highly correlated with that of the side signal. The energy envelope of the mono signal is derived and normalized, it is then used to compensate for the energy envelope of the side signal.

5.5.1.4 Redundancy reduction

For all encoding modes, a balance factor is used in order to remove the portion of the side signal that is correlated with the mono signal. The balance factor is given by

$$balance\_factor = \frac{\sum_n \tilde{x}_{SLo}(n)\tilde{x}_{MLo}(n)}{\sum_n \tilde{x}_{MLo}(n)^2}$$

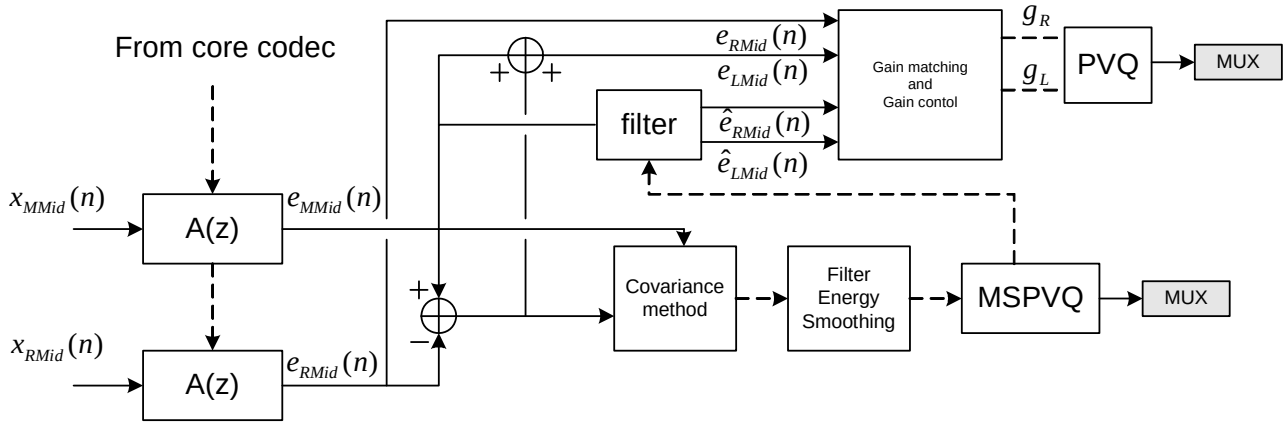
the balance factor is quantized by a uniform scalar quantizer with 7 bits.

5.5.2 Stereo Signal Mid-Band Processing

5.5.2.1 Principle

The encoder takes the mid band mono and right channel signals, $x_{MMid}(n)$ and $x_{RMid}(n)$, and inverse filters it with the core codec LPC filters derived from the mono signal. In the residual domain a shape constrained FIR filter is computed for approximating the side signal. The filter is computed by means of the covariance method using a novel spectral shape constraint. A new filter is computed for each *Ldiv*=256 sample frame with an analysis frame of 320 samples. The energy of the filter is smoothed to avoid sudden energy changes.The smoothed filter is quantized with a multistage predictive vector quantizer (MSPVQ).

The mono residual signal $e_{MMid}(n)$ is filtered with the quantized filter and gain factors are computed for the left and right channels respectively.



5.5.2.2 Residual computation

The residual signal is computed according to

$$e_{MMid}(n) = \sum_{i=0}^{16} a_i(n) x_{MMid}(n-i), \quad 0 \le n < L_{div} + L_{subfr}$$

$$e_{RMid}(n) = \sum_{i=0}^{16} a_i(n) x_{RMid}(n-i), \quad 0 \le n < L_{div} + L_{subfr}$$

The quantized and interpolated LPC coefficients from the core codec are used in the inverse filter operation, in addition an extra subframe is computed for the overlapped analysis section. The residual side signal is computed as

$$e_{SMid}(n) = e_{MMid}(n) - e_{RMid}(n), \quad 0 \le n < L_{div} + L_{subfr}$$

5.5.2.3 Filter computation, smoothing and quantization

The filter is computed that minimizes the expression $\sum_{n=0}^{L_{div}+L_{subfr}} \left[e_{SMid}(n) - \sum_{i=0}^{8} b_i e_{MMid}(n-i) \right]^2$ under the constraint of a spectral null at 0Hz. The filter coefficients for the filter are computed with the well know covariance method using a modified cholesky algorithm taking into account the shape constraint.

To avoid to fast changes in the filter energy, the filter energy is smoothed over time. The filter energy is first computed as

$$E_b = \sum_{i=0}^{8} b_i^{\,2}$$

the new filter energy is computed such that the filter energy is saturated to 16 and that transitions between frames are limited within a +/- 1.5 dB interval.

After the smoothing operation the filter coefficients are quantized using a predictive multistage vector quantizer.

5.5.2.4 Channel energy matching

The quantized filter coefficients $b_{0q}, b_{1q}, \ldots, b_{8q}$ are used to filter the mono signal excitation in order to get an initial estimate of the left and the right channel excitation signals. These estimates are computed as

$$\hat{e}_{LMid}(n) = e_{MMid}(n) - \sum_{i=0}^{8} b_{iq} e_{MMid}(n-i), \quad 0 \le n < L_{div} + L_{subfr}$$

$$\hat{e}_{RMid}(n) = e_{MMid}(n) + \sum_{i=0}^{8} b_{iq} e_{MMid}(n-i), \quad 0 \le n < L_{div} + L_{subfr}$$

The energy matching for the left and right channels is computed as

$$g_L = 10 \log_{10} \frac{\sum_{n=0}^{L_{div}+L_{subfr}} e_{LMid}(n)^2}{\sum_{n=0}^{L_{div}+L_{subfr}} \hat{e}_{LMid}(n)^2}$$

$$g_R = 10\log_{10}\frac{\sum_{n=0}^{L_{div}+L_{subfr}} e_{RMid}(n)^2}{\sum_{n=0}^{L_{div}+L_{subfr}} \hat{e}_{RMid}(n)^2}$$

The computed gains are adjusted in case of anti-correlation by computing a correlation gain

$$g_{CORR} = 10\log_{10}\frac{4\sum_{n=0}^{L_{div}+L_{subfr}} e_{MMid}(n)^2}{\sum_{n=0}^{L_{div}+L_{subfr}} e_{RMid}(n)^2 + e_{RMid}(n)^2}$$

if g_{CORR} < 0dB the gain matching factors are adjusted according to

$$g_L = \begin{cases} g_L & g_L < 0\text{dB} \\ \max(0, g_L + g_{CORR}) & \text{otherwise} \end{cases} \text{ and } g_R = \begin{cases} g_R & g_R < 0\text{dB} \\ \max(0, g_R + g_{CORR}) & \text{otherwise} \end{cases}$$

The energy matching factors are quantized using a two-dimensional vector-quantizer.

5.5.3 Stereo Signal High-Band Processing

In the stereo case two bandwith extensions are used. Both use the mono coder excitation as excitation source.

5.6 Packetization

5.6.1 Packetization of TCX encoded parameters

This section explains how the TCX encoded parameters are put in one or several binary packets for transmission. One packet is used for 256-sample TCX, while respectively 2 and 4 packets are used for 512- and 1024-sample TCX. To split the TCX spectral information in multiple packets (in case of 512- and 1024-sample TCX), the spectrum is divided into interleaved *tracks*, where each track contains a subset of the splits in the spectrum (each split represent 8-dimensional vectors encoded with algebraic VQ, and the bits of individual splits are not divided across different tracks). If we number the splits in the spectrum, from low to high frequency, with the split numbers 0, 1, 2, 3, etc. up to the last split at the highest frequency, then the tracks are as shown in the following table

Table 13: Dividing spectral splits in different tracks for packetization

		Split numbers
256-sample-TCX	Track 1	0, 1, 2, 3, etc (only one track)
512-sample TCX	Track 1	0, 2, 4, 6, etc.
	Track 2	1, 3, 5, 7, etc.
1024-sampleTCX	Track 1	0, 4, 8, 12, etc.
	Track 2	1, 5, 9, 13, etc.
	Track 3	2, 6, 10, 14, etc.
	Track 4	3, 7, 11, 15, etc.

Then, recall that the parameters of each split in algebraic VQ consist of the codebook numbers **n** = [n_0 ... n_{K-1}] and the indices **i** = [i_0 ... i_{K-1}] of all splits. The values of codebooks numbers **n** are in the set of integers {0, 2, 3, 4,…}. The size (number of bits) of each index i_k is given by $4n_k$. To write these bits into the different packets, we associate a track number to each packet. In the case of 256-sample TCX, only one track is used (i.e. all the splits in the spectrum) and it is written in a single packet. In the case of 512-sample TCX, two packets are used: the first packet is used for Track 1

and the second packet for Track 2. In the case of 1024-sample TCX, four packets are used: the first packet is used for Track 1, the second packet for Track 2, the third packet for Track 3 and the fourth packet for Track 4. However, the spectrum quantization and bit allocation was performed without constraining each track to have the same amount of bits, so in general the different tracks do not have the same number of bits allocated to the respective splits. Hence, when writing the encoded splits (codebook numbers and lattice point indices) of a track into their respective packet, two situations can occur: 1) there are not enough bits in the track to fill the packet or 2) there are more bits in a track than the size of the packet so there is overflow. The third possibility (exactly the same number of bits in a track as the packet size) occurs rarely. This overflow has to be managed properly, so all packets are completely filled, and so the decoder can properly interpret and decode the received bits. This overflow management will be explained below when the multiplexing for the case of multiple binary tables (i.e. tracks) is detailed.

The split indices are written in their respective packets starting from the lowest frequency split and scanning the track in the spectrum in increasing value of frequency. The codebook number n_K and index i_K of each split are written in separate sections of the packet. Specifically, the bits of the codebook number n_K (actually, its unary code representation) are written sequentially starting from one end of the packet, and the bits of the index i_K are written sequentially starting from the other end of the packet. Hence, overflow occurs when these concurrent bit writing processes attempt to overwrite each other. Alternatively, when the bits in one track do not completely fill a packet, there will be a "hole" (i.e. available position for writing more bits) somewhere in the middle of the packet. In 512-sample TCX, overflow will only occur in one of the two packets, while the other packet will have this "hole" where the overflowing bits of the other packet will be written. In 1024-sample TCX, there can be "holes" in more than one of the four packets after overflow has happened. In this case, all the "holes" will be grouped together and the overflowing bits of the other packets will be written into these "holes". Details of this procedure are given below.

Then, we note that the use of a unary code to encode the lattice codebook numbers (**n**) implies that each split requires actually $5n_k$ bits, when it is quantized using a point in the lattice codebook with number n_k. That is, n_k bits are used by the unary code (n_k -1 successive "1's" and a final "0") to indicate how many blocks of 4 bits are used in the codebook index, and $4n_k$ bits are used to form the actual lattice codebook index in codebook n_k) for the split. Note also that when a split is not quantized (i.e. set to zero by the TCX quantizer), it still requires 1 bit (a "0") in the unary code, to indicate that the decoder must skip this split and set it to zero.

Now, more details related to the multiplexing of algebraic vector quantizer indices in one or several packets are given below, in particular regarding the splitting of TCX indices in more than one packet (for 512-and 1024-sample TCX) and the management of overflow in writing the bits into the packets.

Recall that the codebook numbers are integers defined in the set {0,2,3,4,...., 36}. Each n_k has to be represented in a proper binary format, denoted hereafter n^E_k, for multiplexing.

5.6.1.1 Multiplexing principle for a single binary table

The multiplexing in a single binary table **t** consists of writing bit-by-bit all the elements of **n** and **i** inside **t**, where the table $\mathbf{t} = (t_0,\ldots, t_{R-1})$ contains R bits (which corresponds to the number of bits allocated to algebraic VQ).

A straightforward strategy amounts to writing sequentially the elements of $\mathbf{n}^E$ and **i** in the binary table **t**, as follows:

$$[n^E_0 \;\; i_0 \;\; n^E_1 \;\; i_1 \; n^E_2 \;\; i_2 \;\ldots.\;\;]$$

In this case, the bits of n^E_0 are written from position 0 in **t** and upward, the bits of i_0 then follow, etc. This format is uniquely decodable, because the encoded codebook number n^E_k indicates the size of i_k.

Instead, an alternative format is used as described below:

$$[\; i_0 \; i_1 \; i_2 \;\ldots\ldots\; n^E_2 \;\; n^E_1 \; n^E_0 \;]$$

The codebook numbers are written sequentially and downward from the end of the binary table **t**, whereas the indices are written sequentially and upward from the beginning of the table. This format has the advantage to separate codebook numbers and indices. This allows to take into account the different bit sensitivity of codebook numbers and indices. Indeed, with the multi-rate lattice vector quantization used, the codebooks numbers are the most sensitive parametersThus, they are written from the beginning of the table **t** and take around 20% of the total bit consumption, giving bitstream ordering according to bit sensitivity.

For the actual multiplexing, two pointers are then defined on the binary table **t**: one for (encoded) codebooks numbers pos_n, another for indices pos_i. The pointer pos_i is initialized to 0 (i.e. the beginning of the binary table), and pos_n to R-1 (i.e. the end of the binary table). Positive increments are used for pos_i, and negative ones for pos_n. At any time, the number of bits left in the binary table is given by pos_n-pos_i+1.

The table **t** is initialized to zero. This guarantees that if no data is written, the data inside this table will correspond to an all-zero codebook numbers **n** (this follows from the definition of the unary code used here). The splits are then written sequentially in the binary table from $k=0$ to $K-1$: $[n^E_0\ i_0]$ then $[n^E_1\ i_1]$ then $[n^E_2\ i_2]$, etc.

The data of the kth split are really written in the binary table **t** only if the minimal bit consumption of the kth split, denoted R_k hereafter, is less than the number of bits left in table **t**, i.e. if $R_k \le pos_n - pos_i + 1$. For the multi-rate lattice vector quantization used here, the minimal bit consumption R_k equals to 0 bit if $n_k=0$, or $5n_k-1$ bits if $n_k \ge 2$.

The multiplexing works as follows as shown in the algorithm of Figure 11.

Initialization:

$pos_i = 0$, $pos_n = R-1$

set binary table **t** to zero

For k=0 to $K-1$ (loop for all splits over the 4 steps below):

Compute the number of left bits in table **t**: $nb = pos_n - pos_i + 1$

Compute the minimal bit consumption of the kth split: $R_k = 0$ if $n_k=0$, $5n_k-1$ if $n_k \ge 2$

Figure 11: Multiplexing algorithm for one binary table

In practice, the binary table **t** is physically represented as having 4-bit elements instead of binary (1-bit) elements, so as to accelerate the write-in-table operations and avoid too many bit manipulations. This optimization is significant because the indices i_k are typically formatted into 4-bit blocks. In this case, the value of pos_i is always a multiple of 4. However, this implies to use bit shifts and modular arithmetic on pointers pos_n and pos_i to locate positions in the table.

5.6.1.2 Multiplexing in case of multiple binary tables

In the case of multiple binary tables, the algebraic VQ parameters are written in P tables $t_0, \ldots, t_{P-1}$ ($P \ge 1$) containing respectively $r_0, \ldots, r_{P-1}$ bits, such that $r_0 + \ldots + r_{P-1} = R$. In other words, the bit budget allocated to algebraic VQ parameters, R, is distributed to P binary tables. Here, L is set to 1 in the 256-sample TCX mode, 2 in the 512-sample TCX mode or 4 in the 1024-sample TCX mode.

Note that the multiplexing of algebraic VQ parameters in TCX modes employs frame-zero-fill if the bit budget allocated to algebraic VQ is not fully used.

We assume that the number of sub-vectors, K, is a multiple of P. Under this assumption, the algebraic VQ parameters are then divided into P groups of equal cardinality: each group comprises K/P (encoded) codebook numbers and K/P indices. By convention, the pth group is defined as the set $(n^E_{p+jP}, i_{p+jP})_{j=0..K/P-1}$. This can be seen as a decimation operation (in the usual multi-rate signal processing sense).

Assuming the size of table t_p is sufficient, the parameters of the pth group are written in table t_p. For the sake of clarity, the division of sub-vectors is explained below in more details for P=1 and 2:

If P=1, the set $(n^E_{p+jP}, i_{p+jP})_{j=0..K/P-1}$ for l=0 simply corresponds to $(n^E_0, i_0, \ldots, n^E_{K-1}, i_{K-1})$. These parameters are written in table t_0. This is the single-table case.

If P=2, we have $(n^E_{p+jP}, i_{p+jP})_{j=0..K/P-1} = (n^E_0, i_0, n^E_2, i_2 \ldots, n^E_{K-2}, i_{K-2})$ for p=0 and $(n^E_1, i_1, n^E_3, i_3 \ldots, n^E_{K-1}, i_{K-1})$ for p=1. Assuming the table sizes are sufficient, the parameters $(n^E_0, i_0, n^E_2, i_2 \ldots, n^E_{K-2}, i_{K-2})$ are written in table t_0, while the other parameters $(n^E_1, i_1, n^E_3, i_3 \ldots, n^E_{K-1}, i_{K-1})$ are written in table t_1.

The case of P=4 can be readily understood from the case of P=2.

As a consequence, in principle the multiplexing in the multiple-table case boils down to applying several times the single-table multiplexing principle: the (encoded) codebook numbers $(n^E_{p+jP})_{j=0..K/P-1}$ can be written upward from the

bottom of each table t_p and the indices $(i_{p+jP})_{j=0..K/P-1}$ can be written downward from the end of each table t_p. Two pointers are defined for each binary table t_p: $pos_{n,p}$ and $pos_{i,p}$. These pointers are initialized to $pos_{i,p} = 0$ and $pos_{n,p} = r_p - 1$, and are respectively incremented and decremented.

Nonetheless, the multiple-table case is not a straightforward extension of the single-packet case. It may happen indeed that the number of bits in $(n^E_{p+jP}, i_{p+jP})_{j=0..K/P-1}$ exceeds, for a given p, the number of bits, r_p, available in the binary table t_p. To deal with such an "overflow", an extra table t_{ex} is defined as temporary buffer to write the bits in excess (which have to be distributed in another table t_q with $q \neq p$). The size of t_{ex} is set to 4*36 bits.

The actual multiplexing algorithm in the multiple-table case is detailed below:

1) Initialize: (We assume that a size of r_p bits for each binary table t_p.)

Set total number of bits to R: $nb = R$

Initialize the maximum position *last* such that $n_{last} \geq 2$:

last = -1

For p=0…*P-1*,

$pos_{i,p} = 0$ and $pos_{n,p} = r_p - 1_1$

set table t_p to zero

2) Split and write all codebook numbers:

For p=0…*P-1*, the (encoded) codebook numbers $(n^E_{p+jP})_{j=0..K/P-1}$ are written sequentially (downward from the end) in table t_p. This is done through two nested loops over p and j. In the illustrative embodiment a single loop is used with modular arithmetic, as detailed below:

For k=0,…,K-1

$p = k \bmod P$

Compute the minimal bit consumption of the kth split: $R_k = 0$ if n_k=0, $5n_k$ -1 if $n_k \geq 2$

If $R_k > nb$, n_k=0 else $nb = nb - R_k$

If $n_k \geq 2$, $last = k$

Write downward n^E_k (except the stop bit) in table $\mathbf{t}_p$ starting from $pos_{n,p}$, and decrement $pos_{n,p}$ by n_k -1

If $nb \geq 0$, write the stop bit of the unary code and decrement $pos_{n,p}$ by 1

It can be checked that for $P \leq 4$ with a near-equal distribution of R in r_p, no overflow (i.e. bit in excess) in tables t_l can happen at this step (for p=0,..,P-1). In general this property must be verified to apply the algorithm.

3) Split and write all indices:

This is the tricky part of the multiplexing algorithm due to the possibility of overflow.

Find the positions pos^{ovf}_p in each binary table t_p (with p = 1…P) from which the bits in overflow can be written. These positions are computed assuming the indices are written by 4-bit block.

For p = 0..P-1

$pos = 0$

$nb = pos_{n,p} + 1$

For $k = p$ to *last* with a step of P

If $n_k > 0$,

If $4n_k \leq nb$, $nb_1 = n_k$

else $nb_1 = nb >> 2$ (where >> is a bit shift operator)

$nb = nb - 4 * nb_1$

$pos = pos + nb_1$

$pos^{ovf}_p = pos*4$

The indices can then be written as follows:

For $p = 0..P-1$

$pos = 0$

For $l = p$ to $N-1$ with a step of P

$nb = pos_{n,p} - pos$

Write the $4n_k$ bits of i_k:

Compute the number, nb_1, of 4-bit blocks which can fit in table $\mathbf{t}_p$ and the number, nb_2, of 4-bit blocks in excess (to be written temporarily in table $\mathbf{t}_{ex}$):

If $4n_k \leq nb$, $nb_1 = n_k$, $nb_2 = 0$

else $nb_1 = nb >> 2$ (where >> is a bit shift operator), $nb_2 = n_k - nb_1$

Write upward the $4nb_1$ bits of i_k from $pos_{i,p}$ to $pos_{i,p}+4nb_1-1$ in table $\mathbf{t}_p$, and increment $pos_{i,p}$ by $4nb_1$

If $nb_2 \geq 0$,

Initialize pos_{ovf} to 0

Write upward the remaining $4nb_2$ bits of i_k from pos_{ovf} to $pos_{ovf}+4nb_2-1$ in table $\mathbf{t}_{ex}$, and increment pos_{ovf} by $4nb_{ovf}$

Distribute the $4nb_2$ bits in table t_p (with $q \neq p$) based on the pointers pos^{ovf}_q and $pos_{n,q}$ and the pointers pos^{ovf}_q are updated.

5.6.2 Packetization procedure for all parameters

The coding parameters computed in a 1024-sample super-frame at the encoder are multiplexed into 4 binary packets of equal size. The packetization consists of a multiplexing loop over 4 iterations. The size of each packet is set to $R_{total} / 4$ where R_{total} is the number of bits allocated to the super-frame.

Recall that the mode selected in the 1024-sample super-frame has the form (m_1, m_2, m_3, m_4), where m_k=0, 1, 2 or 3, with the mapping: 0 → 256-sample ACELP, 1 → 256-sample TCX, 2 → 512-sample TCX, 3 → 1024-sample TCX

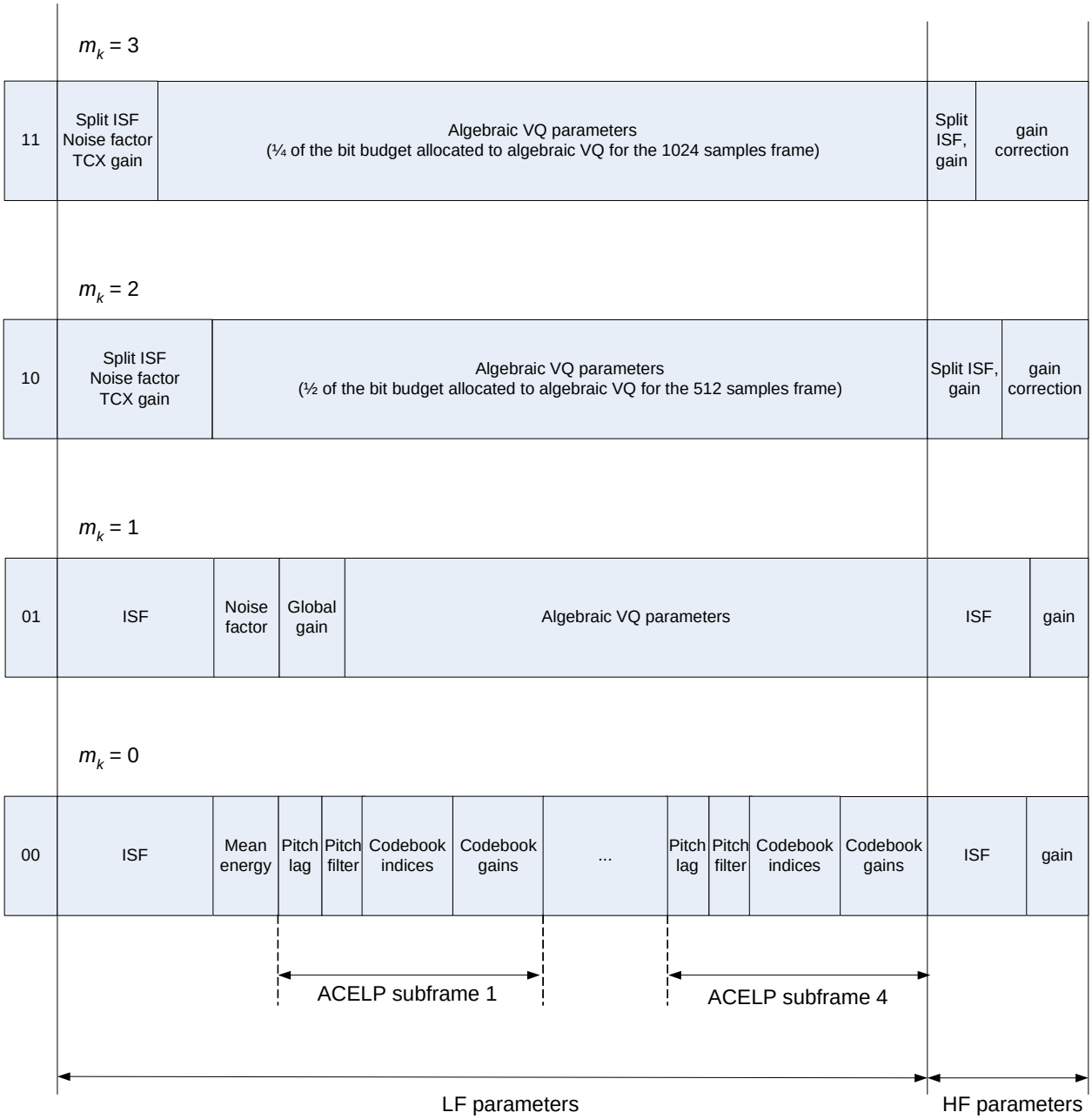


Figure 12: Structure of transmission packets for all four frame types

The multiplexing in the *k*-th packet is performed according to the value of m_k. The corresponding packet format is shown in Figure 12. There are 3 cases:

If m_k=0 or 1, the *k*-th packet simply contains all parameters related to a 256-sample frame, where the parameters are the 2-bit mode information ('00' or '01' in binary format), the parameters of ACELP or those of 256-sample TCX, and the parameters of 256-sample HF coding.

If m_k=2, the *p*-th packet contains half of the bits of the 512-sample TCX mode, half of the bits of 512-sample HF coding, plus the 2-bit mode information ('10' in binary format).

If m_k=3, the *k*-th packet contains one fourth of the bits describing the 512-sample TCX mode, one fourth of the bits of 1024-sample HF coding, plus the 2-bit mode information ('11' in binary format).

The packetization is therefore straightforward if the *k*-th packet corresponds to ACELP or 256-sample TCX. The packetization is slightly more involved if 512- or 1024-sample TCX mode is used, because the bits of the 512- or 1024-sample modes have to be shared into even parts.

5.6.3 TCX gain multiplexing

It was found that the TCX gain is important to maintain audible quality in case of packet loss. Thus, in 512-sample and 1024-sample TCX frames, the TCX gain value is encoded redundantly in multiple packets to protect against packet loss. The TCX gain is encoded at a resolution of 7 bits, and these bits are labelled "Bit 0" to "Bit 6", where "Bit 0" is the Least Significant Bit (LSB) and "Bit 6" is the Most Significant Bit (MSB). We consider two cases, TCX512 and TCX1024, where the encoded bits are split into two or four packets, respectively.

At the Encoder side

TCX512: The first packet contains the full gain information (7 bits). The second packet repeats the most significant 6 bits ("Bit 1" to "Bit 7").

TCX1024: The first packet contains the full gain information (7 bits). The third packet contains a copy of the three bits "Bit 4", "Bit 5" and "Bit 6". The fourth packet contains a copy of the three bits "Bit 1", "Bit 2" and "Bit 3".

Additionally, a 3-bit "parity" is formed as thus: combining by logical XOR "Bit 1" and "Bit 4" to generate "Parity Bit 0", combining by logical XOR "Bit 2" and "Bit 5" to generate "Parity Bit 1", and combining by logical XOR "Bit 3" and "Bit 6" to generate "Parity Bit 2". These three parity bits are sent in the second packet.

At the Decoder side

The logic applied at the decoder to recover the TCX gain when missing packets occur for 512-sample TCX and 1024-sample TCX. We assume that there is at least one packet missing before entering the flowchart.

TCX512: If the fist packet is flagged as being lost, the TCX global gain is taken from the second packet, with the LSB ("Bit 0") being set to zero. If only the second packet is lost, then the full TCX gain is obtained from the first packet.

TCX1024: The gain recovery algorithm is only used if 1 or 2 packets forming an 1024-sample TCX frame are lost; as described in Section 6.5.1.1. If 3 or more packets are lost in a TCX1024 frame, the MODE is changed to (1,1,1,1) and BFI=(1,1,1,1). When only 1 or 2 packets are lost in a TCX1024 frame, the recovery algorithm is as follows:

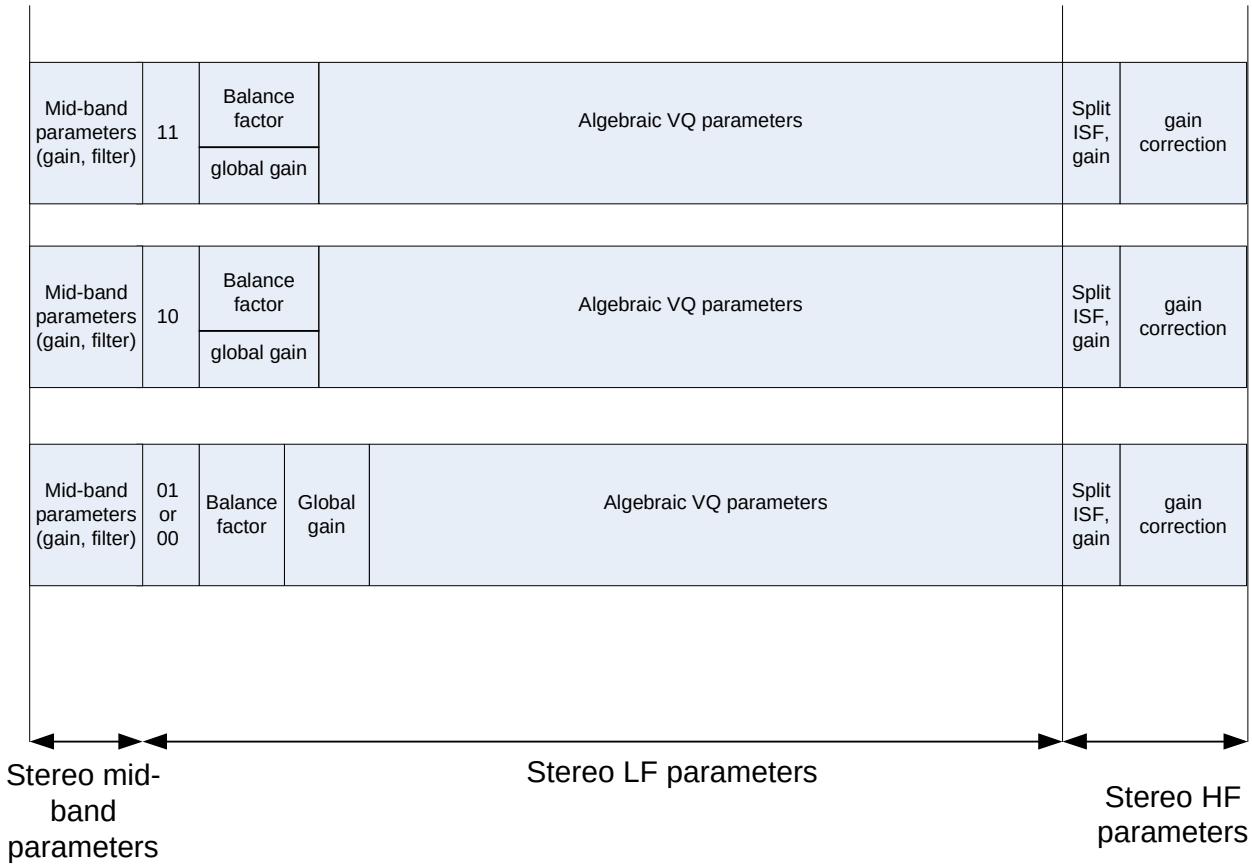
As described above, the second, third and fourth packets of a TCX1024 frame contain the parity bits, "Bit 6" to "Bit 4", and "Bit 3" to "Bit 1" of the TCX gain. These bits (three each) are stored in "parity", "index0" and "index1" respectively.

If the third packet is lost, "index0" is replaced by the logical XOR combination of "parity" and "index1". That is, "Bit 6" is generated from the logical XOR of "Parity Bit 2" and "Bit 3", "Bit 5" is generated from the logical XOR of "Parity Bit 1" and "Bit 2", and "Bit 4" is generated from the logical XOR of "Parity Bit 0" and "Bit 1".

If the fourth packet is lost, "index1" is replaced by the logical XOR combination of "parity" and "index0. That is, "Bit 3" is generated from the logical XOR of "Parity Bit 2" and "Bit 6", "Bit 2" is generated from the logical XOR of "Parity Bit 1" and "Bit 5", and "Bit 1" is generated from the logical XOR of "Parity Bit 0" and "Bit 4".

Finally, the 7-bit TCX gain value is taken from the recovered bits ("Bit 1" to "Bit 6") and "Bit 0" is set to zero.

5.6.4 Stereo Packetization



Stereo parameters computed in a 1024-sample super-frame at the encoder are multiplexed into 4 binary packets of equal size. The packetization consists of a similar multiplexing loop as for the core encoder. The stereo packets are appended at the end of the mono packets.

6 Functional description of the decoder

The function of the decoder consists of decoding the transmitted parameters (LP parameters, ACELP/TCX mode, adaptive codebook vector, adaptive codebook gain, fixed codebook vector, fixed codebook gain, TCX parameters, high-band parameters, stereo information) and performing synthesis to obtain the reconstructed low-frequency and high-frequency signals. For stereo signal synthesis, the stereo low- and mid-band signals are reconstructed using the low-frequency mono signal and the transmitted and decoded stereo parameters (…).

Section 6.1 describes the reconstruction, by the decoder, of the mono low-band signal in the 0-Fs/4 kHz bandwidth (core ACELP/TCX decoder). The reconstruction of the higher frequency band using bandwidth extension and the mixing of the low and high frequencies of the mono signal will be described respectively in Sections 6.2 and 6.3. The generation of the stereo signals will be described in Section 6.4. Finally, Section 6.5 describes the concealment algorithm in the case of missing frames.

6.1 Mono Signal Low-Band synthesis

The 0-Fs/4 kHz band of the mono signal is reconstructed by the core ACELP/TCX decoder. In ACELP mode, the decoder is the same as AMR-WB. The TCX mode of the decoder will be described in more details below. Selection between ACELP and TCX decoding in each 256-sample frame is controlled by the mode indicators described in Section 5.2.2 of the encoder. These mode indicators are transmitted as 2 bits in each 256-sample packet.

6.1.1 ACELP mode decoding and signal synthesis

Same as 3GPP TS 26.190.

6.1.2 TCX mode decoding and signal synthesis

The TCX decoder is shown in Figure 13.

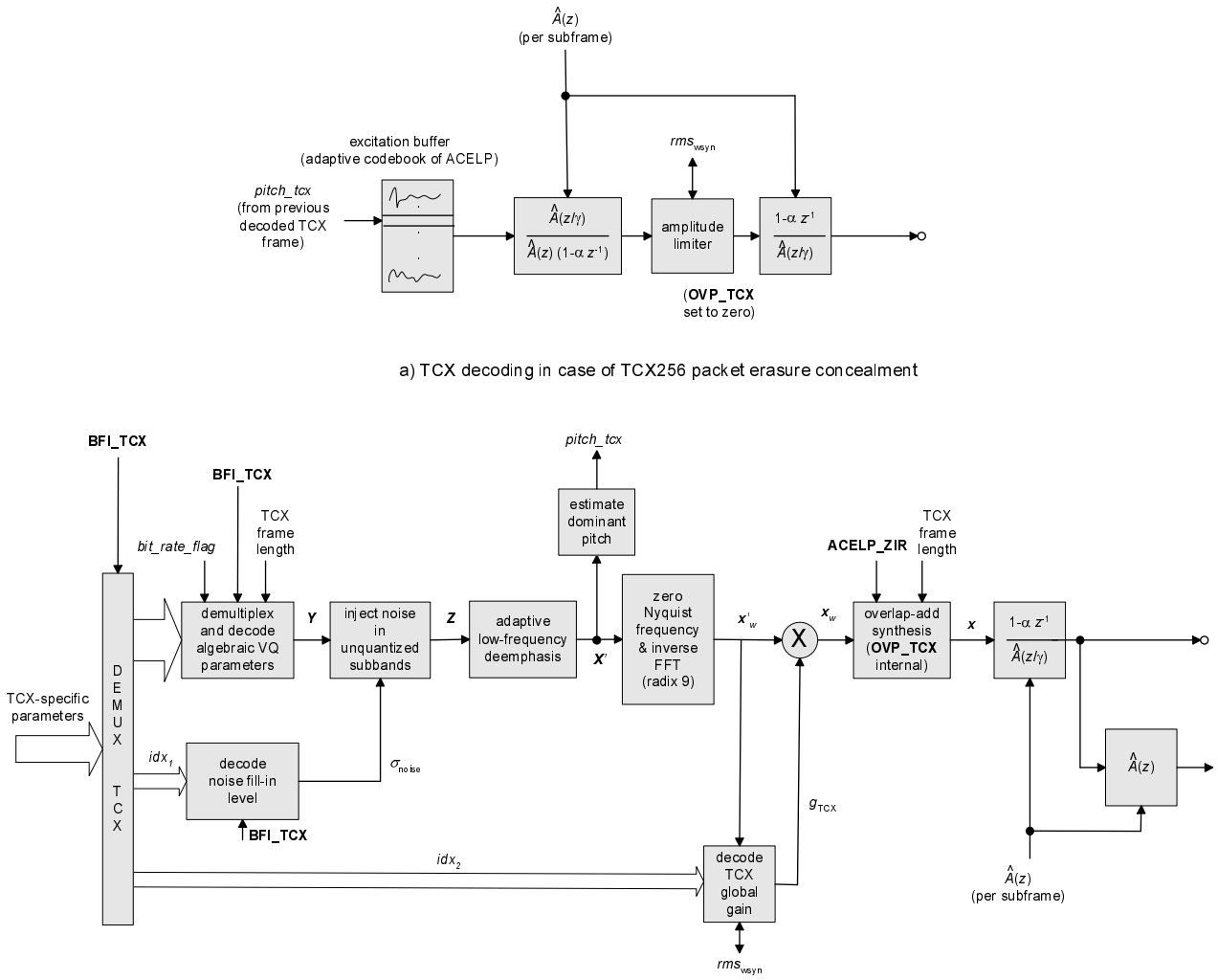


Figure 13: Block diagram of the TCX decoder

Figure 13 shows a block diagram of the TCX decoder including the following two cases:

Case 1: Packet-erasure concealment in TCX-256 when the TCX frame length is 256 samples and the related packet is lost i.e. **BFI_TCX** = (1), as shown in Figure 13-a.

Case 2: Normal TCX decoding, possibly with partial packet losses, as shown in Figure 13-b..

In Case 1, no information is available to decode the 256-sample TCX frame. The TCX synthesis is found by processing the past excitation delayed by *T*, where *T=pitch_tcx* is a pitch lag estimated in the previously decoded TCX frame, by a non-linear filter roughly equivalent to $1/\hat{A}(z)$. A non-linear filter is used instead of $1/\hat{A}(z)$ to avoid clicks in the synthesis. This filter is decomposed in 3 steps:

Step 1: filtering by

$$\frac{\hat{A}(z/\gamma)}{\hat{A}(z)} \quad \frac{1}{1-\alpha z^{-1}}$$

to map the excitation delayed by *T* into the TCX target domain;

Step 2: applying a limiter (the magnitude is limited to ± rms_{wsyn})

Step 3: filtering by

$$\frac{1-\alpha z^{-1}}{\hat{A}(z/\gamma)}$$

to find the synthesis. Note that the buffer **OVLP_TCX** is set to zero in this case.

Decoding of the algebraic VQ parameters

In Case 2, TCX decoding involves decoding the algebraic VQ parameters describing each quantized block $\hat{B}'_k$ of the scaled spectrum *X'*, where *X'* is as described in Step 2 of Section 5.3.5.7. Recall that ***X'*** has dimension *N*, where *N* = 288, 576 and 1152 for TCX-256, 512 and 1024 respectively, and that each block B'_k has dimension 8. The number *K* of blocks B'_k is thus 36, 72 and 144 for TCX-256, 512 and 1024 respectively. The algebraic VQ parameters for each block B'_k are described in Step 5 of Section 5.3.5.7. For each block B'_k , three sets of binary indices are sent by the encoder:

a) the codebook index n_k, transmitted in unary code as described in Step 5 of Section 5.3.5.7;

b) the rank I_k of a selected lattice point *c* in a so-called *base codebook*, which indicates what permutation has to be applied to a specific *leader* (see Step 5 of Section 5.3.5.7) to obtain a lattice point ***c***;

c) and, if the quantized block $\hat{B}'_k$ (a lattice point) was not in the base codebook, the 8 indices of the Voronoi extension index vector ***k*** calculated in sub-step V1 of Step 5 in Section; from the Voronoi extension indices, an extension vector ***z*** can be computed as in reference [7]. The number of bits in each component of index vector ***k*** is given by the extension order *r*, which can be obtained from the unary code value of index n_k . The scaling factor *M* of the Voronoi extension is given by $M = 2^r$.

Then, from the scaling factor *M*, the Voronoi extension vector ***z*** (a lattice point in RE_8) and the lattice point ***c*** in the base codebook (also a lattice point in RE_8), each quantized scaled block $\hat{B}'_k$ can be computed as

$$\hat{B}'_k = M\,\mathbf{c} + \boldsymbol{z}$$

When there is no Voronoi extension (i.e. $n_k < 5$, M=1 and z=0), the base codebook is either codebook Q_0, Q_2, Q_3 or Q_4 from reference [6]. No bits are then required to transmit *vector* ***k***. Otherwise, when Voronoi extension is used because $\hat{B}'_k$ is large enough, then only Q_3 or Q_4 from reference [6] is used as a base codebook. The selection of Q_3 or Q_4 is implicit in the codebook index value n_k,, as described in Step 5 of Section 5.3.5.7.

Decoding of the noise-fill parameter

The noise fill-in level σ_{noise} is decoded by inverting the 3-bit uniform scalar quantization calculated at the encoder as in Step 4 of Section 5.3.5.7 . For an index $0 \le idx_1 \le 7$, σ_{noise} is given by: $\sigma_{noise} = 0.1 * (8 - idx_1)$. However, it may happen that the index idx_1 is not available. This is the case when **BFI_TCX** = (1) in TCX-256, (1 *X*) in TCX-512 and (X 1 X *X*) in TCX-1024, with X representing an arbitrary binary value. In this case, σ_{noise} is set to its maximal value, i.e. σ_{noise} = 0.8.

Comfort noise is injected in the subvectors B_k rounded to zero and which correspond to a frequency above Fs/2 kHz 4. More precisely, ***Z*** is initialized as ***Z*** = ***Y*** and for $K/6 \le k \le K$ (only), if $\mathbf{Y}_k$ = (0, 0, …,0), $\mathbf{Z}_k$ is replaced by the 8-dimensional vector:

$$\sigma_{noise} * [\ \cos(\theta_1)\ \sin(\theta_1)\ \cos(\theta_2)\ \sin(\theta_2)\ \cos(\theta_3)\ \sin(\theta_3)\ \cos(\theta_4)\ \sin(\theta_4)\ ],$$

where the phases θ_1, θ_2, θ_3 and θ_4 are randomly selected.

Low-frequency de-emphasis

After decoding the algebraic VQ parameters and noise-fill parameter, we obtain the quantized pre-shaped TCX spectrum X'. De-shaping is then applied as in Section 5.3.5.6.

Estimation of the dominant pitch value

The estimation of the dominant pitch is performed so that the next frame to be decoded can be properly extrapolated if it corresponds to TCX-256 and if the related packet is lost. This estimation is based on the assumption that the peak of maximal magnitude in spectrum of the TCX target corresponds to the dominant pitch. The search for the maximum M is restricted to a frequency below Fs/64 kHz

$$M = \max_{i=1..N/32} (X'_{2i})^2 + (X'_{2i+1})^2$$

and the minimal index $1 \le i_{max} \le N/32$ such that $(X'_{2i})^2 + (X'_{2i+1})^2 = M$ is also found. Then the dominant pitch is estimated in number of samples as $T_{est} = N / i_{max}$ (this value may not be integer). Recall that the dominant pitch is calculated for packet-erasure concealment in TCX-256. To avoid buffering problems (the excitation buffer being limited to 256 samples), if $T_{est} > 256$ samples, *pitch_tcx* is set to 256 ; otherwise, if $T_{est} \le 256$, multiple pitch period in 256 samples are avoided by setting *pitch_tcx* to

$$pitch\_tcx = \max \{ \lfloor n\, T_{est} \rfloor \mid n \text{ integer} > 0 \text{ and } n\, T_{est} \le 256 \}$$

where $\lfloor . \rfloor$ denotes the rounding to the nearest integer towards -∞.

Inverse transform

To obtain the quantized perceptual signal, an inverse transform is applied to the de-shaped spectrum X'. The transform used at the encoder and decoder is a the discrete Fourier transform, and is implemented as an FFT and IFFT, respectively. Recall that due to the ordering used at the TCX encoder, the transform coefficients $\mathbf{X}'=(X'_0,\ldots,X'_{N-1})$ are such that:

X'_0 corresponds to the DC coefficient,

X'_1 corresponds to the Nyquist frequency, and

the coefficients X'_{2k} and X'_{2k+1}, for k=1..N/2-1, are the real and imaginary parts of the Fourier component of frequency of $k(/N/2)$ * Fs/4 kHz.

X'_1 is always forced to 0. After this zeroing, the time-domain TCX target signal $\boldsymbol{x}'_w$ is found by applying an inverse FFT to the quantized scaled spectrum X. Rescaling will be applied in the following section, to obtain the total quantized weighted signal prior to windowing and overlapping.

Decoding of the glocal TCX gain and scaling

The (global) TCX gain g_{TCX} is decoded by inverting the 7-bit logarithmic quantization calculated in the TCX encoder as in Section 5.2.5.10 . First, the r.m.s. value of the TCX target signal $\boldsymbol{x}'_w$ is computed as:

$$rms = \mathrm{sqrt}(1/N\, (x'_{w0}{}^2 + x'_{w1}{}^2 + \ldots + x'_{wL-1}{}^2))$$

From the received 7-bit index $0 \le idx_2 \le 127$, the TCX gain is given by:

$$g_{TCX} = 10^{idx_2 / 28 / (4 \times rms)}$$

The (logarithmic) quantization step is around 0.71 dB.

This gain is used to scale $\boldsymbol{x}'_w$ into $\boldsymbol{x}_w$. Note that from the mode extrapolation and the gain repetition strategy, the index idx_2 is available in case of frame loss. However, in case of partial packet losses (1 loss for TCX-512 and up to 2 losses for TCX-1024) the least significant bit of idx_2 may be set by default to 0 in the demultiplexer.

Windowing and overlap

Since the TCX encoder employs windowing with overlap and weighted ZIR removal prior to transform coding of the target signal, the reconstructed TCX target signal $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})$ is actually found by overlap-add. The overlap-add depends on the type of the previous decoded frame (ACELP or TCX). The TCX target signal is first multiplied by a window $\mathbf{w} = [w_0\ w_1\ \ldots\ w_{N-1}]$, whose shape is described in Section 5.3.5.4.

Then, the overlap from the past decoded frame (**OVLP_TCX**) is added to the present windowed signal **x**. The overlap length OVLP_TCX depends on the past TCX framelength and on the mode of the past frame (ACELP or TCX).

Computation of the synthesis signal

The reconstructed TCX target is then filtered through the zero-state inverse perceptual filter $W^{-1}(z) = (1 - \alpha z^{-1}) / \hat{A}(z/\gamma)$ to find the synthesis signal which will be applied to the synthesis filter. The excitation is also calculated to update the ACELP adaptive codebook and allow to switch from TCX to ACELP in a subsequent frame. Note that the length of the TCX synthesis is given by the TCX frame length (without the overlap): 256, 512 or 1024 samples.

6.1.3 Post-processing of Mono Low-Band signal

In the low-frequency pitch enhancement, two-band decomposition is used and adaptive filtering is applied only to the lower band. This results in a total post-processing that is mostly targeted at frequencies near the first harmonics of the synthesized speech signal.

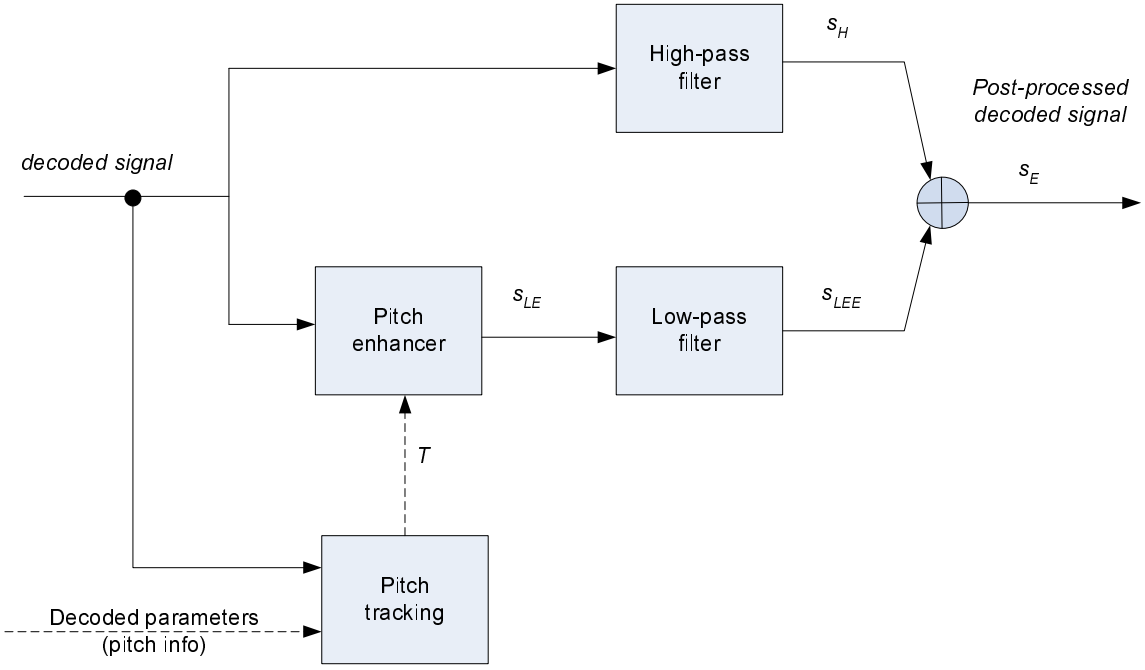


Figure 14: Block diagram of the low frequency pitch enhancer

Figure 14 shows the block diagram of the two-band pitch enhancer. In the higher branch the decoded signal is filtered by a high-pass filter to produce the higher band signal (s_H). In the lower branch, the decoded signal is first processed through an adaptive pitch enhancer, and then filtered through a low-pass filter to obtain the lower band post-processed signal (s_{LEF}). The post-processed decoded signal is obtained by adding the lower band post-processed signal and the higher band signal. The object of the pitch enhancer is to reduce the inter-harmonic noise in the decoded signal, which is achieved here by a time-varying linear filter with a transfer function

$$H_E(z) = (1-\alpha) + \frac{\alpha}{2} z^{T} + \frac{\alpha}{2} z^{-T}$$

and described by the following equation:

$$s_{LE}(n) = (1-\alpha)\hat{s}(n) + \frac{\alpha}{2}\hat{s}(n-T) + \frac{\alpha}{2}\hat{s}(n+T) \qquad (1)$$

where α is a coefficient that controls the inter-harmonic attenuation, T is the pitch period of the input signal $\hat{s}(n)$, and $s_{LE}(n)$ is the output signal of the pitch enhancer. Parameters T and α vary with time and are given by the pitch tracking module. With a value of α = 1, the gain of the filter described by Equation (1) is exactly 0 at frequencies

1/(2*T*),3/(2*T*), 5/(2*T*), etc.; i.e. at the mid-point between the harmonic frequencies 1/*T*, 3/*T*, 5/*T*, etc. When α approaches 0, the attenuation between the harmonics produced by the filter of Equation (1) decreases.

To confine the post-processing to the low frequency region, the enhanced signal s_{LE} is low pass filtered to produce the signal s_{LEF} which is added to the high-pass filtered signal s_H to obtain the post-processed synthesis signal s_E.

Another configuration equivalent to the one in Figure 14 is used here which eliminates the need to high-pass filtering. This is explained as follows.

Let $h_{LP}(n)$ be the impulse response of the low-pass filter and $h_{HP}(n)$ is the impulse response of the complementary high-pass filter. The post-processed signal $s_E(n)$ is given by

$$
\begin{aligned}
s_E(n) &= \hat{s}(n) * h_{HP}(n) + s_{LE}(n) * h_{LP}(n) \\
&= \hat{s}(n) * h_{HP}(n) + \left((1-\alpha)\hat{s}(n) + \frac{\alpha}{2}\hat{s}(n-T) + \frac{\alpha}{2}\hat{s}(n+T) \right) * h_{LP}(n) \\
&= \hat{s}(n) * h_{HP}(n) + \hat{s}(n) * h_{LP}(n) - \left(\alpha\hat{s}(n) - \frac{\alpha}{2}\hat{s}(n-T) - \frac{\alpha}{2}\hat{s}(n+T) \right) * h_{LP}(n) \\
&= \hat{s}(n) - \alpha \left(\hat{s}(n) - \frac{1}{2}\hat{s}(n-T) - \frac{1}{2}\hat{s}(n+T) \right) * h_{LP}(n) \\
&= \hat{s}(n) - \alpha\, e_{LT}(n) * h_{LP}(n)
\end{aligned}
$$

Thus, the post-processing is equivalent to subtracting the scaled low-pass filtered long-term error signal from the synthesis signal $\hat{s}(n)$. The transfer function of the long-term prediction filter is given by

$$P_{LT}(z) = 1 - 0.5z^{T} - 0.5z^{-T}$$

The alternative post-processing configuration is depicted in Figure 15.

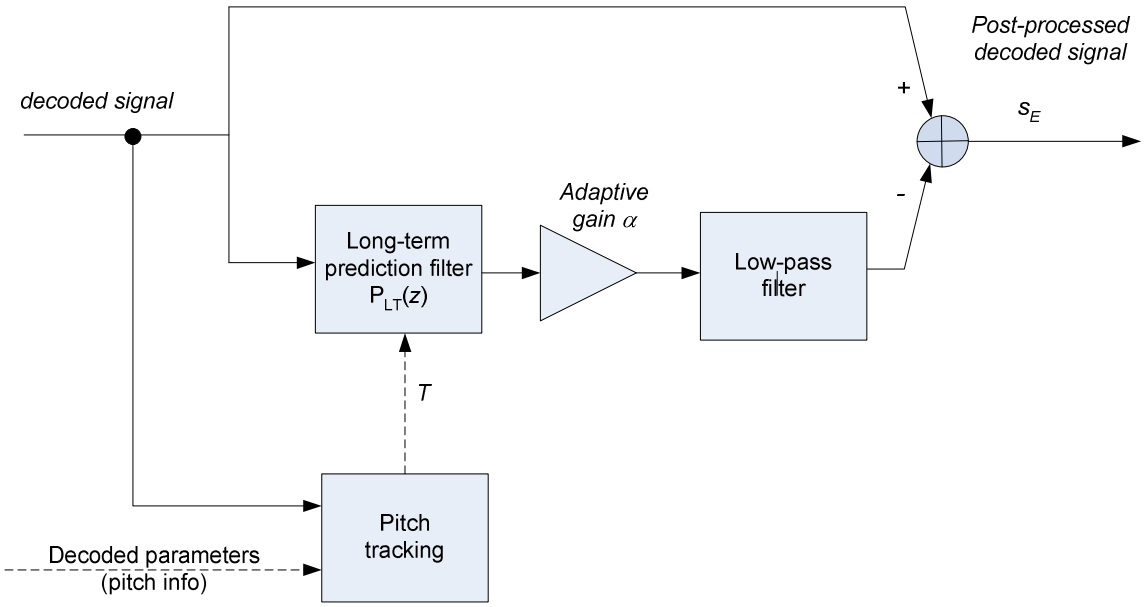


Figure 15: Implemented post-processing configuration

The value *T* is given by the received closed-loop pitch lag in each subframe (the fractional pitch lag rounded to the nearest integer). A simple tracking for checking pitch doubling is performed. If the normalized pitch correlation at delay T/2 is larger than 0.95 then the value T/2 is used as the new pitch lag for post-processing.

The factor α is by

$$\alpha = 0.5 g_p, \quad \text{constrained to } 0 \le \alpha \le 0.5$$

where g_p is the decoded pitch gain. Note that in TCX mode the value of α is set to zero.

A linear phase FIR low-pass filter with 25 coefficients is used, with a cut-off frequency at 5Fs/256 kHz (the filter delay is 12 samples).

6.2 Mono Signal High-Band synthesis

The synthesis of the HF signal implements a kind of bandwidth extension (BWE) mechanism and uses some data from the LF decoder. It is an evolution of the BWE mechanism used in the AMR-WB speech decoder. The HF decoder is detailed in Figure 16. The HF signal is synthesized in 2 steps: calculation of the HF excitation signal and computation of the HF signal from the HF excitation. The HF excitation is obtained by shaping in time-domain the LF excitation signal with scalar factors (or gains) per 64-sample subframes. This HF excitation is post-processed to reduce the "buzziness" of the output, and then filtered by a HF linear-predictive synthesis filter $1/A_{HF}(z)$. Recall that the LP order used to encode and then decode the HF signal is 8. The result is also post-processed to smooth energy variations.

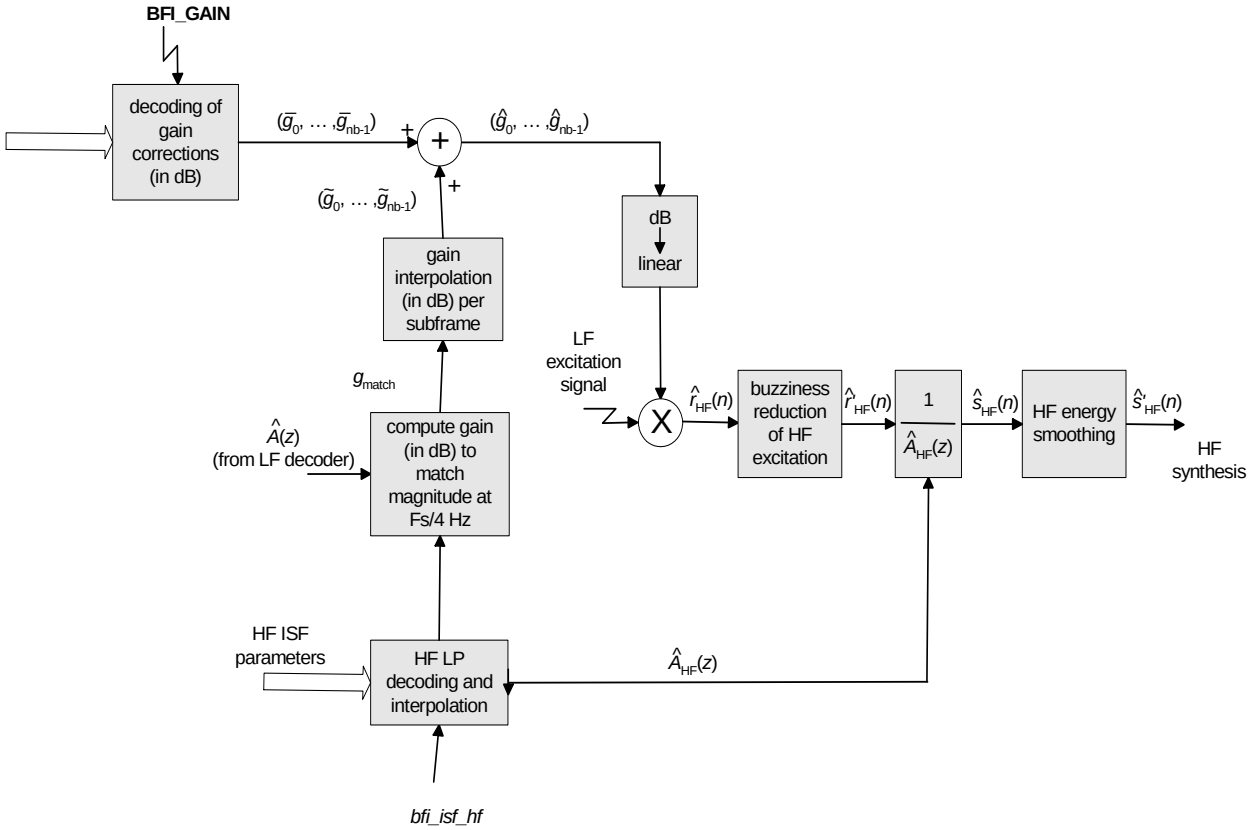


Figure 16: Block diagram of high frequency decoder

The HF decoder synthesizes an 1024-sample HF superframe. This superframe is segmented according to **MODE** = (m_0, m_1, m_2, m_3). To be more specific, the decoded frames used in the HF decoder are synchronous with the frames used in the LF decoder. Hence, $m_k \le 1$, $m_k = 2$ and $m_k = 3$ indicate respectively a 256, 512 and 1024-sample frame. These frames are referred to as HF-256, HF-512 and HF-1024, respectively.

From the synthesis chain described above, it is clear that the only parameters needed for HF decoding are ISF and gain parameters. The ISF parameters represent the filter $1/A_{HF}(z)$, while the gain parameters are used to shape the LF excitation signal. These parameters are demultiplexed based on **MODE** and knowing the format of the bitstream.

Control data which are internal to the HF decoder are generated from the bad frame indicator vector **BFI** = (bfi_0, bfi_1, bfi_2, bfi_3). These data are *bfi_isf_hf*, **BFI_GAIN**, and the number of subframes for ISF interpolation. The nature of these data is defined in more details below:

bfi_isf_hf is a binary flag indicating loss of the ISF parameters. Its definition is given below from **BFI**.

For HF-256 in packet *k*, $bfi\_isf\_hf = bfi_k$,

For HF-512 in packets *k* and *k+1*, $bfi\_isf\_hf = bfi_k$,

For HF-1024 (in packets k=*0 to 3*), $bfi\_isf\_hf = bfi_0$

This definition can be readily understood from the bitstream format. Recall that the ISF parameters for the HF signal are always in the first packet describing HF-256, -512 or –1024 frames.

BFI_GAIN is a binary vector used to signal packet losses to the HF gain decoder: **BFI_GAIN** = (bfi_k) for HF-256 in packet *k*, (bfi_k bfi_{k+1}) for HF-512 in packets *k* and *k+1*, **BFI_GAIN** = **BFI** for HF-1024.

The number of subframes for ISF interpolation refers to the number of 64-sample subframes in the decoded frame. This number is 4 for HF-256, 8 for HF-512 and 16 for HF-1024.

The ISF vector **isf_hf_q** is decoded using AR(1) predictive VQ. If *bfi_isf_hf* = 0, the 2-bit index i_1 of the 1st stage and the 7-bit index i_2 of the 2nd stage are available and **isf_hf_q** is given by

$$\mathbf{isf\_hf\_q} = \mathbf{cb1}(i_1) + \mathbf{cb2}(i_2) + \mathbf{mean\_isf\_hf} + \mu_{isf\_hf} * \mathbf{mem\_isf\_hf}$$

where $\mathbf{cb1}(i_1)$ is the i_1–th codevector of the 1st stage, $\mathbf{cb2}(i_2)$ is the i_2–th codevector of the 2st stage, **mean_isf_hf** is the mean ISF vector, $\mu_{isf\_hf} = 0.5$ is the AR(1) prediction coefficient and **mem_isf_hf** is the memory of the ISF predictive decoder.

If *bfi_isf_hf* = 1, the decoded ISF vector corresponds to the previous ISF vector shifted towards the mean ISF vector:

$$\mathbf{isf\_hf\_q} = \alpha_{isf\_hf} * \mathbf{mem\_isf\_hf} + \mathbf{mean\_isf\_hf}$$

with $\alpha_{isf\_hf} = 0.9$. After calculating **isf_hf_q**, the ISF reordering defined in AMR-WB speech coding is applied to **isf_hf_q** with an ISF gap of 9 Fs/1280 Hz. Finally the memory **mem_isf_hf** is updated for the next HF frame as:

$$\mathbf{mem\_isf\_hf} = \mathbf{isf\_hf\_q} - \mathbf{mean\_isf\_hf}$$

Note that the initial value of **mem_isf_hf** (at the reset of the decoder) is zero.

A simple linear interpolation between the ISP parameters of the previous decoded HF frame (HF-256, HF-512 or HF-1024) and the new decoded ISP parameters is performed. The interpolation is conducted in the ISP domain and results in ISP parameters for each 64-sample subframe, according to the formula:

$$\mathbf{isp}_{\text{subframe-}i} = i/nb * \mathbf{isp}_{new} + (1-i/nb) * \mathbf{isp}_{old},$$

where *nb* is the number of subframes in the current decoded frame (*nb*=4 for HF-256, 8 for HF-512, 16 for HF-1024), *i*=0,…,*nb*-1 is the subframe index, $\mathbf{isp}_{old}$ is the set of ISP parameters obtained from the ISF parameters of the previously decoded HF frame and $\mathbf{isp}_{new}$ is the set of ISP decoded. The interpolated ISP parameters are then converted into linear-predictive coefficients for each subframe.

The computation of the gain g_{match} in dB is detailed in the next paragraphs. This gain is interpolated for each 64-sample subframe based on its previous value old_g_{match} as:

$$\tilde{g}_i = i/nb * g_{match} + (1-i/nb) * old\_g_{match},$$

where *nb* is the number of subframes in the current decoded frame (*nb*=4 for HF-256, 8 for HF-512, 16 for HF-1024), *i*=0,…,*nb*-1 is the subframe index. This results in a vector ($\tilde{g}_0$, … $\tilde{g}_{nb-1}$).

Gain estimation computation to match magnitude at Fs/4 kHz

Same as section 5.6 (Figure 9)

Decoding of correction gains and gain computation

Recall that after gain interpolation the HF decoder gets the estimated gains (g^{est}_0, g^{est}_1, …, g^{est}_{nb-1}) in dB for each of the *nb* subframes of the current decoded frame. Furthermore, *nb* = 4, 8 and 16 in HF-256, -512 and –1024, respectively.

The correction gains in dB are then decoded which will be added to the estimated gains per subframe to form the decode gains $\hat{g}_0, \hat{g}_1, \ldots, \hat{g}_{nb-1}$:

$(\hat{g}_0\,(dB), \hat{g}_1\,(dB), \ldots, \hat{g}_{nb-1}(dB)) = (\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{nb-1}) + (\bar{g}_0, \bar{g}_1, \ldots, \bar{g}_{nb-1})$

where

$(\bar{g}_0, \bar{g}_1, \ldots, \bar{g}_{nb-1}) = (g^{c1}_1, g^{c1}_1, \ldots, g^{c1}_{nb-1}) + (g^{c2}_0, g^{c2}_1, \ldots, g^{c2}_{nb-1})$.

Therefore, the gain decoding corresponds to the decoding of predictive two-stage VQ-scalar quantization, where the prediction is given by the interpolated Fs/4 kHz junction matching gain. The quantization dimension is variable and is equal to *nb*.

Decoding of the 1st stage:

The 7-bit index $0 \le idx \le 127$ of the 1st stage 4-dimensional HF gain codebook is decoded into 4 gains (G_0, G_1, G_2, G_3). A bad frame indicator $bfi = BFI\_GAIN_0$ in HF-256, -512 and –1024 allows to handle packet losses. If $bfi = 0$, these gains are decoded as

(G_0, G_1, G_2, G_3) = **cb_gain_hf**(idx) + *mean_gain_hf*

where **cb_gain_hf**(*idx*) is the *idx*-th codevector of the codebook **cb_gain_hf**. If *bfi* =1, a memory *past_gain_hf_q* is shifted towards –20 dB:

past_gain_hf_q:= α_{gain_hf} * (*past_gain_hf_q + 20) – 20.*

where $\alpha_{gain\_hf} = 0.9$ and the 4 gains (G_0, G_1, G_2, G_3) are set to the same value:

G_k = *past_gain_hf_q* + *mean_gain_hf*, for $k = 0,1,2$ and 3

Then the memory *past_gain_hf_q* is updated as:

past_gain_hf_q:= ($G_0 + G_1 + G_2 + G_3$)/4 - *mean_gain_hf.*

The computation of the 1st stage reconstruction is then given as:

HF-256: $(g^{c1}_0, g^{c1}_1, g^{c1}_2, g^{c1}_3) = (G_0, G_1, G_2, G_3)$.

HF-512: $(g^{c1}_0, g^{c1}_1, \ldots, g^{c1}_7) = (G_0, G_0, G_1, G_1, G_2, G_2, G_3, G_3)$.

HF-1024: $(g^{c1}_0, g^{c1}_1, \ldots, g^{c1}_{15}) = (G_0, G_0, G_0, G_0, G_1, G_1, G_1, G_1, G_2, G_2, G_2, G_2, G_3, G_3, G_3, G_3)$.

Decoding of 2nd stage:

In TCX-256, ($g^{c2}_0, g^{c2}_1, g^{c2}_2, g^{c2}_3$) is simply set to (0,0,0,0) and there is no real 2nd stage decoding. In HF-512, the 2-bit index $0 \le idx_i \le 3$ of the *i*-th subframe, where $i=0, \ldots, 7$, is decoded as:

If $bfi = 0$, $g^{c2}_i = 3 * idx_i - 4.5$ else $g^{c2}_i = 0$.

In TCX-1024, 16 subframes 3-bit index the $0 \le idx_i \le 7$ of the *i*-th subframe, where $i=0, \ldots, 15$, is decoded as:

If $bfi = 0$, $g^{c2}_i = 3 * idx - 10.5$ else $g^{c2}_i = 0$.

In TCX-512 the magnitude of the second scalar refinement is up to ± 4.5 dB and in TCX-1024 up to ± 10.5 dB. In both cases, the quantization step is 3 dB.

HF gain reconstruction:

The gain for each subframe is then computed as: $10^{\hat{g}_i / 20}$

Buzziness reduction and energy smoothing

The role of energy smoothing is to attenuate pulses in the time-domain HF excitation signal $r_{HF}(n)$, which often cause the audio output to sound "buzzy". Pulses are detected by checking if the absolute value $| r_{HF}(n) | > 2 * thres(n)$, where

thres(*n*) is an adaptive threshold corresponding to the time-domain envelope of $r_{HF}(n)$. The samples $r_{HF}(n)$ which are detected as pulses are limited to $\pm 2 * thres(n)$, where ± is the sign of $r_{HF}(n)$.

Each sample $r_{HF}(n)$ of the HF excitation is filtered by a 1st order low-pass filter $0.02/(1 - 0.98\, z^{-1})$ to update *thres*(*n*). Note that the initial value of *thres*(*n*) (at the reset of the decoder) is 0. The amplitude of the pulse attenuation is given by:

$\Delta = \max(|r_{HF}(n)|-2*thres(n) , 0.0)$.

Thus, Δ is set to 0 if the current sample is not detected as a pulse, which will let $r_{HF}(n)$ unchanged. Then, the current value *thres*(*n*) of the adaptive threshold is changed as:

thres(*n*):= *thres*(*n*) + 0.5 * Δ.

Finally each sample $r_{HF}(n)$ is modified to: $r'_{HF}(n) = r_{HF}(n) - \Delta$ if $r_{HF}(n) \geq 0$, and $r'_{HF}(n) = r_{HF}(n) + \Delta$ otherwise.

The short-term energy variations of the HF synthesis $s_{HF}(n)$ are then smoothed. The energy is measured by subframe. The energy of each subframe is modified by up to ± 1.5 dB based on an adaptive threshold.

For a given subframe $s_{HF}(n)$, n=0,…,63, the subframe energy is calculated as

$$\varepsilon = 0.0001 + \sum_{n=0}^{63} s_{HF}^2(n)$$

The value *t* of the threshold is updated as:

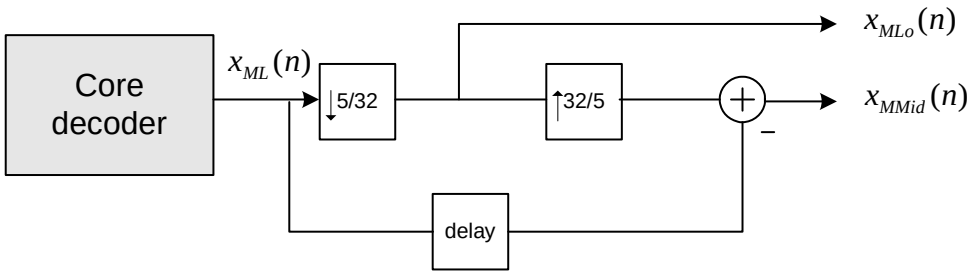
t := min(ε* 1.414, *t*), if ε < *t*

max(ε / 1.414, *t*), otherwise.

The current subframe is then scaled by $\sqrt{t/\varepsilon}$:

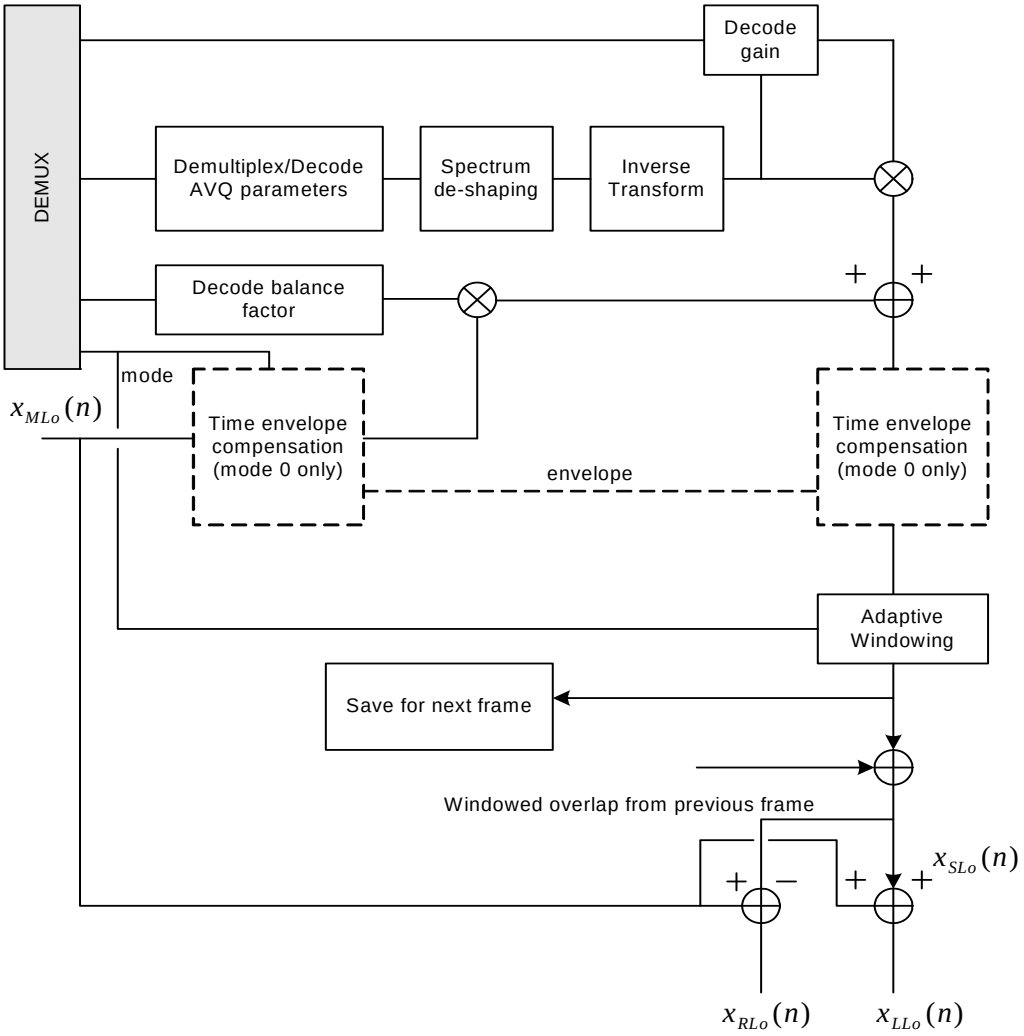
$s'_{HF}(n) = \sqrt{t/\varepsilon}\ s_{HF}(n)$, n=0,…,63

6.3 Stereo Signal synthesis



For stereo signal decoding, the mono signal is needed. First, the output of the core decoder is split into two bands , these two signals $x_{MLo}(n)$ and $x_{MMid}(n)$ are fed to the low band and high band stereo decoder.

6.3.1 Stereo signal low-band synthesis



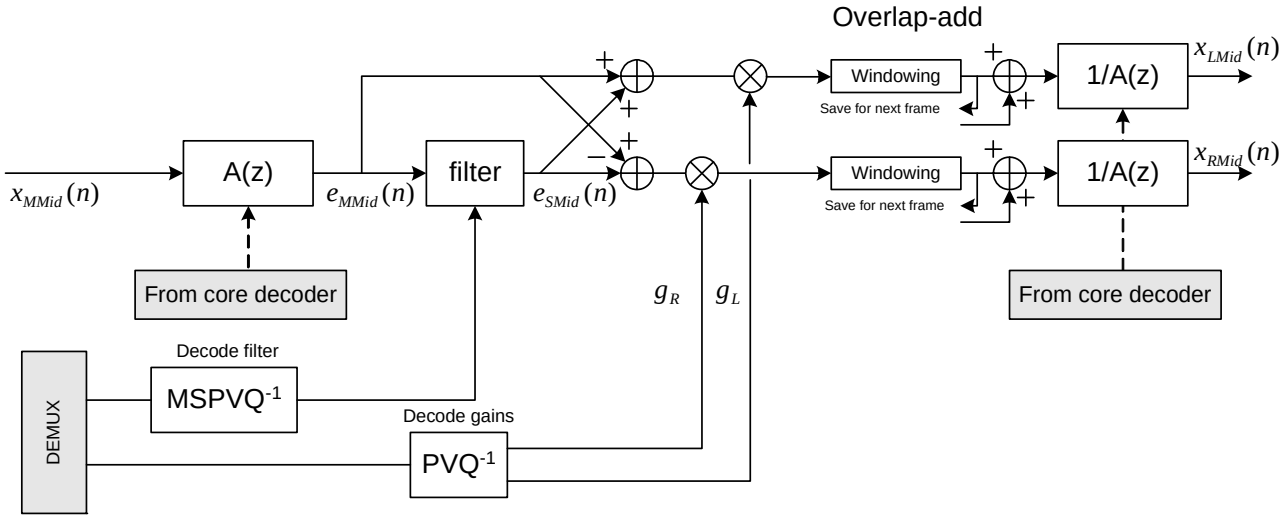
The AVQ parameters are decoded in a similar way as described in section 6.1.2, after decoding the step of spectrum de-shaping is performed in order to de-emphasize the spectral coefficients.

The global gain decoding is performed according to Section 6.1. by inverting the 7-bit logarithmic quantization calculated in the encoder as in Section 5.5.1. The signal after inverse transform is multiplied by the global gain in order to produce a time domain signal $y(n)$.

Decoding the balance factor is done by inverting the 7 bit scalar quantizer. In the case of mode 0, a signal envelope is estimated from the mono signal $x_{MLo}(n)$ and this signal is compensated for the envelope. The resulting signal is multiplied by the balance factor and added to the time domain signal $y(n)$. The estimated time domain envelope is used in order to shape the resulting signal and reduce the pre-echo artefact when mode 0 is selected in the encoder.

Adaptive overlap-add is used in order to synthesise the time domain side signal $x_{SLo}(n)$ which is used to produce the low frequencies left and right signal $x_{RLo}(n)$, $x_{LLo}(n)$.

6.3.2 Stereo Signal Mid-Band synthesis



The mono mid-band signal is filtered in order to obtain the mid-band excitation signal $e_{MMid}(n)$. The filter and the corresponding gains are decoded by using the inverse operation of the predictive multistage quantizer (MSPVQ^{-1}) and the predictive 2 dimensional vector quantizer (PVQ^{-1}).

The filter is used to filter the excitation signal $e_{MMid}(n)$ according to

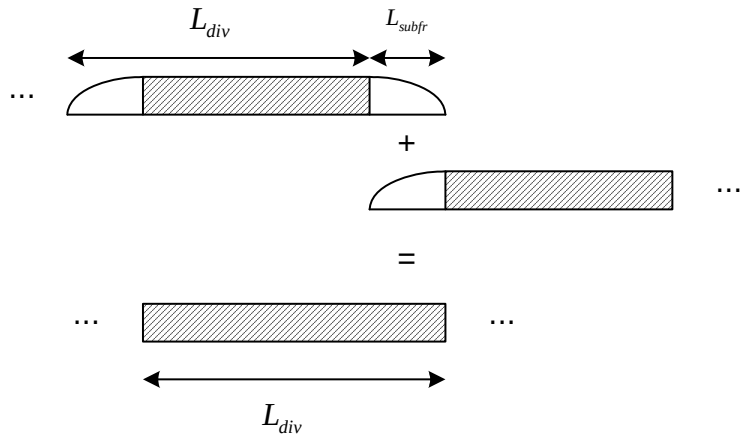
$$e_{SMid}(n) = \sum_{i=0}^{8} b_{iq} e_{MMid}(n-i), \qquad 0 \le n < L_{div} + L_{subfr}$$

the two channels pseudo-excitations are computed as:

$$e_{LMid}(n) = g_L e_{MMid}(n) - g_L \sum_{i=0}^{8} b_{iq} e_{MMid}(n-i), \qquad 0 \le n < L_{div} + L_{subfr}$$

$$e_{RMid}(n) = g_R e_{MMid}(n) + g_R \sum_{i=0}^{8} b_{iq} e_{MMid}(n-i), \qquad 0 \le n < L_{div} + L_{subfr}$$

Windowing and overlap add is used in norder to smooth the frame transitions. The anount of overlap is equal to L_{subfr} according to the following figure



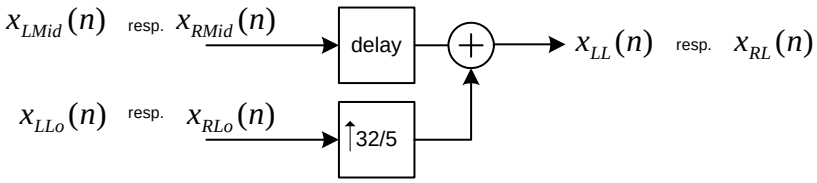
The output of the overlap add sections are finally inverse filtered by the core low band signal lpc filter $1/A(z)$ to produce the left and right mid-band signals $x_{LMid}(n)$ and $x_{RMid}(n)$.

6.3.3 Stereo Signal High-Band synthesis

Same as mono signal high band synthesis, the high band for each signal is delayed slightly to be aligned with the low and midband stereo signal

6.3.4 Stereo output signal generation

The low and high frequencies of the low band left and right channels are combined together according to the following figure



The obtained low band left and right channels are combined with the high band signals in order to restore the full band left and right channels.

6.4 Stereo to mono conversion

In the case the application or hardware does not support stereo output, the output of the synthesised output stereo bit stream needs to be mixed to mono.

6.4.1 Low-Band synthesis

The low band stereo output is converted to mono by omitting the stereo information in the bit stream and skipping the stereo decoder. Hence, the mono output is synthesised only according to Section 6.1.

6.4.2 High-Band synthesis

The stereo bit stream contains separate bandwidth extensions for left and right High-Band synthesis. To minimise the decoder complexity, the stereo to mono down mixing is done on parameter level.

First the High-Band gain factors for both left and right channel are decoded as explained in Section 6.2. The gain difference of left and right channels is calculated in dB scale. When the difference is greater than 20 dB, the LP synthesis filter coefficients of the higher energy channel are selected for the mono output synthesis filtering and the LP parameters from the lower energy channel are omitted completely. On the other hand, when the gain difference is less than the 20 dB threshold, the LP synthesis filter coefficients of left and right channel are averaged in ISP domain. The gain parameters from left and right channels are averaged in both cases. When the LP synthesis filter coefficients and gain factor is determined, the High-Band synthesis is done according to mono signal High-Band synthesis in Section 6.2.

6.5 Bad frame concealment

6.5.1 Mono

6.5.1.1 Mode decoding and extrapolation

In the presence of packet losses, the decoder tries to recover the missing mode indicators from the available ones (including also mode indicators of previous superframes). Recall that the mode selected in a given super-frame is given by **MODE** = (m_0, m_1, m_2, m_3) where $0 \le m_k \le 3$ and k=0,..,3. The 26 valid modes are enumerated in Table10. When frame k is missing at the receiver, bfi_k is set to 1. When $bfi_k = 1$, the value m_k is not available and has to be estimated from other received information.

The mode extrapolation is essentially based on a mode repetition logic. The mode indicators from the previous super-frame only are reused in the extrapolation. More precisely, only the last indicator of the previous mode is used. Hence, the modes of the four 256-sample frames in the previous superframe are seen as (X, X, X, m_{-1}) where the value X is not relevant (this value is not used here) and $0 \le m_{-1} \le 3$ is the final indicator of the previous mode. Note that if m_{-1} was not available, the extrapolated value of m_{-1} is used.

A high-level block diagram of the mode extrapolation module is given in Figure 17.

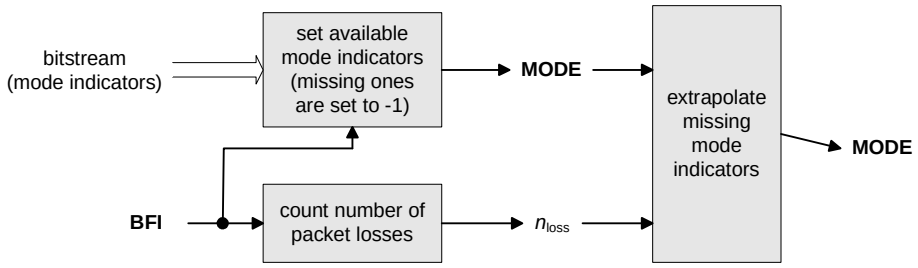


Figure 17: High-level block diagram of mode extrapolation module

Based on the values in **BFI**, the available mode indicators are set from the bits coming from the demultiplexer. The number of packet losses n_{loss} is counted as the number of bfi_k values set to 1. The mode is given by **MODE =** (m_0, m_1, m_2, m_3) with $0 \le m_k \le 3$ when the indicator m_k is available (i.e. $bfi_k = 0$), and $m_k = -1$ when $bfi_k = 1$. Then, the missing mode indicators (for which $m_k = -1$) are extrapolated. The logic of this mode extrapolation is shown in Figure 18.

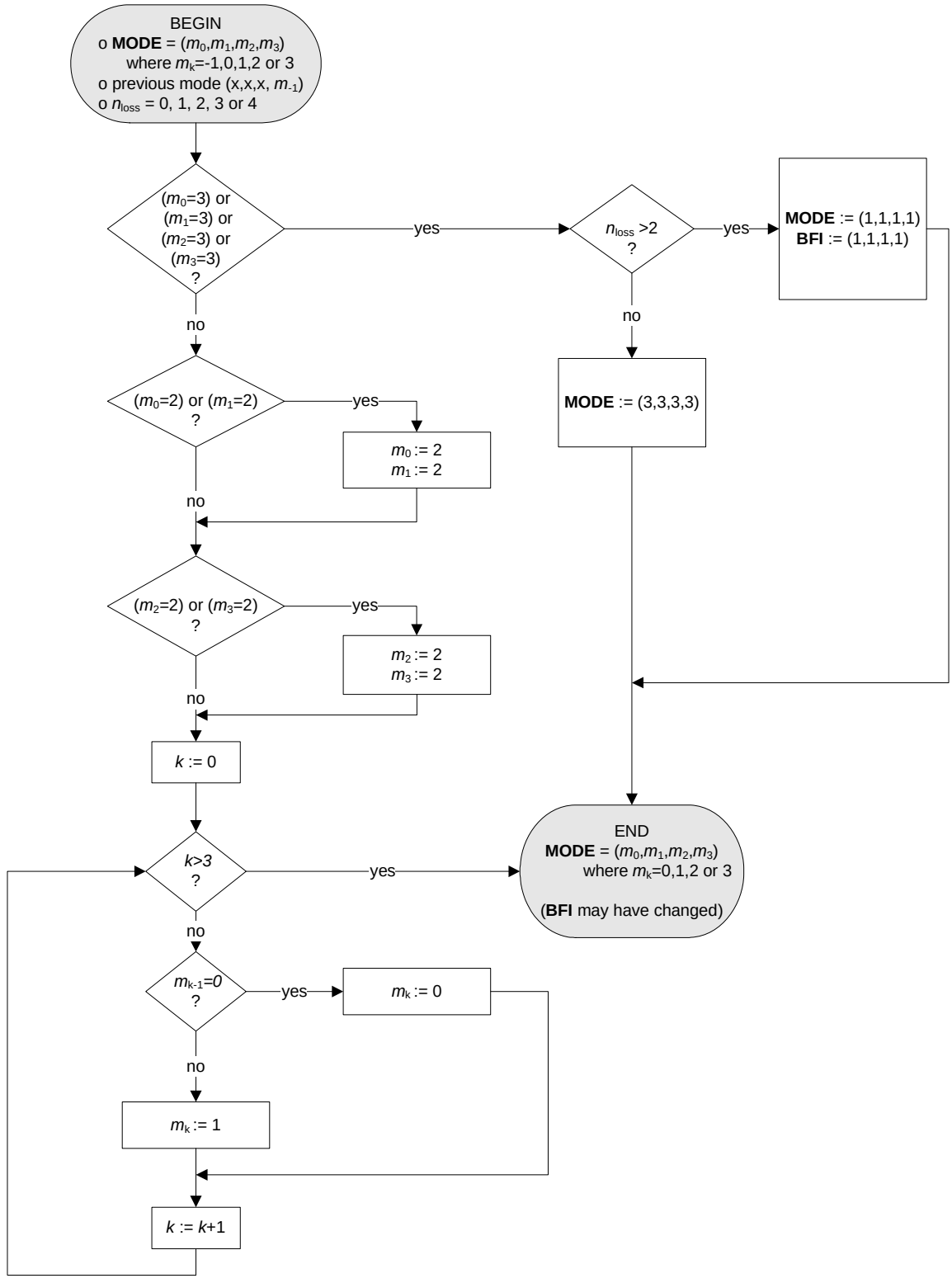


Figure 18: Mode extrapolation logic

The rationale behind the mode extrapolation logic in Figure 18 is as follows:

- There exists redundancy in the definition of mode indicators. A TCX-1024 frame is described by **MODE** = (3,3,3,3), and a TCX-512 frame is described by (2,2,X,X) or (X,X,2,2). Therefore, *in the absence of bit errors*, the mode indicators describing a TCX-512 or TCX-1024 frame can be easily extrapolated in case of partial packet losses, when a single value $m_k = 2$ or 3 is available.

- The frame-erasure concealment in ACELP mode relies on the pitch delay and codebook gains of the previous ACELP frame. However in switched ACELP/TCX coding there is no guarantee that the frame preceding an

ACELP frame was also encoded by ACELP. Assuming that m_k is not available and that the extrapolation has to choose between $m_k = 0$ or $m_k = 1$, the extrapolation will select ACELP decoding ($m_k = 0$) only if $m_{k-1} = 0$ Otherwise the ACELP parameters needed for concealment would not be up-to-date. As a consequence, under the above assumptions, if $m_{k-1}>0$, the value $m_k = 1$ will be selected.

- If 3 packets are lost and if the only available mode indicator is $m_k = 3$ with k =0,1,2 or 3, a mode (3,3,3,3) corresponding to TCX-1024 should normally be extrapolated. Yet, with the bitstream format described in Section 5.6, losing 3 packets out of 4 in TCX-1024 means

  1) losing roughly 3 quarters of the TCX target spectrum and

  2) having no information about the TCX global gain since the gain repetition in TCX-1024 is designed to perform well for up to 2 packet losses.

  As a consequence, the mode (3,3,3,3) is rather replaced by the mode (1,1,1,1) in the extrapolation when more than 2 packets are lost. Note that this causes the concealment of TCX-256 to be used (the synthesis will actually be progressively faded out).

6.5.1.2 TCX bad frame concealment

Concealment of TCX256 erased frames was described in Section 6.2.1.

In the case of a TCX1024 partial frame loss and given that the previous decoded frame was also a TCX1024 frame a spectral fill-in strategy is used in order to conceal the lost packets. The fill-in strategy assumes that since we have a case of two consecutive 1024-sample TCX frames, the signal is quasi stationary so that lost subvectors can be interpolated from the previous frame.

6.5.1.2.1 Spectrum de-shaping

Spectrum de-shaping is applied to the quantized spectrum as described in Section 5.3.5.8. In case of frame erasure, the de-shaping uses a prediction of the new maximum using the previously saved quantized spectrum. De-shaping is done according to the following steps:

- Compute the maximum energy $OldE_{max}$ of the 8-dimensional block at position index m of the previous 1024-sample TCX frame
- Compute the maximum energy E_{max} of the 8-dimensional block at position index m of the current 1024-sample TCX frame
- If $E_{max} < OldE_{max}$, then set $E_{max} = OldE_{max}$
- Calculate the energy E_m of the 8-dimensional block at position index m
- Compute the ratio $R_m = E_{max} / E_m$
- Compute the value $(R_m)^{1/2}$
- if $R_m > 10$, then set $R_m = 10$ (maximum gain of 20 dB)
- also, if $R_m > R_{m-1}$ then $R_m = R_{m-1}$

This allows in case of the loss of the 8-dimensional block corresponding to the maximum to use the previous maximum.

6.5.1.2.2 Spectrum Extrapolation

Spectrum extrapolation is applied to the quantized spectrum prior to applying the inverse FFT. Spectrum extrapolation consists of amplitude and phase extrapolation applied to the lost spectral coefficients. The extrapolated amplitude and phase are combined to form the extrapolated spectral coefficient. Combining the extrapolated and the received spectral coefficients is done in order to form quantized spectrum $\widehat{X}[k]$.

6.5.1.2.3 Amplitude Extrapolation

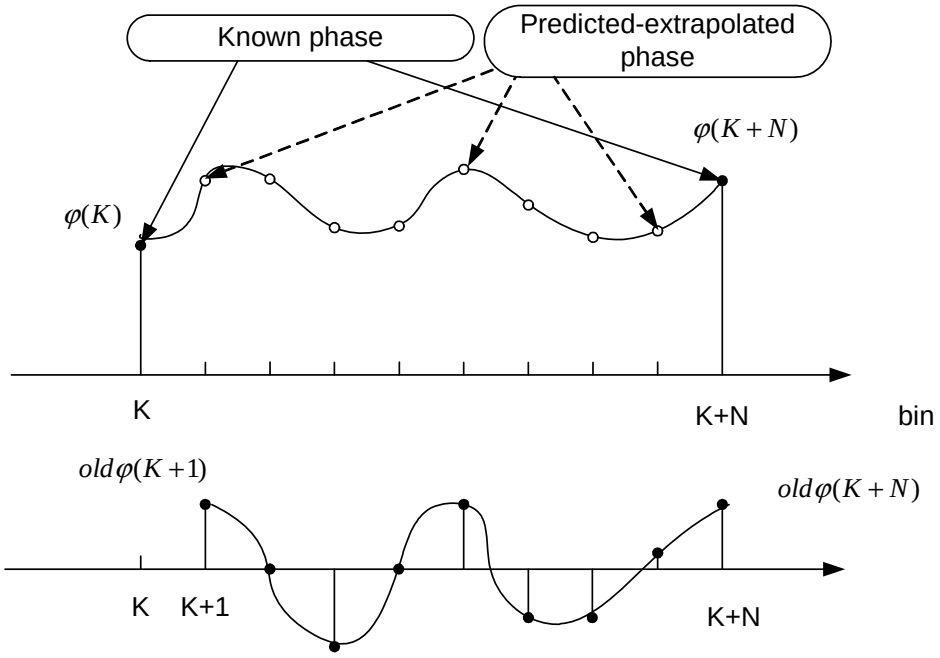
Spectral amplitude extrapolation consists of is performed according to the following steps;

- Compute the previous frame amplitude spectrum, $oldA[k] = \left| old\hat{X}[k] \right|$
- Compute the current frame spectrum, $A[k] = \left| \hat{X}[k] \right|$
- Compute the gain difference of energy of non-lost spectral coefficients between the previous and the current frame

$$gain = \sqrt{\frac{\sum_{k,S[k]\neq 0} A[k]^2}{\sum_{k,S[k]\neq 0} oldS[k]^2}}$$

- Extrapolate the amplitude of the missing spectral coefficients using
if (*lost[k]*) $A[k] = gain \cdot oldA[k]$

6.5.1.2.4 Phase Extrapolation



Phase extrapolation uses the principle of group delay conservation for quasi-stationary signals. First the group delay is estimated on the previous frame and then used in the current frame in order to extrapolate the phase on the missing spectral coefficients. The estimation of the group delay is done by computing

$$\Delta old\varphi(k) = old\varphi(k) - old\varphi(k-1)$$

The phase of the missing spectral coefficients $\hat{\varphi}(k)$ is computed by using the following recursive algorithm

$$\hat{\varphi}(k) = \hat{\varphi}(k-1) + \Delta old\varphi(k) + \Delta\varphi_c, \quad k = K+1,...,K+N-1$$

where

$$\Delta\varphi_c = (1/N) \cdot (\varphi(K+N) - old\varphi(K+N) - \varphi(K) + old\varphi(K))$$

$\hat{\varphi}(K) = \varphi(K)$ is used to start the recursion., and $K,.K+N$ are received (non lost) bins.

6.5.2 Stereo

The stereo error concealment is controlled by the bad frame indicators bfi_k, k = 0…3. In response to the bad frame indicators of the present super frame together with some bad frame indicator history proper actions are taken mitigating the perceptual impact of bad frames. Particular error mitigation actions are taken on the stereo low-band, the mid-band, and the high-band.

6.5.2.1 Low-band

Balance factor

The balance factor to be used for the derivation of the side signal is not available depending on bfi_k and on the stereo TCX frame length, i.e. if

bfi_k = 1, k = 0…3 in case of 256-sample stereo TCX frames,

bfi_k = 1, k = 0, 2 in case of 512-sample stereo TCX frames, or

bfi_k = 1, k = 0 in case of 1024-sample stereo TCX frames,

In this case, the balance factor is derived from the balance factor of the previous stereo TCX frame, however, attenuated by 0.9.

For the case of a future frame loss, the balance factor of the present stereo TCX frame is stored in a history buffer.

Side signal error signal

The side signal error signal is derived using the TCX decoder and the associated bad frame concealment described above (6.6.1.3). Input to the TCX bad frame concealment is a flag, signalling if any of the frames associated with the present stereo TCX frame is bad.

Side signal

Stereo TCX frames of size 512 samples and 1024 samples are reconstructed as in the case without bad frames, however using the balance factor and side signal error signal derived as specified above.

Bad stereo TCX frames of size 256 samples are, however, reconstructed differently. A parametric model with transfer function

$$H(z) = \sum_{i=0}^{P} h(i) \cdot z^{-i} \text{ , } P=8$$

is applied to the windowed mono signal for reconstructing a substitution signal for the side signal. The filter coefficients are taken from a state memory and are always derived during preceding stereo TCX frames if the associated bfi_k flags are equal to zero. The coefficients are calculated by solving the following equation system:

$$\underline{\underline{R}}_{mm} \cdot \underline{h} = \underline{r}_{ca},$$

where $\underline{\underline{R}}_{mm}$ is a Toeplitz matrix of autocorrelations r_{mm} of the windowed mono signal:

$$\underline{\underline{R}}_{mm} = \left[r_{mm}(j-k)\right], \quad j,k \in \left[0...P\right],$$

and where $\underline{r}_{ms}$ is a vector of cross-correlations r_{ms} of the windowed mono signal and the side signal*:*

$$\underline{r}_{ms} = \left[r_{ms}(k)\right], \quad k \in \left[0...P\right].$$

Left/right signal reconstruction

The side signal used for left/right signal reconstruction is attenuated in case of severe frame loss conditions of an estimated frame loss rate of greater than 1%. Using an estimate $\bar{f}$ of the present average frame loss rate, an attenuation factor α is derived according to the following formula:

$$\alpha = 1 - \min(0.7, 17.5 \cdot (\bar{f} - 0.01)),$$

where, in addition, α is limited to be within the range of 0…1.

Before reconstructing left and right signals, the side signal is multiplied with factor α.

The average frame loss rate is estimated according to the following algorithm.

A first estimate f is calculated according to

$$f = \frac{1}{375} \cdot \sum_{j=0}^{N-1} w_j \cdot bfi\_buf(-j), \; N=500,$$

where w_j is a weighting factor defined as

$$w_j = \min(1, 1.5 - \frac{j}{N}),$$

and *bfi_buf* is a buffer comprising the N most recent flags bfi_k.

The final frame loss rate estimate is then obtained by AR-1 filtering:

$$\bar{f} = 0.9 \cdot \bar{f}' + 0.1 \cdot f,$$

where $\bar{f}'$ is the frame loss estimate calculated during processing of the preceding frame.

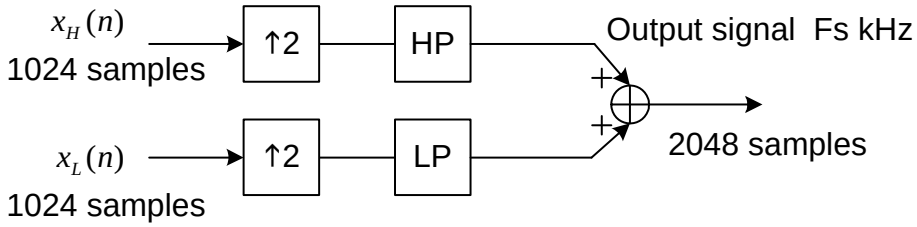
6.5.2.2 Mid-band

The mid-band synthesis is performed on every frame. When a frame is lost the parameters of the mid-band decoder are extrapolated by using the predictive decoders fed with a zero error signal. This implies that the error will propagate to few frames which does not impact the overall quality. The extrapolated parameters are the filter coefficients and the channel gains and are computed as:

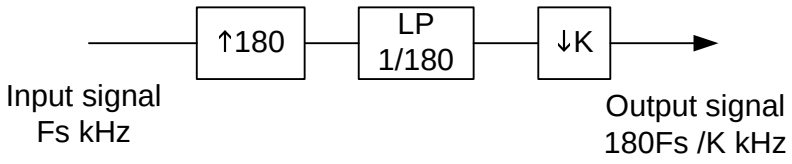
$$g_L(frame) = 0.5 \cdot g_L(frame - 1),$$
$$g_R(frame) = 0.5 \cdot g_R(frame - 1),$$
$$b_i(frame) = 0.5 \cdot b_i(frame - 1), i = 0, \cdots, 8$$

6.6 Output signal generation

The decoder output signal(s) are generated by combining the low and high band signal to produce full band signals. This operation is the inverse of the encoder band-splitting operation described in section 5.1. The following figure shows how the operation is performed.



The exact same filters used in the encoder are re-used in the decoder. Furthermore, if the desired output sampling rate is different from the internal sampling rate, then a resampling operation is preformed which is the reverse operation of that performed in the encoder.



7 Detailed bit allocation of the Extended AMR-WB codec

The detailed allocation of the bits in the AMR-WB+ audio encoder is shown for each frame type in tables 14-17 for mono and tables 18-20 for stereo. These tables show the order of the bits produced by the audio encoder. Note that the most significant bit (MSB) of each codec parameter is always sent first. For TCX512 frames, the frame is split in two equal packets. For TCX1024 frames, the frame is split in four equal packets. The splitting of TCX512 and TCX1024 frames in several packets is explained in Section 5.6.1.

Table 14: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of ACELP coding type

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
Mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
1st ISP subvec	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9
2nd ISP subvec	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17
3rd ISP subvec	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23
4th ISP subvecr	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30
5th ISP subvec	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37
6th ISP subvec	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42
7th ISP subvec	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47
index of mean energy	b48 – b49	b48 – b49	b48 – b49	b48 – b49	b48 – b49	b48 – b49	b48 – b49	b48 – b49
subframe 1								
Adaptive CB index	b50 – b58	b50 – b58	b50 – b58	b50 – b58	b50 – b58	b50 – b58	b50 – b58	b50 – b58
LTP-filtering-flag	b59	b59	b59	b59	b59	b59	b59	b59
Algebraic CB indices	b60 – b147	b60 – b131	b60 – b123	b60 – b111	b60 – b103	b60 – b95	b60 – b87	b60 – b79
codebook gains	b148 – b154	b132 – b138	b124 – b130	b112 – b118	b104 – b110	b96 – b102	b88 – b94	b80 – b86
subframe 2								
Adaptive CB index	b155 – b160	b139 – b144	b131 – b136	b119 – b124	b111 – b116	b103 – b108	b95 – b100	b87 – b92
LTP-filtering-flag	b161	b145	b137	b125	b117	b109	b101	b93
Algebraic CB indices	b162 – b249	b146 – b217	b138 – b201	b126 – b177	b118 – b161	b110 – b145	b102 – b129	b94 – b113
codebook gains	b250 – b256	b218 – b224	b202 – b208	b178 – b184	b162 – b168	b146 – b152	b130 – b136	b114 – b120
subframe 3								
Adaptive CB index	b257 – b265	b225– b233	b209 – b217	b185 – b193	b169 – b177	b153 – b161	b137 – b145	b121 – b129
LTP-filtering-flag	b266	b234	b218	b194	b178	b162	b146	b130
Algebraic CB indices	b267 – b354	b235 – b306	b219 – b282	b195 – b246	b179 – b222	b163 – b198	b147 – b174	b131 – b150
codebook gains	b355 – b361	b307 – b313	b283 – b289	b247 – b253	b223 – b229	b199 – b205	b175 – b181	b151 – b157
subframe 4								
Adaptive CB index	b362 – b367	b314 – b319	b290 – b295	b254 – b259	b230 – b235	b206 – b211	b182 – b187	b158 – b163
LTP-filtering-flag	b368	b320	b296	b260	b236	b212	b188	b164
Algebraic CB indices	b369 – b456	b321 – b392	b297 – b360	b261 – b312	b237 – b280	b213 – b248	b189 – b216	b165 – b184
codebook gains	b457 – b463	b493 – b399	b361 – b367	b313 – b319	b281 – b287	b249 – b255	b217 – b223	b185 – b191
Bandwidth extension								
Index of HF ISP	b464 – b472	b400 – b408	b368 – b376	b320 – b328	b288 – b296	b256 – b264	b224 – b232	b192 – b200
Index of HF gain	b473 – b479	b409 – b415	b377 – b383	b329– b335	b297 – b303	b265 – b271	b233 – b239	b201 – b207

Table 15: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX256 frame type

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
1st ISP subvec	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9
2nd ISP subvec	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17
3rd ISP subvec	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23	b18 - b23
4th ISP subvecr	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30	b24 – b30
5th ISP subvec	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37	b31 – b37
6th ISP subvec	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42	b38 – b42
7th ISP subvec	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47	b43 – b47
Noise factor	b48-b50	b48-b50	b48-b50	b48-b50	b48-b50	b48-b50	b48-b50	b48-b50
Global gain	b51 – b57	b51 – b57	b51 – b57	b51 – b57	b51 – b57	b51 – b57	b51 – b57	b51 – b57
Algebraic VQ	b58 – b463	b58 – b399	b58 – b367	b58 – b319	b58 – b287	b58 – b255	b58 – b223	b58 – b191
Bandwidth extension								
Index of HF ISP	b464 – b472	b400 – b408	b368 – b376	b320 – b328	b288 – b296	b256 – b264	b224 – b232	b192 – b200
Index of HF gain	b473 – b479	b409 – b415	b377 – b383	b329– b335	b297 – b303	b265 – b271	b233 – b239	b201 – b207

Table 16a: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX512 frame type – First Packet

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
1st ISP subvec	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9
2nd ISP subvec	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17
Noise factor	b18-b20	b18-b20	b18-b20	b18-b20	b18-b20	b18-b20	b18-b20	b18-b20
Global gain	b21 – b27	b21 – b27	b21 – b27	b21 – b27	b21 – b27	b21 – b27	b21 – b27	b21 – b27
Split Algebraic VQ	b28 – b463	b28 – b399	b28 – b367	b28 – b319	b28 – b287	b28 – b255	b28 – b223	b28 – b191
Bandwidth extension								
Index of HF ISP	b464 – b472	b400 – b408	b368 – b376	b320 – b328	b288 – b296	b256 – b264	b224 – b232	b192 – b200
Index of HF gain	b473– b479	b409 – b415	b377 – b383	b329– b335	b297 – b303	b265 – b271	b233 – b239	b201 – b207

Table 16b: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX512 frame type – Second Packet

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
3rd ISP subvec	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7
4th ISP subvecr	b8 – b14	b8 – b14	b8 – b14	b8 – b14	b8 – b14	b8 – b14	b8 – b14	b8 – b14
5th ISP subvec	b15 – b21	b15 – b21	b15 – b21	b15 – b21	b15 – b21	b15 – b21	b15 – b21	b15 – b21
6th ISP subvec	b22 – b26	b22 – b26	b22 – b26	b22 – b26	b22 – b26	b22 – b26	b22 – b26	b22 – b26
7th ISP subvec	b27 – b31	b27 – b31	b27 – b31	b27 – b31	b27 – b31	b27 – b31	b27 – b31	b27 – b31
Gain redundancy (6 MSBs)	b32-b37	b32-b37	b32-b37	b32-b37	b32-b37	b32-b37	b32-b37	b32-b37
Split Algebraic VQ	b38 – b463	b38 – b399	b38 – b367	b38 – b319	b38 – b287	b38 – b255	b38 – b223	b38 – b191
Bandwidth extension								
Gain correction 8x2 bits	b464-b479	b400-b415	b368-b383	b320-b335	b288-b303	b256-b271	b224-b239	b192-b207

Table 17a: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX1024 frame type – First Packet

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
1st ISP subvec	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9	b2 – b9
2nd ISP subvec	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17	b10 – b17
Global gain	b18 – b24	b18 – b24	b18 – b24	b18 – b24	b18 – b24	b18 – b24	b18 – b24	b18 – b24
Split Algebraic VQ	b25 – b463	b25 – b399	b25 – b367	b25 – b319	b25 – b287	b25 – b255	b25 – b223	b25 – b191
Bandwidth extension								
Index of HF ISP	b464 – b472	b400 – b408	b368 – b376	b320 – b328	b288 – b296	b256 – b264	b224 – b232	b192 – b200
Index of HF gain	b473 – b479	b409 – b415	b377 – b383	b329– b335	b297 – b303	b265 – b271	b233 – b239	b201 – b207

Table 17b: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX1024 frame type – Second packet

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
3rd ISP subvec	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7	b2 – b7
Noise factor	b8-b10	b8-b10	b8-b10	b8-b10	b8-b10	b8-b10	b8-b10	b8-b10
Global gain parity	b11 – b13	b11 – b13	b11 – b13	b11 – b13	b11 – b13	b11 – b13	b11 – b13	b11 – b13
Split Algebraic VQ	b14– b463	b14 – b399	b14 – b367	b14 – b319	b14 – b287	b14 – b255	b14 – b223	b14 – b191
Bandwidth extension								
Gain correction 8x2 bits (MSBs 1st 8 subframes)	b464-b479	b400-b415	b368-b383	b320-b335	b288-b303	b256-b271	b224-b239	b192-b207

Table 17c: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX1024 frame type – Third packet

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
4th ISP subvecr	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8
6th ISP subvec	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13
Global gain redundancy	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16
Split Algebraic VQ	b17– b463	b17– b399	b17 – b367	b17 – b319	b17 – b287	b17 – b255	b17 – b223	b17 – b191
Bandwidth extension								
Gain correction 8x2 bits (MSBs 2nd 8 subframes)	b464-b479	b400-b415	b368-b383	b320-b335	b288-b303	b256-b271	b224-b239	b192-b207

Table 17d: Source encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX1024 frame type – Fourth packet

	Bits (MSB-LSB)							
Description	480 bits/frame	416 bits/frame	384 bits/frame	336 bits/frame	304 bits/frame	272 bits/frame	240 bits/frame	208 bits/frame
mode bits	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1	b0-b1
5th ISP subvec	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8	b2 – b8
7th ISP subvec	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13	b9 – b13
Global gain redundancy	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16	b14 – b16
Split Algebraic VQ	b17– b463	b17– b399	b17 – b367	b17 – b319	b17 – b287	b17 – b255	b17 – b223	b17 – b191
Bandwidth extension								
Gain correction 16x1 bits (LSB 16 subframes)	b464-b479	b400-b415	b368-b383	b320-b335	b288-b303	b256-b271	b224-b239	b192-b207

N_1 is the number of bits per frame allocated for the low and midband and is calculated according to

Table 18: Stereo encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX48 frame type, mode 0 and 1

	Bits (MSB-LSB)	
Description	N bits/frame <=76	N bits/frame > 76
Midband stereo		
Midband filter	b0-b3	b0-b6
Midband gain	b4-b5	b7-b11
Lowband stereo		
Mode bits	b6-b7	b12-b13
reserved	b8	b14
Balance factor	B9-b15	b15-b21
Global gain	b16 – b22	b22-b28
Algebraic VQ	b23 – bN_1	b29- bN_1

Table 19a: Stereo encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX96 frame type, mode 2 - First packet

	Bits (MSB-LSB)	
Description	N bits/frame <=76	N bits/frame > 76
Midband stereo		
Midband filter	b0-b3	b0-b6
Midband gain	b4-b5	b7-b11
Lowband stereo		
Mode bits	b6-b7	b12-b13
reserved	b8	b14
Balance factor	B9-b15	b15-b21
Algebraic VQ	b16 – bN_1	b22- bN_1

Table 19b: Stereo encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX96 frame type, mode 2 - Second packet

	Bits (MSB-LSB)	
Description	N bits/frame <=76	N bits/frame > 76
Midband stereo		
Midband filter	b0-b3	b0-b6
Midband gain	b4-b5	b7-b11
Lowband stereo		
Mode bits	b6-b7	b12-b13
reserved	b8	b14
Global gain	b9-b15	b15-b21
Algebraic VQ	b16– bN_1	b22- bN_1

Table 20a: Stereo encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX96 frame type, mode 3 - First packet

	Bits (MSB-LSB)	
Description	N bits/frame <=76	N bits/frame > 76
Midband stereo		
Midband filter	b0-b3	b0-b6
Midband gain	b4-b5	b7-b11
Lowband stereo		
Mode bits	b6-b7	b12-b13
reserved	b8	b14
Balance factor	b9-b15	b15-b21
Algebraic VQ	b16– bN_1	b22- bN_1

Table 20b: Stereo encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX96 frame type, mode 3 - Second packet

	Bits (MSB-LSB)	
Description	N bits/frame <=76	N bits/frame > 76
Midband stereo		
Midband filter	b0-b3	b0-b6
Midband gain	b4-b5	b7-b11
Lowband stereo		
Mode bits	b6-b7	b12-b13
reserved	b8	b14
Algebraic VQ	b9 – bN_1	b15- bN_1

Table 20c: Stereo encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX96 frame type, mode 3 - Third packet

	Bits (MSB-LSB)	
Description	N bits/frame <=76	N bits/frame > 76
Midband stereo		
Midband filter	b0-b3	b0-b6
Midband gain	b4-b5	b7-b11
Lowband stereo		
Mode bits	b6-b7	b12-b13
reserved	b8	b14
Global gain	B9-b15	b15-b21
Algebraic VQ	b16 – bN_1	b22- bN_1

Table 20d: Stereo encoder output parameters in order of occurrence and bit allocation within the audio frame of TCX96 frame type, mode 3 - Fourth packet

	Bits (MSB-LSB)	
Description	N bits/frame <=76	N bits/frame > 76
Midband stereo		
Midband filter	b0-b3	b0-b6
Midband gain	b4-b5	b7-b11
Lowband stereo		
Mode bits	b6-b7	b12-b13
reserved	b8	b14
Algebraic VQ	b9 – bN_1	b15- bN_1

8 Storage and Transport Interface formats

The AMR-WB+ codec storage and transport interface formats are described in this section.

8.1 Available Modes and Bitrates

The AMR-WB+ format contains the AMR-WB modes and a set of AMR-WB+ extension modes.
The AMR-WB+ codec includes the AMR-WB modes, as shown in Table 21 below.

Table 21: AMR-WB+ modes.

Index	*Mode*	*Sampling rate (kHz)*	*Mono/Stereo*	*Number of data octets per frame (AMR-WB modes zero-padded)*
0	AMR-WB 6.60 kbit/s	16	Mono	17
1	AMR-WB 8.85 kbit/s	16	Mono	23
2	AMR-WB 12.65 kbit/s	16	Mono	32
3	AMR-WB 14.25 kbit/s	16	Mono	36
4	AMR-WB 15.85 kbit/s	16	Mono	40
5	AMR-WB 18.25 kbit/s	16	Mono	46
6	AMR-WB 19.85 kbit/s	16	Mono	50
7	AMR-WB 23.05 kbit/s	16	Mono	58
8	AMR-WB 23.85 kbit/s	16	Mono	60
9	AMR-WB SID	16	Mono	5
10	AMR-WB+ 13.6 kbit/s	16/24	Mono	34
11	AMR-WB+ 18 kbit/s	16/24	Stereo	45
12	AMR-WB+ 24 kbit/s	16/24	Mono	60
13	AMR-WB+ 24 kbit/s	16/24	Stereo	60
14	FRAME_ERASURE	-	-	0
15	NO_DATA	-	-	0

There are four special extension modes (Index 10-13 in table 21) that have a fixed internal sampling frequency (25600 Hz) and audio input frequencies (16 or 24 kHz). These modes share the property with the AMR-WB modes that each frame is only capable of representing 20 ms.

Besides the AMR-WB+ operation according to the modes specified in table 21, AMR-WB+ operation is specified by three parameters: mono bit-rate as given in Table 22, stereo bit-rate as given in Table 23, and internal sampling frequency (ISF) as given in Table 24.

Table 22: Mono rate indices.

Mono Index	Mono rate	Bit rate at 25.6 kHz ISF	Octets per frame
0	AMR-WB+ 208 bit/frame	10.4 kbit/s	26
1	AMR-WB+ 240 bit/frame	12 kbit/s	30
2	AMR-WB+ 272 bit/frame	13.6 kbit/s	34
3	AMR-WB+ 304 bit/frame	15.2 kbit/s	38
4	AMR-WB+ 336 bit/frame	16.8 kbit/s	42
5	AMR-WB+ 384 bit/frame	19.2 kbit/s	48
6	AMR-WB+ 416 bit/frame	20.8 kbit/s	52
7	AMR-WB+ 480 bit/frame	24 kbit/s	60

Table 23: Stereo rate indices.

Stereo index	*Stereo extension rate*(bits/frame)	*Stereo rate for ISF of 25.6 kHz*	*Number of data octets per frame*
0	40 bits/frame	2.0 kbit/s	5
1	48 bits/frame	2.4 kbit/s	6
2	56 bits/frame	2.8 kbit/s	7
3	64 bits/frame	3.2 kbit/s	8
4	72 bits/frame	3.6 kbit/s	9
5	80 bits/frame	4.0 kbit/s	10
6	88 bits/frame	4.4 kbit/s	11
7	96 bits/frame	4.8 kbit/s	12
8	104 bits/frame	5.2 kbit/s	13
9	112 bits/frame	5.6 kbit/s	14
10	120 bits/frame	6.0 kbit/s	15
11	128 bits/frame	6.4 kbit/s	16
12	136 bits/frame	6.8 kbit/s	17
13	144 bits/frame	7.2 kbit/s	18
14	152 bits/frame	7.6 kbit/s	19
15	160 bits/frame	8.0 kbit/s	20

It is to be noted that the number of samples each frame corresponds to is always the same but the duration of each frame varies depending on the internal sampling frequency. There is no preferred sampling frequency for the codec to operate at, but in order to limit the possible settings for an effective transmission, the format supports the sampling frequencies given in Table 24. . Herein, index 0 is used for AMR-WB and the 4 extension modes of Table 21.

Table 24: Internal sampling frequencies and corresponding frame lengths in time

ISF Index	*Internal Sampling Rate (Hz)*	*Frame duration (ms)*	*Bit Rate factor*
0	N/A	20	N/A
1	12800	40	½
2	14400	35.55	9/16
3	16000	32	5/8
4	17067	30	2/3
5	19200	26.67	¾
6	21333	24	5/6
7	24000	21.33	15/16
8	25600	20	1
9	28800	17.78	9/8
10	32000	16	5/4
11	34133	15	4/3
12	36000	14.22	45/32
13	38400	13.33	3/2

The bit-rate will be dependent on the internal sampling frequency. The last column of Table 24 indicates which multiplication factor, any bit-rate value for 25600 Hz internal sampling factor should be converted with. The ISF index is carried in the bitstream format to indicate which internal sampling frequency is used for each AMR-WB+ encoded frame.

The frame type is used to identify the content of an AMR-WB+ encoded frame. This type indicates if it is; an AMR-WB mode, Comfort noise, NO_DATA, AMR-WB+ core mode in mono usage, or a combination of a core mode and a stereo mode. The frame types are presented in Table 25 below. The core mode and stereo mode index values are according to Table 22 and 23 respectively. The bit-rate value assumes an internal sampling frequency of 25600 Hz.

Table 25: Normative frame type table. Bit-rates assumes 25600 Hz internal sampling frequency.

Frame type	Core mode	Stereo mode	Bit rate	Octets per frame
0-15	As specified in Table 21			
16	0	None	10.4	26
17	1	None	12.0	30
18	2	None	13.6	34
19	3	None	15.2	38
20	4	None	16.8	42
21	5	None	19.2	48
22	6	None	20.8	52
23	7	None	24.0	60
24	0	0	12.4	31
25	0	1	12.8	32
26	0	4	14	35
27	1	1	14.4	36
28	1	3	15.2	38
29	1	5	16	40
30	2	2	16.4	41
31	2	4	17.2	43
32	2	6	18	45
33	3	3	18.4	46
34	3	5	19.2	48
35	3	7	20	50
36	4	4	20.4	51
37	4	6	21.2	53
38	4	9	22.4	56
39	5	5	23.2	58
40	5	7	24	60
41	5	11	25.6	64
42	6	8	26	65
43	6	10	26.8	67
44	6	15	28.8	72
45	7	9	29.6	74
46	7	10	30	75
47	7	15	32	80
48-127	Reserved			

8.2 AMR-WB+ Transport Interface Format

The transport interface format serves as an intermediate interface to the transport format. The transport interface frame contains a two-octet header followed by data octets.

The header in each frame contains the following two octets.

	MSB						**LSB**
Octet	**bit 8**	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**
1	0	Frame type (FT)						
2	TFI		0	ISF mode (5 bits)				

Frame type (FT) (7 bits): Indicates the frame type setting of the codec used for the corresponding frame (the combination of AMR-WB+ core and stereo mode, the AMR-WB mode, or comfort noise, as specified by Table 25 above).

Transport Frame Index (TFI) (2 bits): An index from 0 (first) to 3 (last) indicating this transport frame"s position in the superframe.

ISF index (5 bits): Indicates the internal sampling frequency employed for the corresponding frame. The index values correspond to internal sampling frequency as specified in Table 24 above. This field SHALL be set to 0 for operation according to the AMR-WB+ modes defined in table 21 (Frame types 0-13).

FT=14 (AUDIO_LOST) is used to indicate frames that are lost. NO_DATA (FT=15) frame could mean either that there is no data produced by the audio encoder for that frame or that no data for that frame is transmitted in the current packet (i.e., valid data for that frame could be sent in either an earlier or later packet). The duration for these non-included frames is dependent on the internal sampling frequency indicated by the ISF mode field.

For operation according to FT 0-13 the ISF field shall be set 0 and has no meaning. The frame length for that operation is fixed to 20 ms in time.

If receiving a frame with an FT value not defined the whole frame SHOULD be discarded and assumed erased.

The AMR-WB+ SCR/DTX is identical with AMR-WB SCR/DTX described in [8] and SHALL only be used in combination with the AMR-WB modes (0-8).

The audio data follows the header octets. The number of data octets per frame corresponding to a certain frame type is given in Table 25.

Example

The following diagram (Table 26) shows a frame of AMR-WB+ using 14 kbit/s frame type (FT=26) with a frame length of 35 octets (280 bits). The internal sampling frequency in this example is 25.6 kHz (ISF mode = 8). FT 26 corresponds to mono mode 0 (208 bits/frame) and stereo mode 4 (72 bits/frame). The frame is the first frame in the superframe (TFI=0).

The data octets are placed according to the detailed bit allocation given in tables 14 to 20. The first bit of the AMR-WB+ data b0 is placed in bit 8 of octet 3.

Table 26: AMR-WB+ transport interface format for 14 kbit/s operation with ISF mode 8 (bit rate factor=1).

	MSB							LSB
Octet	**bit 8**	**bit 7**	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**
1	FT = 26							
	0	0	0	1	1	0	1	0
2	TFI=0		ISF = 8					
	0	0	0	0	1	0	0	0
3	AMR-WB+ data (octet 1)							
	b0	b1	b2	b3	b4	b5	b6	b7
4..27	AMR-WB+ data (octets 2 to 25)							
	b8	…	…	…	…	…	…	…
28	AMR-WB+ data (octet 26)							
	b200	b201	b202	b203	B204	b205	b206	b207
29	AMR-WB+ data (octet 27)							
	s0	s1	s2	s3	s4	s5	s6	s7
30..36	AMR-WB+ data (octet 28 to 34)							
	s8	…	…	…	…	…	…	…
37	AMR-WB+ data (octet 35)							
	S64	S65	S66	S67	S68	S69	S70	S71

8.3 AMR-WB+ File Storage Format

This format is relevant only for file storage and defines a storage unit contained in an AMR-WB+ sample of a 3GP file [9]. It is quite similar to transport interface format with the exception that the two-octet header is used once per superframe for AMR-WB+ extension modes and once per frame for AMR-WB modes. Note that in AMR-WB+, the operation code and internal sampling frequency can be switched only on a superframe basis boundaries so the header octets are needed only once per superframe.
All media streams in a 3GP file are stored in timed units called samples. This format defines the syntax of the basic component of a sample, which is here called a storage unit..

A storage unit consists of a two-octet header followed by data octets corresponding to either:

1. A whole superframe (4 transport frames) when FT = 10..13 or OC = 16…47 .
2. A frame otherwise

For the first case, the number of data octets per superframe is given by 4 times the number of octets per frame (the right-most column in Table 25).

The length of an AMR-WB+ storage unit in ms (corresponding to one superframe) depends on the internal sample frequency and given by 80×ISF/25600 where ISF is the internal sampling frequency in Hz (ISF modes are shown in Table 24).

The header in each storage unit contains the following two octets.

	MSB							LSB
Octet	**bit 8**	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**
1	0	Frame type (7 bits)						
2	0	0	0	ISF mode (5 bits)				

Frame type (FT) (7 bits): Indicates the frame type setting of the codec used for the corresponding frame (the combination of AMR-WB+ core and stereo mode, the AMR-WB mode, or comfort noise, as specified by Table 25 above).

ISF index (5 bits): Indicates the internal sampling frequency employed for the corresponding frame. The index values correspond to internal sampling frequency as specified in Table 24 above. This field SHALL be set to 0 for operation according to the AMR-WB+ modes defined in table 21 (frame types 0-13).

For frame types according to FT 0-13 the ISF field shall be set 0 and has no meaning. The frame length for that operation is fixed to 20 ms in time.

The audio data follows the header octets. The number of data octets per storage unit corresponding to frame types 10..13 and 16…47 are given as 4 times the number of octets per frame (right-most column in Table 25), for the other frame types, the number of octets are those corresponding to 1 frame only.

It should be noticed that when FT <10, i.e. AMR-WB frames, the original AMR-WB storage format should be preferred in order to ensure backward decoding compatibility.

Example

The following diagram (Table 27) shows a storage sample of AMR-WB+ using 14 kbit/s frame types (FT=26) with a superframe length of 4×35=140 octets. The internal sampling frequency in this example is 25.6 kHz (ISF mode = 8). FT 26 corresponds to mono mode 0 (208 bits/frame) and stereo mode 4 (72 bits/frame).

The data octets are packetized according to the detailed bit allocation given in tables 14 to 20. The first bit of the AMR-WB+ data b0 is placed in bit 8 of octet 3.

Table 27: AMR-WB+ storage sample (superframe) for 14 kbit/s operation with ISF mode 8 (bit rate factor=1).

	MSB							LSB
Octet	**bit 8**	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**
1	FT = 26							
	0	0	0	1	1	0	1	0
2	ISF = 8							
	0	0	0	0	1	0	0	0
3	Frame 1 AMR-WB+ data (octet 1)							
	b0	b1	b2	b3	b4	b5	b6	b7
4..27	Frame 1 AMR-WB+ data (octets 2 to 25)							
	b8	…	…	…	…	…	…	…
28	Frame 1 AMR-WB+ data (octet 26)							
	b200	b201	b202	b203	B204	b205	b206	b207
29	Frame 1 AMR-WB+ data (octet 27)							
	s0	s1	s2	s3	s4	s5	s6	s7
30..36	Frame 1 AMR-WB+ data (octet 28 to 34)							
	s8	…	…	…	…	…	…	…
37	Frame 1 AMR-WB+ data (octet 35)							
	S64	S65	S66	S67	S68	S69	S70	S71
38	Frame 2 AMR-WB+ data (octet 1)							
	b0	b1	b2	b3	b4	b5	b6	b7
39..62	Frame 2 AMR-WB+ data (octets 2 to 25)							
	b8	…	…	…	…	…	…	…
63	Frame 2 AMR-WB+ data (octet 26)							
	b200	b201	b202	b203	B204	b205	b206	b207
64	Frame 2 AMR-WB+ data (octet 27)							
	s0	s1	s2	s3	s4	s5	s6	s7
65..71	Frame 2 AMR-WB+ data (octet 28 to 34)							
	s8	…	…	…	…	…	…	…
72	Frame 2 AMR-WB+ data (octet 35)							
	S64	S65	S66	S67	S68	S69	S70	S71
73	Frame 3 AMR-WB+ data (octet 1)							
	b0	b1	b2	b3	b4	b5	b6	b7
74..97	Frame 3 AMR-WB+ data (octets 2 to 25)							
	b8	…	…	…	…	…	…	…
98	Frame 3 AMR-WB+ data (octet 26)							
	b200	b201	b202	b203	B204	b205	b206	b207
99	Frame 3 AMR-WB+ data (octet 27)							
	s0	s1	s2	s3	s4	s5	s6	s7
100..106	Frame 3 AMR-WB+ data (octet 28 to 34)							
	s8	…	…	…	…	…	…	…
107	Frame 3 AMR-WB+ data (octet 35)							
	S64	S65	S66	S67	S68	S69	S70	S71
108	Frame 4 AMR-WB+ data (octet 1)							
	b0	b1	b2	b3	b4	b5	b6	b7
109..132	Frame 4 AMR-WB+ data (octets 2 to 25)							
	b8	…	…	…	…	…	…	…
133	Frame 4 AMR-WB+ data (octet 26)							
	b200	b201	b202	b203	B204	b205	b206	b207
134	Frame 4 AMR-WB+ data (octet 27)							
	s0	s1	s2	s3	s4	s5	s6	s7
135..141	Frame 4 AMR-WB+ data (octet 28 to 34)							
	s8	…	…	…	…	…	…	…
142	Frame 4 AMR-WB+ data (octet 35)							
	S64	S65	S66	S67	S68	S69	S70	S71

Annex A (informative): Change history

Change history							
Date	**TSG SA#**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2004-09	25	SP-040639	-	-	Approved at TSG SA#25	2.0.0	6.0.0
2004-12	26	SP-040841	001		Correction of stereo bit allocation tables	6.0.0	6.1.0
2004-12	26	SP-040841	002		Correction of storage format for AMR-WB+	6.0.0	6.1.0
2004-12	26	SP-040841	003	1	Editorial changes	6.0.0	6.1.0
2004-12	26	SP-040841	004		Editorial changes. Note that in Tdoc S4-040722, this CR was meant to be CR 004 and not CR 012; a remark was put in the CR database.	6.0.0	6.1.0
2005-03	27	SP-050096	005	1	Update for TCX coding mode selection table	6.1.0	6.2.0
2005-06	28	SP-050252	006		Correction of a value in Table 21	6.2.0	6.3.0
2007-03	35	SP-070029	0007	1	Reference to users guide	6.3.0	7.0.0
2008-12	42				Version for Release 8	7.0.0	8.0.0

History

Document history		
V8.0.0	January 2009	Publication