International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.722

(09/2012)

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Digital terminal equipments – Coding of voice and audio signals

7 kHz audio-coding within 64 kbit/s

Recommendation ITU-T G.722

International Telecommunication Union

ITU-T G-SERIES RECOMMENDATIONS

TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

INTERNATIONAL TELEPHONE CONNECTIONS AND CIRCUITS	G.100–G.199
GENERAL CHARACTERISTICS COMMON TO ALL ANALOGUE CARRIER-TRANSMISSION SYSTEMS	G.200–G.299
INDIVIDUAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON METALLIC LINES	G.300–G.399
GENERAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON RADIO-RELAY OR SATELLITE LINKS AND INTERCONNECTION WITH METALLIC LINES	G.400–G.449
COORDINATION OF RADIOTELEPHONY AND LINE TELEPHONY	G.450–G.499
TRANSMISSION MEDIA AND OPTICAL SYSTEMS CHARACTERISTICS	G.600–G.699
DIGITAL TERMINAL EQUIPMENTS	G.700–G.799
General	G.700–G.709
Coding of voice and audio signals	**G.710–G.729**
Principal characteristics of primary multiplex equipment	G.730–G.739
Principal characteristics of second order multiplex equipment	G.740–G.749
Principal characteristics of higher order multiplex equipment	G.750–G.759
Principal characteristics of transcoder and digital multiplication equipment	G.760–G.769
Operations, administration and maintenance features of transmission equipment	G.770–G.779
Principal characteristics of multiplexing equipment for the synchronous digital hierarchy	G.780–G.789
Other terminal equipment	G.790–G.799
DIGITAL NETWORKS	G.800–G.899
DIGITAL SECTIONS AND DIGITAL LINE SYSTEM	G.900–G.999
MULTIMEDIA QUALITY OF SERVICE AND PERFORMANCE – GENERIC AND USER-RELATED ASPECTS	G.1000–G.1999
TRANSMISSION MEDIA CHARACTERISTICS	G.6000–G.6999
DATA OVER TRANSPORT – GENERIC ASPECTS	G.7000–G.7999
PACKET OVER TRANSPORT ASPECTS	G.8000–G.8999
ACCESS NETWORKS	G.9000–G.9999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T G.722

7 kHz audio-coding within 64 kbit/s

Summary

Recommendation ITU-T G.722 describes the characteristics of an audio wideband (WB, 50 to 7 000 Hz) coding system which may be used for a variety of higher quality speech applications. The coding system uses sub-band adaptive differential pulse code modulation (SB-ADPCM) within a bit rate of 64 kbit/s. The system is henceforth referred to as 64 kbit/s (7 kHz) audio-coding. In the SB-ADPCM technique used, the frequency band is split into two sub-bands (higher and lower) and the signals in each sub-band are encoded using ADPCM. The system has three basic modes of operation corresponding to the bit rates used for 7 kHz audio-coding: 64, 56 and 48 kbit/s. The latter two modes allow an auxiliary data channel of 8 and 16 kbit/s, respectively, to be provided within 64 kbit/s by making use of bits from the lower sub-band. Erratum 1 was incorporated in this new edition, as well as some additional typos identified within the main body of ITU-T G.722.

Annex A provides three frequency masks that can be used to simplify evaluation of the mass-produced equipment using ITU-T G.722 codecs, and make easier checks carried out during installation. The masks therein are specifically not intended to supplant any requirements of this Recommendation, but rather to suggest the needs of acceptance testing for production quantities of equipment using ITU-T G.722 codecs. They concern the measure of the signal-to-total distortion ratio in a loop with SB-ADPCM. Thus, these specifications do not aim at taking the place of the test digital sequences of the ITU-T G.722 algorithm, but rather to ensure, once these sequences have been checked on a first model, that the quality of the equipment using these codecs is maintained.

Annex B describes a scalable superwideband (SWB, 50-14 000 Hz) speech and audio-coding algorithm operating at 64, 80 and 96 kbit/s. The ITU-T G.722 superwideband extension codec is interoperable with ITU-T G.722. The output of the ITU-T G.722 SWB coder has a bandwidth of 50-14 000 Hz. The coder operates with 5 ms frames, has an algorithmic delay of 12.3125 ms and a worst case complexity of 22.76 WMOPS. By default, the encoder input and decoder output are sampled at 32 kHz. The superwideband encoder for improved ITU-T G.722 64 kbit/s core produces an embedded bitstream structured in two layers corresponding to two available bit rates from 80 to 96 kbit/s. The superwideband encoder for improved ITU-T G.722 56 kbit/s core produces an embedded bitstream structured in one layer corresponding to one available bit rate of 64 kbit/s. This 64 kbit/s mode is also scalable with the 80 kbit/s and 96 kbit/s modes. The bitstream can be truncated at the decoder side or by any component of the communication system to instantaneously adjust the bit rate to the desired value (96 kbit/s – 80 kbit/s – 64 kbit/s) with no need for out-of-band signalling. The underlying algorithm includes three main parts: higher band enhancements, bandwidth extension (BWE) and transform coding in modified discrete cosine transform (MDCT) domain based on algebraic vector quantization (AVQ). In this revised version, an update was done to the text vectors of Annex B, so they can better assist in checking compliance of implementations.

Annex C describes an alternative implementation of ITU-T G.722 Annex B based on floating-point arithmetic. While Annex B provides a bit-exact, fixed-point specification with the fixed-point C-source code available from the ITU-T, alternative floating implementation is useful for platforms equipped with floating-point processors. This alternative floating-point arithmetic was found to be fully interoperable with Annex B in all configurations including the cross configurations.

Annex D describes a stereo extension of the wideband codec ITU-T G.722 and its superwideband extension, ITU-T G.722 Annex B. It is optimized for the transmission of stereo signals with limited additional bitrate, while keeping full compatibility with both codecs. Annex D operates from 64 to 128 kbit/s with four superwideband stereo bitrates at 80, 96, 112 and 128 kbit/s and two wideband stereo bitrates at 64 and 80 kbit/s. The wideband stereo modes are backward compatible with legacy ITU-T G.722, while the superwideband modes offer the backward compatibility with both mono

wideband ITU-T G.722 and superwideband ITU-T G.722 Annex B. The stereo codec operates on 5 ms frames with an algorithmic delay of 13.625 ms for wideband stereo and 15.9375 ms for superwideband stereo. The encoder input and decoder output are sampled at 16 kHz and 32 kHz for wideband and superwideband operating modes respectively. The underlying algorithm includes three main parts: stereo parameter analysis and down-mix at the encoder and stereo synthesis at the decoder. The first stereo extension layer is an 8 kbit/s layer comprising the basic stereo parameters, wideband inter-channel time difference/inter-channel phase difference/inter-channel coherence and sub-band inter-channel level differences. The second stereo layer, also an 8 kbit/s layer, enhances the stereo image by encoding low frequency sub-band inter-channel phase differences. Finally, the third stereo layer is a 16 kbit/s layer. In this last layer, the inter-channel phase differences of a larger bandwidth are transmitted which allow to further improve the stereo image. The bitstream can be truncated by the decoder, or by any components of the communication system, to instantaneously adjust the bitrate to the desired value, including wideband ITU-T G.722 and superwideband ITU-T G.722 Annex B bitrates, with no need for out-of-band signalling.

Networking aspects and test sequences for the main body algorithm are addressed in Appendices I and II respectively to this Recommendation. In this new edition, Appendix II was updated to reflect a restructuring of the test sequences for ITU-T G.722 main body.

Packet loss concealment (PLC) algorithms, also known as frame erasure concealment algorithms, hide transmission losses in audio systems where the input signal is encoded and packetized, sent over a network, received and decoded before play out. PLC algorithms can be found in most standard recent speech coders. ITU-T G.722 was initially designed without such a feature. Therefore, Appendices III and IV provide two PLC mechanisms for ITU-T G.722. The algorithms in both appendices were verified to have high quality performance with alternative quality/complexity trade-offs. At an additional complexity of 2.8 WMOPS worst-case and 2 WMOPS average compared with the ITU-T G.722 decoder without PLC, the ITU-T G.722 PLC algorithm described in Appendix III provides better speech quality whereas the ITU-T G.722 PLC specified in ITU-T G.722 Appendix IV provides lower complexity adding almost no additional complexity to that of the main body ITU-T G.722 decoding (worst-case additional complexity is 0.07 WMOPS).

The algorithm in Appendix III performs the packet loss concealment in the 16 kHz output domain of the ITU-T G.722 decoder. Periodic waveform extrapolation is used to fill in the waveform of lost packets, mixing with filtered noise according to signal characteristics prior to the loss. The extrapolated 16 kHz signal is passed through the QMF analysis filter bank, and the sub-band signals are passed to partial sub-band ADPCM encoders to update the states of the sub-band ADPCM decoders. Additional processing takes place for each packet loss in order to provide a smooth transition from the extrapolated waveform to the waveform decoded from the received packets. Among other things, the states of the sub-band ADPCM decoders are phase aligned with the first received packet after a packet loss, and the decoded waveform is time-warped in order to align with the extrapolated waveform before the two are overlap-added to smooth the transition. For protracted packet loss, the algorithm gradually mutes the output. The algorithm operates on an intrinsic 10-ms frame size. It can operate on any packet or frame size that is a multiple of 10 ms. The longer input frame becomes a super frame, for which the packet loss concealment is called an appropriate number of times at its intrinsic frame size of 10 ms. It results in no additional delay when compared with regular ITU-T G.722 decoding using the same frame size.

In Appendix IV, the decoder comprises three stages: lower sub-band decoding, higher sub-band decoding and quadrature mirror filter (QMF) synthesis. In the absence of frame erasures, the decoder structure is identical to ITU-T G.722, except for the storage of the two decoded signals, of the higher and lower sub-bands. In case of frame erasures, the decoder is informed by the bad frame indication (BFI) signalling. It then performs an analysis of the past lower-band reconstructed signal and extrapolates the missing signal using linear-predictive coding (LPC), pitch-synchronous period repetition and adaptive muting. Once a good frame is received, the decoded signal is cross-faded with the extrapolated signal. In the higher sub-band, the decoder repeats the previous frame

pitch-synchronously, with adaptive muting and high-pass post-processing. The adaptive differential pulse code modulation (ADPCM) states are updated after each frame erasure.

Appendix V defines a coding scheme for mid-side (MS) stereo using the superwideband extension defined in Annex B of [ITU-T G.722]. By introducing the mid-side stereo coding into stereo terminals, interoperability with the monaural devices could be obtained in very low complexity. The basic coding scheme is as follows: two channels of the left-right (LR) stereo are converted to those of the mid-side stereo and then the signals of each channel are independently encoded using ITU-T G.722 Annex B; then, at the decoder side, the mid-side channels of the bitstream from the encoder are decoded respectively and then the decoded signals of the mid-side channels are reversed to those of the LR channels. The LR-MS conversion and its inverse are conducted in a conventional way. On the encoder side, two additional arithmetic operations per sample are required for the LR-MS conversion and one operator for the MS-LR conversion in the decoder. In an STL2009 (see ITU-T G.191) basic operator implementation, the conversion complexity amounts to about 0.2 WMOPS in total. The coding algorithm for each channel is identical to the one in Recommendation ITU-T G.722 Annex B.

Annexes B, C and D contain an electronic attachment provided with the ANSI C source code, which is an integral part of these annexes. ANSI C source code is also provided as an integral part of Appendices III and IV.

NOTE – An ANSI-C code reference implementation for the algorithm in the main body of ITU-T G.722 is found in the ITU-T G722 module of the ITU-T G.191 Software Tools Library.

Test sequences are provided for compliance testing of the ITU-T G.722 algorithm in the main body of this Recommendation. Test vectors are provided to assist in checking the correct operation of Annexes B, C and D and Appendices III and IV.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T G.722	1987-02-28	XVIII
2.0	ITU-T G.722	1988-11-25	
2.1	ITU-T G.722 (1988) App. II	1988-11-25	
2.2	ITU-T G.722 (1988) Annex A	1993-03-12	XV
2.3	ITU-T G.722 (1988) App. III	2006-11-24	16
2.4	ITU-T G.722 (1988) App. IV	2006-11-24	16
2.5	ITU-T G.722 (1988) App. IV	2007-07-06	16
2.6	ITU-T G.722 (1988) App. IV	2009-11-06	16
2.7	ITU-T G.722 (1988) Amd. 1	2010-11-13	16
2.8	ITU-T G.722 (1988) Amd. 2	2011-03-25	16
3.0	ITU-T G.722	2012-09-13	16

Keywords

ADPCM, ITU-T G.722, ITU-T G.722 Annex B, packet loss concealment, PLC, stereo coding, sub-band coding, superwideband, wideband.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Electronic attachment: ANSI-C code for Annexes B, C and D, and Appendix III, and test sequences for Appendix II

Recommendation ITU-T G.722

7 kHz audio-coding within 64 kbit/s

1 General

1.1 Scope and outline description

This Recommendation describes the characteristics of an audio (50 to 7 000 Hz) coding system which may be used for a variety of higher quality speech applications. The coding system uses sub-band adaptive differential pulse code modulation (SB-ADPCM) within a bit rate of 64 kbit/s. The system is henceforth referred to as 64 kbit/s (7 kHz) audio-coding. In the SB-ADPCM technique used, the frequency band is split into two sub-bands (higher and lower) and the signals in each sub-band are encoded using ADPCM. The system has three basic modes of operation corresponding to the bit rates used for 7 kHz audio-coding: 64, 56 and 48 kbit/s. The latter two modes allow an auxiliary data channel of 8 and 16 kbit/s respectively to be provided within the 64 kbit/s by making use of bits from the lower sub-band.

Figure 1 identifies the main functional parts of the 64 kbit/s (7 kHz) audio codec as follows:

i) 64 kbit/s (7 kHz) audio encoder comprising:
   - a transmit audio part which converts an audio signal to a uniform digital signal which is coded using 14 bits with 16 kHz sampling;
   - a SB-ADPCM encoder which reduces the bit rate to 64 kbit/s.

ii) 64 kbit/s (7 kHz) audio decoder comprising:
   - a SB-ADPCM decoder which performs the reverse operation to the encoder, noting that the effective audio-coding bit rate at the input of the decoder can be 64, 56 or 48 kbit/s depending on the mode of operation;
   - a receive audio part which reconstructs the audio signal from the uniform digital signal which is encoded using 14 bits with 16 kHz sampling.

The following two parts, identified in Figure 1 for clarification, will be needed for applications requiring an auxiliary data channel within the 64 kbit/s:

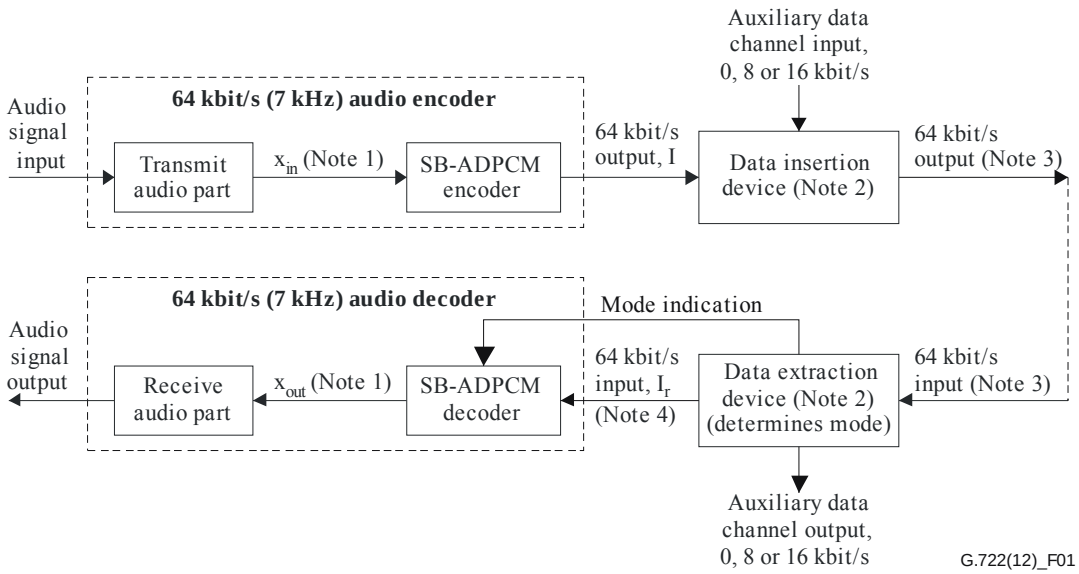
– a data insertion device at the transmit end which makes use of, when needed, 1 or 2 audio bits per octet depending on the mode of operation and substitutes data bits to provide an auxiliary data channel of 8 or 16 kbit/s respectively;

– a data extraction device at the receive end which determines the mode of operation according to a mode control strategy and extracts the data bits as appropriate.

Clause 1.2 contains a functional description of the transmit and receive audio parts, clause 1.3 describes the modes of operation and the implication of inserting data bits on the algorithms, whilst clauses 1.4 and 1.5 provide the functional descriptions of the SB-ADPCM encoding and decoding algorithms respectively. Clause 1.6 deals with the timing requirements. Clause 2 specifies the transmission characteristics of the 64 kbit/s (7 kHz) audio codec and of the transmit and receive audio parts, clauses 3 and 4 give the principles of the SB-ADPCM encoder respectively whilst clauses 5 and 6 specify the computational details of the quadrature mirror filters (QMF) and of the ADPCM encoders and decoders respectively.

Networking aspects and test sequences are addressed in Appendices I and II respectively to this Recommendation.

Recommendation ITU-T G.725 contains specifications for in-channel handshaking procedures for terminal identification and for mode control strategy, including interworking with existing 64 kbit/s PCM terminals.



NOTE 1 – X_{in} and X_{out} are digital signals uniformly coded with 14 bits and 16 kHz sampling.
NOTE 2 – These devices are only necessary for applications requiring an auxiliary data channel within the 64 kbit/s.
NOTE 3 – Comprises 64, 56 or 48 kbit/s for audio coding and 0, 8 or 16 kbit/s for data.
NOTE 4 – 64 kbit/s signal comprising 64, 56 or 48 kbit/s for audio coding depending on the mode of operation.

Figure 1 – Simplified functional block diagram

1.2 Functional description of the audio parts

Figure 2 shows a possible arrangement of audio parts in a 64 kbit/s (7 kHz) audio-coding terminal. The microphone, pre-amplifier, power amplifier and loudspeaker are shown simply to identify the audio parts and are not considered further in this Recommendation.

In order to facilitate the measurement of the transmission characteristics as specified in clause 2, test points A and B need to be provided as shown. These test points may either be for test purposes only or, where the audio parts are located in different units from the microphone, loudspeaker, etc., correspond to physical interfaces.

The transmit and receive audio parts comprise either the following functional units or any equivalent items satisfying the specifications of clause 2:

i) transmit:
   - an input level adjustment device;
   - an input anti-aliasing filter;
   - a sampling device operating at 16 kHz;
   - an analogue-to-uniform digital converter with 14 bits and with 16 kHz sampling;

ii) receive:
   - a uniform digital-to-analogue converter with 14 bits and with 16 kHz sampling;
   - a reconstructing filter which includes x/sin x correction;
   - an output level adjustment device.

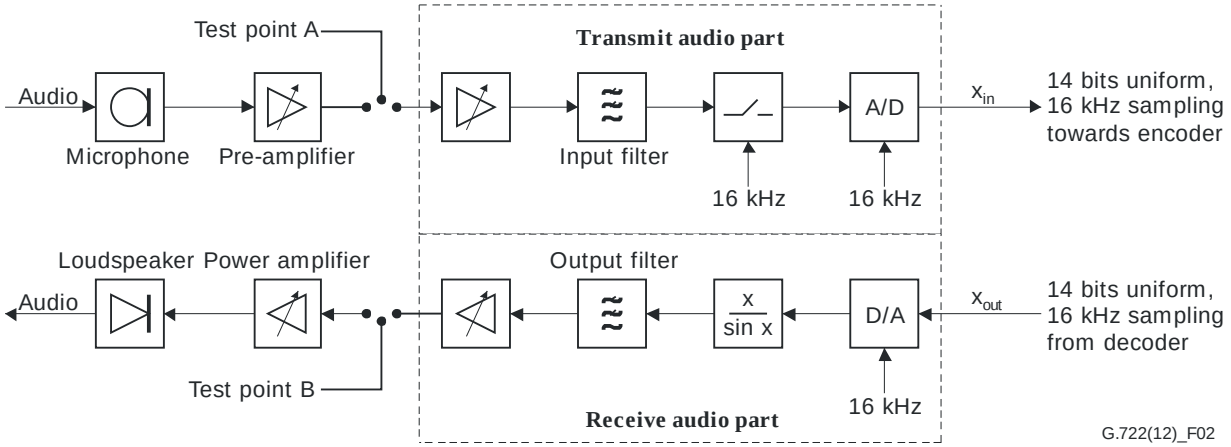


Figure 2 – Possible implementation of the audio parts

1.3 Possible modes of operation and implications of inserting data

The three basic possible modes of operation which correspond to the bit rates available for audio-coding at the input of the decoder are defined in Table 1.

Table 1 – Basic possible modes of operation

Mode	7 kHz audio-coding bit rate	Auxiliary data channel bit rate
1	64 kbit/s	0 kbit/s
2	56 kbit/s	8 kbit/s
3	48 kbit/s	16 kbit/s

See Appendix I for examples of applications using one or several of these modes and for their corresponding subjective quality.

The 64 kbit/s (7 kHz) audio encoder uses 64 kbit/s for audio-coding at all times irrespective of the mode of operation. The audio-coding algorithm has been chosen such that, without sending any indication to the encoder, the least significant bit or two least significant bits of the lower sub-band may be used downstream from the 64 kbit/s (7 kHz) audio encoder in order to substitute the auxiliary data channel bits. However, to maximize the audio performance for a given mode of operation, the 64 kbit/s (7 kHz) audio decoder must be optimized to the bit rate available for audio-coding. Thus, this Recommendation describes three variants of the SB-ADPCM decoder and, for applications requiring an auxiliary data channel; an indication must be forwarded to select in the decoder the variant appropriate to the mode of operation. Figure 1 illustrates the arrangement. It should be noted that the bit rate at the input of the 64 kbit/s (7 kHz) audio decoder is always 64 kbit/s but comprising 64, 56 or 48 kbit/s for audio-coding depending on the mode of operation. From an algorithm viewpoint, the variant used in the SB-ADPCM decoder can be changed in any octet during the transmission. When no indication about the mode of operation is forwarded to the decoder, the variant corresponding to Mode 1 should be used.

A mode mismatch situation, where the variant used in the 64 kbit/s (7 kHz) audio decoder for a given octet does not correspond to the mode of operation, will not cause misoperation of the decoder. However, to maximize the audio performance, it is recommended that the mode control strategy adopted in the data extraction device should be such as to minimize the duration of the mode mismatch. Appendix I gives further information on the effects of a mode mismatch. To ensure compatibility between various types of 64 kbit/s (7 kHz) audio-coding terminals, it is recommended that, as a minimum, the variant corresponding to Mode 1 operation is always implemented in the decoder.

The mode control strategy could be derived from the auxiliary data channel protocol (see Recommendation ITU-T G.725).

1.4 Functional description of the SB-ADPCM encoder

Figure 3 is a block diagram of the SB-ADPCM encoder. A functional description of each block is given below in clauses 1.4.1 to 1.4.4.

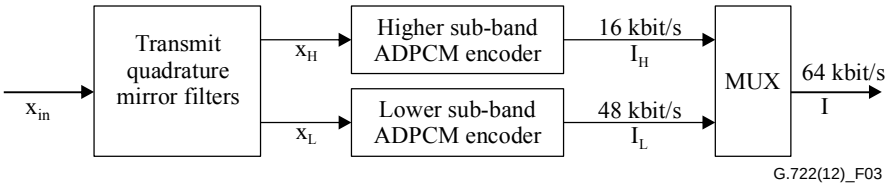


Figure 3 – Block diagram of the SB-ADPCM encoder

1.4.1 Transmit quadrature mirror filters (QMFs)

The transmit QMFs comprise two linear-phase non-recursive digital filters which split the frequency band 0 to 8 000 Hz into two sub-bands: the lower sub-band (0 to 4 000 Hz) and the higher sub-band (4 000 to 8 000 Hz). The input to the transmit QMFs, x_{in}, is the output from the transmit audio part and is sampled at 16 kHz. The outputs, x_L and x_H, for the lower and higher sub-bands respectively, are sampled at 8 kHz.

1.4.2 Lower sub-band ADPCM encoder

Figure 4 is a block diagram of the lower sub-band ADPCM encoder. The lower sub-band input signal, x_L after subtraction of an estimate, s_L, of the input signal produces the difference signal, e_L. An adaptive 60-level non-linear quantizer is used to assign six binary digits to the value of the difference signal to produce a 48 kbit/s signal, I_L.

In the feedback loop, the two least significant bits o*f* I_L are deleted to produce a 4-bit signal I_{Lt}, which is used for the quantizer adaptation and applied to a 15-level inverse adaptive quantizer to produce a quantized difference signal, d_{Lt}. The signal estimate, s_L is added to this quantized difference signal to produce a reconstructed version, r_{Lt}, of the lower sub-band input signal. Both the reconstructed signal and the quantized difference signal are operated upon by an adaptive predictor which produce the estimate, s_L, of the input signal, thereby completing the feedback loop.

4-bit operation, instead of 6-bit operation, in the feedback loops of both the lower sub-band ADPCM encoder, and the lower sub-band ADPCM decoder allows the possible insertion of data in the two least significant bits as described in clause 1.3 without causing misoperation in the decoder. Use of a 60-level quantizer (instead of 64-level) ensures that the pulse density requirements as described in Recommendation ITU-T G.802 are met under all conditions and in all modes of operation.

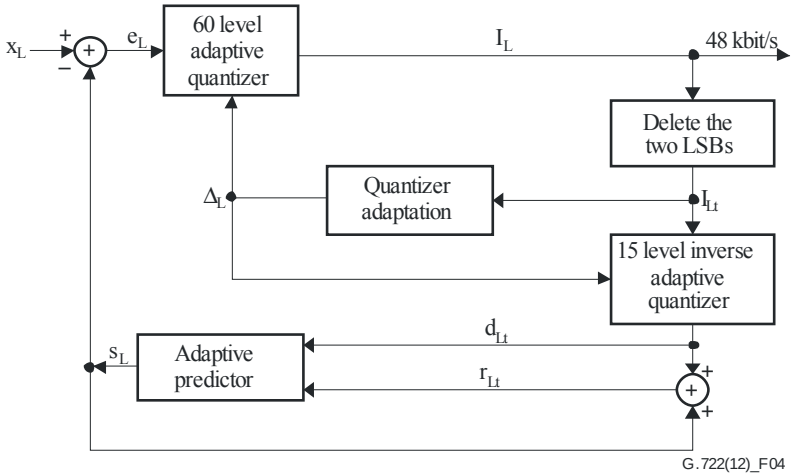


Figure 4 – Block diagram of the lower sub-band ADPCM encoder

1.4.3 Higher sub-band ADPCM encoder

Figure 5 is a block diagram of the higher sub-band ADPCM encoder. The higher sub-band input signal, x_H after subtraction of an estimate, s_H, of the input signal, produces the difference signal, e_H. An adaptive 4-level non-linear quantizer is used to assign two binary digits to the value of the difference signal to produce a 16 kbit/s signal, I_H.

An inverse adaptive quantizer produces a quantized difference signal, d_H, from these same two binary digits. The signal estimate, s_H, is added to this quantized difference signal to produce a reconstructed version, r_H, of the higher sub-band input signal. Both the reconstructed signal and the quantized difference signal are operated upon by an adaptive predictor which produces the estimate, s_H, of the input signal, thereby completing the feedback loop.

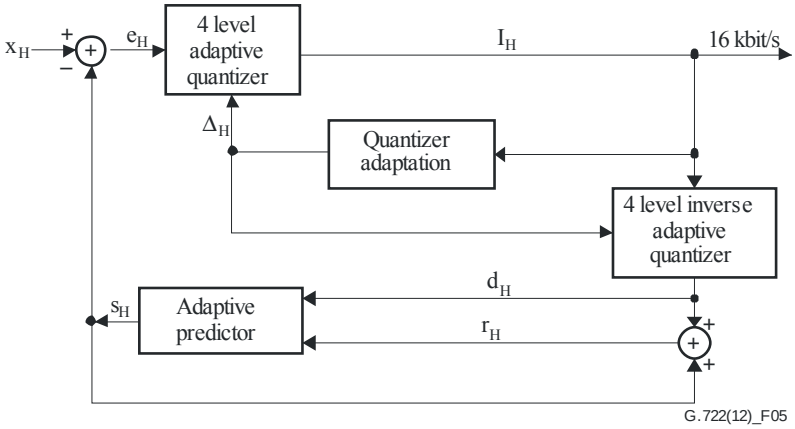


Figure 5 – Block diagram of the higher sub-band ADPCM encoder

1.4.4 Multiplexer

The multiplexer (MUX) shown in Figure 3 is used to combine the signals, I_L and I_H, from the lower and higher sub-band ADPCM encoders respectively into a composite 64 kbit/s signal, I, with an octet format for transmission.

The output octet format, after multiplexing, is as follows:

$$I_{H1}\ I_{H2}\ I_{L1}\ I_{L2}\ I_{L3}\ I_{L4}\ I_{L5}\ I_{L6}$$

where I_{H1} is the first bit transmitted, and where I_{H1} and I_{L1} are the most significant bits of I_H and I_L respectively, whilst I_{H2} and I_{L6} are the least significant bits of I_H and I_L respectively.

1.5 Functional description of the SB-ADPCM decoder

Figure 6 is a block diagram of the SB-ADPCM decoder. A functional description of each block is given below in clauses 1.5.1 to 1.5.4.

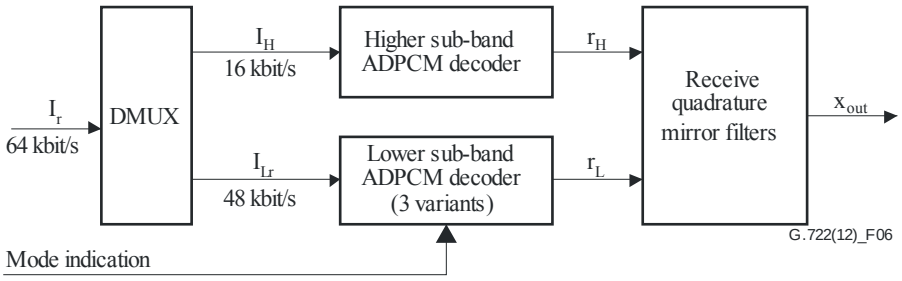


Figure 6 – Block diagram of the SB-ADPCM decoder

1.5.1 Demultiplexer

The demultiplexer (DMUX) decomposes the received 64 kbit/s octet-formatted signal, I_r, into two signals, I_{Lr} and I_H, which form the codeword inputs to the lower and higher sub-band ADPCM decoders respectively.

1.5.2 Lower sub-band ADPCM decoder

Figure 7 is a block diagram of the lower sub-band ADPCM decoder. This decoder can operate in any of three possible variants depending on the received indication of the mode of operation.

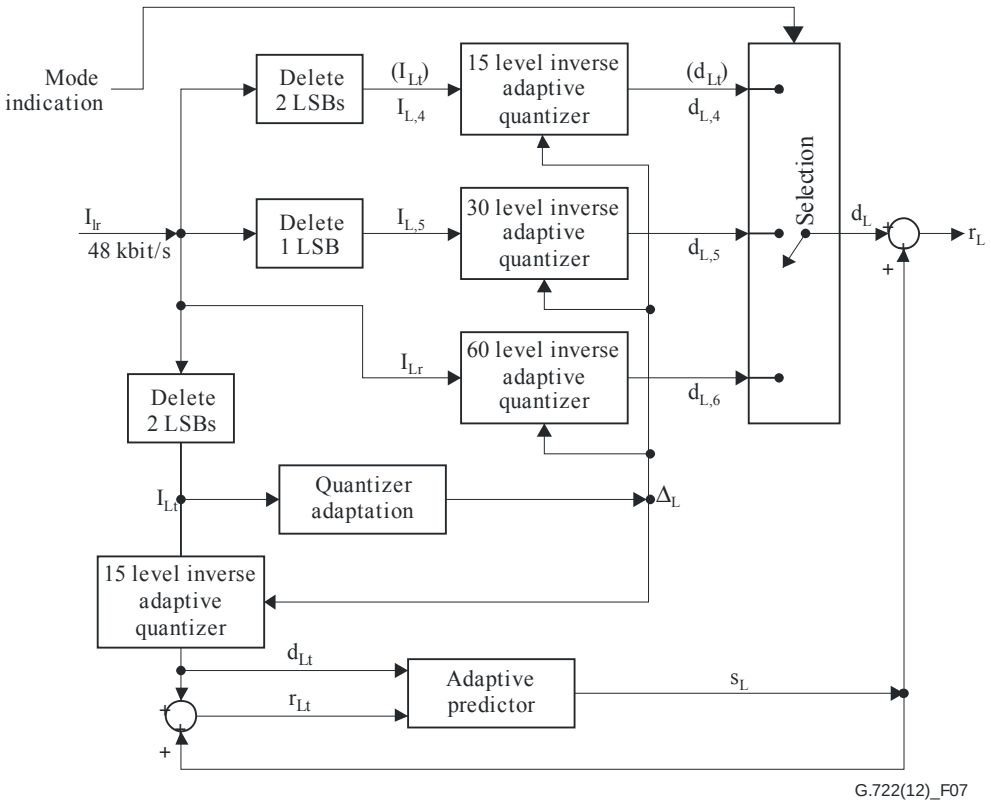


Figure 7 – Block diagram of the lower sub-band ADPCM decoder

The path which produces the estimate, s_L, of the input signal including the quantizer adaptation, is identical to the feedback portion of the lower sub-band ADPCM encoder described in clause 1.4.2. The reconstructed signal, r_L, is produced by adding to the signal estimate one of three possible quantized difference signals, $d_{L,6}$, $d_{L,5}$ or $d_{L,4}$ (= d_{Lt} – see Note), selected according to the received

indication of the mode of operation. For each indication, Table 2 shows the quantized difference signal selected, the inverse adaptive quantizer used and the number of least significant bits deleted from the input codeword.

Table 2 – Lower sub-band ADPCM decoder variants

Received indication of mode of operation	Quantized difference signal selected	Inverse adaptive quantizer used	Number of least significant bits deleted from input codeword, I_{Lr}
Mode 1	$d_{L,6}$	60-level	0
Mode 2	$d_{L,5}$	30-level	1
Mode 3	$d_{L,4}$	15-level	2

NOTE – For clarification purposes, all three inverse quantizers have been indicated in the upper portion of Figure 7. In an optimized implementation, the signal d_{Lt}, produced in the predictor loop, could be substituted for $d_{L,4}$.

1.5.3 Higher sub-band ADPCM decoder

Figure 8 is a block diagram of the higher sub-band ADPCM decoder. This decoder is identical to the feedback portion of the higher sub-band ADPCM encoder described in clause 1.4.3, the output being the reconstructed signal, r_H.

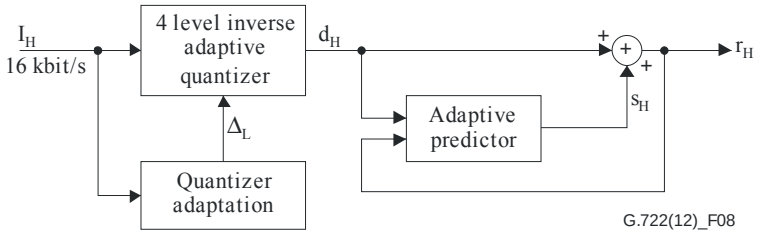


Figure 8 – Block diagram of the higher sub-band ADPCM decoder

1.5.4 Receive QMFs

The receive QMFs shown in Figure 6 are two linear-phase non-recursive digital filters which interpolate the outputs, r_L and r_H, of the lower and higher sub-band ADPCM decoders from 8 kHz to 16 kHz and which then produce an output, x_{out}, sampled at 16 kHz which forms the input to the receive audio parts.

Excluding the ADPCM coding processes, the combination of the transmit and the receive QMFs has an impulse response which closely approximates a simple delay whilst, at the same time, the aliasing effects associated with the 8 kHz sub-sampling are cancelled.

1.6 Timing requirements

64 kHz bit timing and 8 kHz octet timing should be provided by the network to the audio decoder.

For a correct operation of the audio-coding system, the precision of the 16 kHz sampling frequencies of the A/D and D/A converters must be better than $\pm 50 \cdot 10^{-6}$.

2 Transmission characteristics

2.1 Characteristics of the audio ports and the test points

Figure 2 indicates the audio input and output ports and the test points (A and B). It is for the designer to determine the characteristics of the audio ports and the test points (i.e., relative levels, impedances, whether balanced or unbalanced). The microphone, pre-amplifier, power amplifier and loudspeaker should be chosen with reference to the specifications of the audio parts: in particular their nominal bandwidth, idle noise and distortion.

It is suggested that input and output impedances should be high and low, respectively, for an unbalanced termination whilst for a balanced termination these impedances should be 600 ohms. However, the audio parts should meet all audio parts specifications for their respective input and output impedance conditions.

2.2 Overload point

The overload point for the analogue-to-digital and digital-to-analogue converters should be +9 dBm0 ± 0.3 dB. This assumes the same nominal speech level (see Recommendation ITU-T G.232) as for 64 kbit/s PCM, but with a wider margin for the maximum signal level which is likely to be necessary with conference arrangements. The measurement method of the overload point is under study.

2.3 Nominal reference frequency

Where a nominal reference frequency of 1 000 Hz is indicated below, the actual frequency should be chosen equal to 1 020 Hz. The frequency tolerance should be +2 to –7 Hz.

2.4 Transmission characteristics of the 64 kbit/s (7 kHz) audio codec

The values and limits specified below should be met with a 64 kbit/s (7 kHz) audio encoder and decoder connected back-to-back. For practical reasons, the measurements may be performed in a looped configuration as shown in Figure 9a). However, such a looped configuration is only intended to simulate an actual situation where the encoder and decoder are located at the two ends of a connection.

These limits apply to operation in Mode 1.

2.4.1 Nominal bandwidth

The nominal 3 dB bandwidth is 50 to 7 000 Hz.

2.4.2 Attenuation/frequency distortion

The variation with frequency of the attenuation should satisfy the limits shown in the mask of Figure 10. The nominal reference frequency is 1 000 Hz and the test level is –10 dBm0.

2.4.3 Absolute group delay

The absolute group delay, defined as the minimum group delay for a sine wave signal between 50 and 7 000 Hz, should not exceed 4 ms. The test level is –10 dBm0.

2.4.4 Idle noise

The unweighted noise power measured in the frequency range 50 to 7 000 Hz with no signal at the input port (test point A) should not exceed –66 dBm0. When measured in the frequency range 50 Hz to 20 kHz the unweighted noise power should not exceed –60 dBm0.

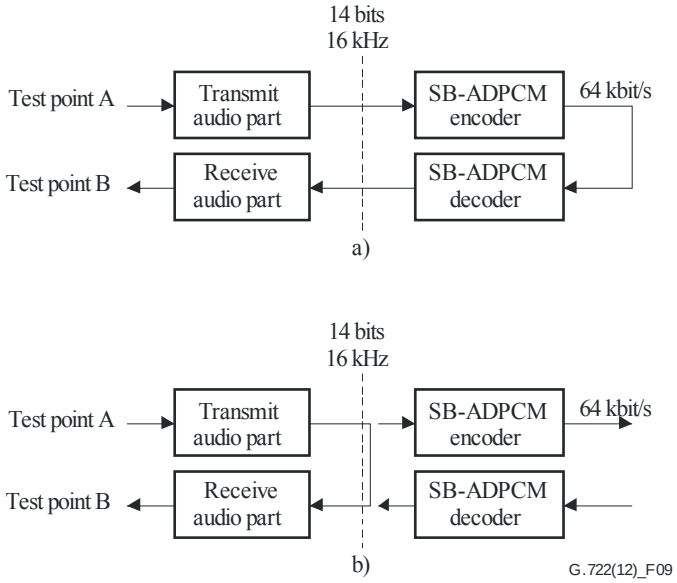


Figure 9 – Looped measurement configurations

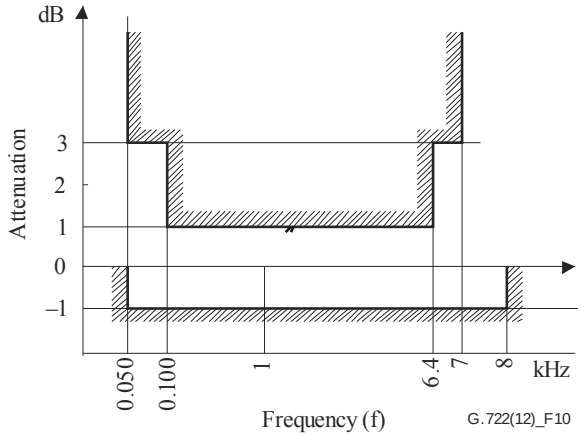


Figure 10 – Attenuation distortion versus frequency

2.4.5 Single frequency noise

The level of any single frequency (in particular 8 000 Hz, the sampling frequency and its multiples), measured selectively with no signal at the input port (test point A) should not exceed –70 dBm0.

2.4.6 Signal-to-total distortion ratio

Under study.

2.5 Transmission characteristics of the audio parts

When the measurements indicated below for the audio parts are from audio-to-audio, a looped configuration as shown in Figure 9b) should be used. The audio parts should also meet the specifications of clause 2.4 with the measurement configuration of Figure 9b).

2.5.1 Attenuation/frequency response of the input anti-aliasing filter

The in-band and out-of-band attenuation/frequency response of the input anti-aliasing filter should satisfy the limits of the mask shown in Figure 11. The nominal reference frequency is 1 000 Hz and the test level for the in-band characteristic is –10 dBm0. Appropriate measurements should be made to check the out-of-band characteristic taking into account the aliasing due to the 16 kHz sampling.

2.5.2 Attenuation/frequency response of the output reconstructing filter

The in-band and out-of-band attenuation/frequency response of the output reconstructing filter should satisfy the limits of the mask shown in Figure 12. The nominal reference frequency is 1 000 Hz and the test level for the in-band characteristic is –10 dBm0. Appropriate measurements should be made to check the out-of-band characteristic taking into account the aliasing due to the 16 kHz sampling. The mask of Figure 12 is valid for the whole of the receive audio part including any pulse amplitude modulation distortion and x/sin x correction.

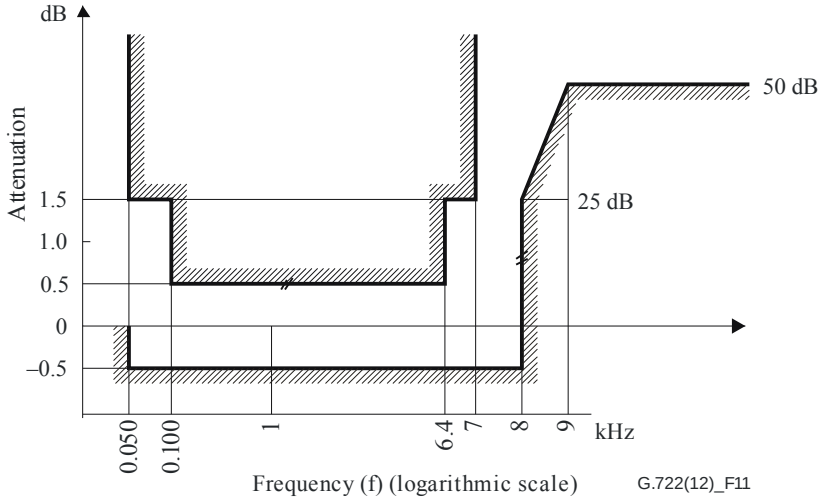


Figure 11 – Attenuation/frequency response of the input anti-aliasing filter

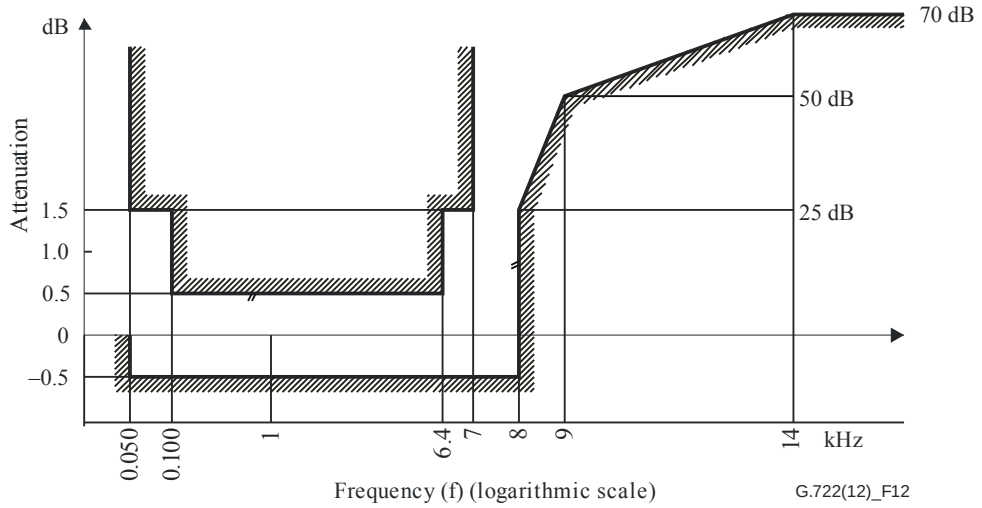


Figure 12 – Attenuation/frequency response of the output reconstructing filter (including x/sin x correction)

2.5.3 Group-delay distortion with frequency

The group-delay distortion, taking the minimum value of group delay as a reference, should satisfy the limits of the mask shown in Figure 13.

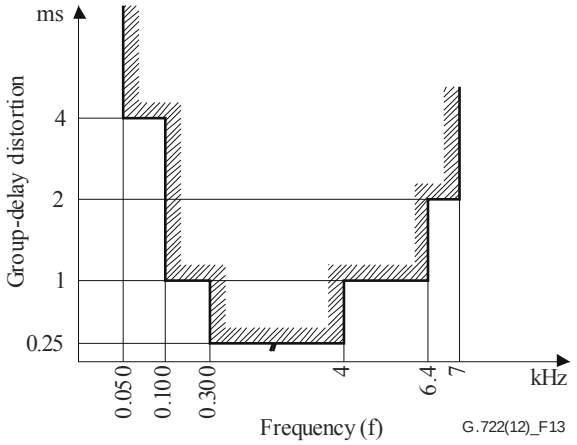


Figure 13 – Group-delay distortion versus frequency

2.5.4 Idle noise for the receive audio part

The unweighted noise power of the receive audio part measured in the frequency range 50 to 7 000 Hz with 14-bit all-zero signal at its input should not exceed –75 dBm0.

2.5.5 Signal-to-total distortion ratio as a function of input level

With a sine wave signal at a frequency excluding simple harmonic relationships with the 16 kHz sampling frequency, applied to test point A, the ratio of signal-to-total distortion power as a function of input level measured unweighted in the frequency range 50 to 7 000 Hz at test point B, should satisfy the limits of the mask shown in Figure 14. Two measurements should be performed, one at a frequency of about 1 kHz and the other at a frequency of about 6 kHz.

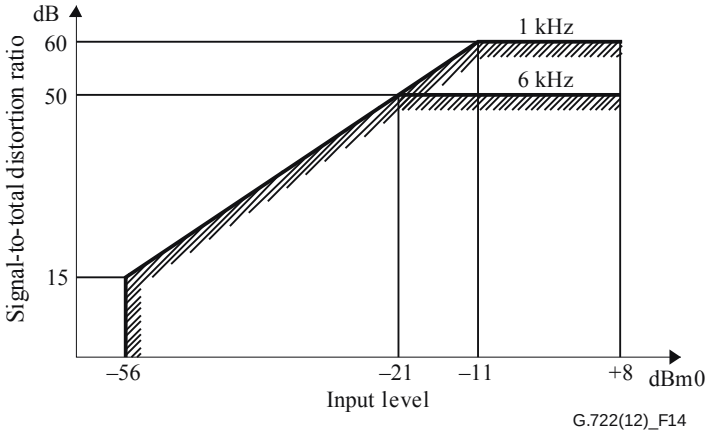


Figure 14 – Signal-to-total distortion ratio as a function of input level

2.5.6 Signal-to-total distortion ratio as a function of frequency

With a sine wave signal at a level of –10 dBm0 applied to test point A, the ratio of signal-to-total distortion power as a function of frequency measured unweighted in the frequency range 50 to 7 000 Hz at test point B should satisfy the limits of the mask shown in Figure 15.

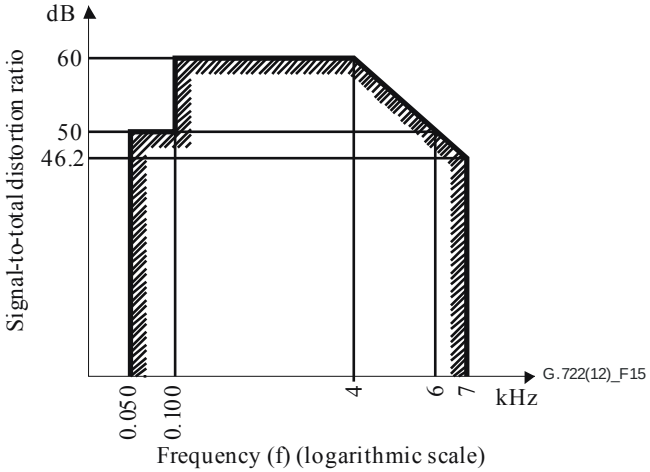


Figure 15 – Signal-to-total distortion ratio as a function of frequency

2.5.7 Variation of gain with input level

With a sine wave signal at the nominal reference frequency of 1 000 Hz, but excluding the sub-multiple of the 16 kHz sampling frequency, applied to test point A, the gain variation as a function of input level relative to the gain at an input level of –10 dBm0 measured selectively at test point B, should satisfy the limits of the mask shown in Figure 16.

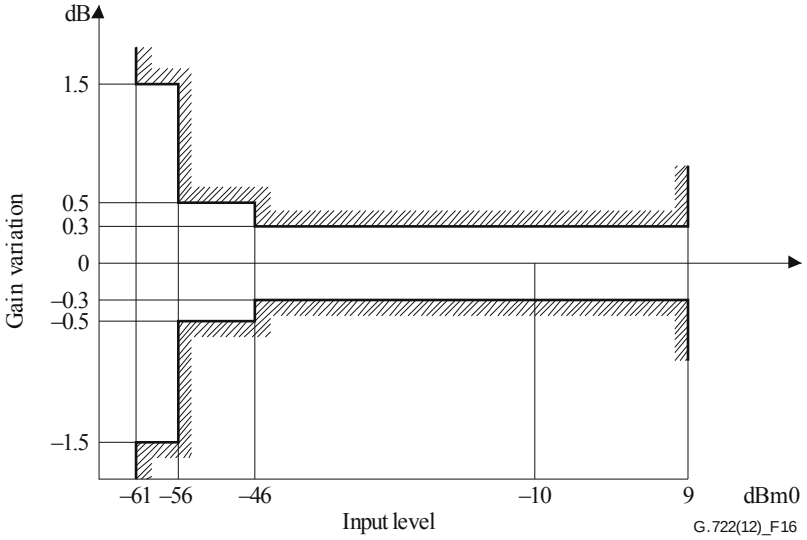


Figure 16 – Variation of gain with input level

2.5.8 Intermodulation

Under study.

2.5.9 Go/return crosstalk

The crosstalk from the transmit direction to the receive direction should be such that, with a sine wave signal at any frequency in the range 50 to 7 000 Hz and at a level of +6 dBm0 applied to test point A, the crosstalk level measured selectively at test point B should not exceed –64 dBm0. The measurement should be made with a 14-bit all-zero digital signal at the input to the receive audio part.

The crosstalk from the receive direction to the transmit direction should be such that, with a digitally simulated sine wave signal at any frequency in the range of 50 to 7 000 Hz and a level of +6 dBm0 applied to the input of the receive audio part, the crosstalk level measured selectively and

with the measurement made digitally at the output of the transmit audio part should not exceed −64 dBm0. The measurement should be made with no signal at test point A, but with the test point correctly terminated.

2.6 Transcoding to and from 64 kbit/s PCM

For compatibility reasons with 64 kbit/s PCM, transcoding between 64 kbit/s (7 kHz) audio-coding and 64 kbit/s PCM should take account of the relevant specifications of Recommendations ITU-T G.712, ITU-T G.713 and ITU-T G.714. When the audio signal is to be heard through a loudspeaker, more stringent specifications may be necessary. Further information may be found in Appendix I.

3 SB-ADPCM encoder principles

A block diagram of the SB-ADPCM encoder is given in Figure 3. Block diagrams of the lower and higher sub-band ADPCM encoders are given respectively in Figures 4 and 5.

Main variables used for the descriptions in clauses 3 and 4 are summarized in Table 3. In these descriptions, index (*j*) indicates a value corresponding to the current 16 kHz sampling interval, index (*j*-1) indicates a value corresponding to the previous 16 kHz sampling interval, index (*n*) indicates a value corresponding to the current 8 kHz sampling interval, and index (*n*-1) indicates a value corresponding to the previous 8 kHz sampling interval. Indices are not used for internal variables, i.e., those employed only within individual computational blocks.

3.1 Transmit QMF

A 24-coefficient QMF is used to compute the lower and higher sub-band signal components. The QMF coefficient values, h_i, are given in Table 4.

The output variables, $x_L(n)$ and $x_H(n)$, are computed in the following way:

$$x_L(n) = x_A + x_B \tag{3-1}$$

$$x_H(n) = x_A - x_B \tag{3-2}$$

$$x_A = \sum_{i=0}^{11} h_{2i} \cdot x_{in}(j-2i) \tag{3-3}$$

$$x_B = \sum_{i=0}^{11} h_{2i+1} \cdot x_{in}(j-2i-1) \tag{3-4}$$

3.2 Difference signal computation

The difference signals, $e_L(n)$ *and* $e_H(n)$, are computed by subtracting predicted values, $s_L(n)$ *and* $s_H(n)$, from the lower and higher sub-band input values, $x_L(n)$ and $x_H(n)$:

$$e_L(n) = x_L(n) - s_L(n) \tag{3-5}$$

$$e_H(n) = x_H(n) - s_H(n) \tag{3-6}$$

Table 3 – Variables used in the SB-ADPCM encoder and decoder descriptions

Variable	Description
x_{in}	Input value (uniform representation)
x_L, x_H	QMF output signals

Table 3 – Variables used in the SB-ADPCM encoder and decoder descriptions

Variable	Description
S_{Lp}, S_{Hp}	Pole-predictor output signals
$a_{L,i}$, $a_{H,i}$	Pole-predictor coefficients
r_L, r_{Lt}, r_H	Reconstructed signals (non truncated and truncated)
$b_{L,i}$, $b_{H,i}$	Zero-predictor coefficients
d_L, d_{Lt}, d_H	Quantized difference signals (non truncated and truncated)
S_{Lz}, S_{Hz}	Zero-predictor output signals
S_L, S_H	Predictor output signals
e_L, e_H	Difference signals to be quantized
∇_L, ∇_H	Logarithmic quantizer scale factors
∇_L, ∇_H	Quantizer scale factor (linear)
I_L, I_{Lt}, I_H	Codewords (non truncated and truncated)
P_{Lt}, P_H	Partially reconstructed signals
I_{Lr}	Received lower sub-band codeword
X_{out}	Output value (uniform)

NOTE – Variables used exclusively within one section are not listed. Subscripts L and H refer to lower sub-band and higher sub-band values. Subscript Lt denotes values generated from the truncated 4-bit codeword as opposed to the non truncated 6-bit (encoder) or 6-, 5- or 4-bit (decoder) codewords.

Table 4 – Transmit and receive OMF coefficient values

h_0, h_{23}	0.366211E-03
h_1, h_{22}	–0.134277E-02
h_2, h_{21}	–0.134277E-02
h_3, h_{20}	0.646973E-02
h_4, h_{19}	0.146484E-02
h_5, h_{18}	–0.190430E-01
h_6, h_{17}	0.390625E-02
h_7, h_{16}	0.441895E-01
h_8, h_{15}	–0.256348E-01
h_9, h_{14}	–0.982666E-01
h_{10}, h_{13}	0.116089E+00
h_{11}, h_{12}	0.473145E+00

3.3 Adaptive quantizer

The difference signals, $e_L(n)$ and $e_H(n)$, are quantized to 6 and 2 bits for the lower and higher sub-bands respectively. Tables 5 and 6 give the decision levels and the output codes for the 6- and 2-bit quantizers respectively. In these tables, only the positive decision levels are indicated, the negative levels can be determined by symmetry. m_L *and* m_H are indices for the quantizer intervals. The interval boundaries, *LL*6, *LU*6, *HL* and *HU,* are scaled by computed scale factors, $\Delta_L(n)$ and $\Delta_H(n)$ (see clause 3.5). Indices, m_L and m_H, are then determined to satisfy the following:

$$LL6(m_L) \cdot \Delta_L(n) \le e_L(n) < LU6(m_L) \cdot \Delta_L(n) \qquad (3\text{-}7)$$

$$HL(m_H) \cdot \Delta_H(n) \le e_H(n) < HU(m_H) \cdot \Delta_H(n) \tag{3-8}$$

for the lower and higher sub-bands respectively.

The output codes, *ILN and IHN,* represent negative intervals, whilst the output codes, *ILP and IHP,* represent positive intervals. The output codes, $I_L(n)$ and $I_H(n)$, are then given by:

$$I_L(n) = \begin{cases} ILP(m_L), & \text{if } e_L(n) \ge 0 \\ ILN(m_L), & \text{if } e_L(n) < 0 \end{cases} \tag{3-9}$$

$$I_H(n) = \begin{cases} IHP(m_H), & \text{if } e_H(n) \ge 0 \\ IHN(m_H), & \text{if } e_H(n) < 0 \end{cases} \tag{3-10}$$

for the lower and higher sub-bands respectively.

Table 5 – Decision levels and output codes for the 6-bit lower sub-band quantizer

m_L	LL6	LU6	ILN	ILP
1	0.00000	0.06817	111111	111101
2	0.06817	0.14103	111110	111100
3	0.14103	0.21389	011111	111011
4	0.21389	0.29212	011110	111010
5	0.29212	0.37035	011101	111001
6	0.37035	0.45482	011100	111000
7	0.45482	0.53929	011011	110111
8	0.53929	0.63107	011010	110110
9	0.63107	0.72286	011001	110101
10	0.72286	0.82335	011000	110100
11	0.82335	0.92383	010111	110011
12	0.92383	1.03485	010110	110010
13	1.03485	1.14587	010101	110001
14	1.14587	1.26989	010100	110000
15	1.26989	1.39391	010011	101111
16	1.39391	1.53439	010010	101110
17	1.53439	1.67486	010001	101101
18	1.67486	1.83683	010000	101100
19	1.83683	1.99880	001111	101011
20	1.99880	2.19006	001110	101010
21	2.19006	2.38131	001101	101001
22	2.38131	2.61482	001100	101000
23	2.61482	2.84833	001011	100111
24	2.84833	3.14822	001010	100110
25	3.14822	3.44811	001001	100101
26	3.44811	3.86796	001000	100100
27	3.86796	4.28782	000111	100011
28	4.28782	4.99498	000110	100010
29	4.99498	5.70214	000101	100001
30	5.70214	∞	000100	100000

NOTE – If a transmitted codeword for the lower sub-band signal has been transformed, due to transmission errors to one of the four suppressed codewords "0000XX", the received codeword is set at "111111".

Table 6 – Decision levels and output codes for the 2-bit higher sub-band quantizer

m_H	HL	HH	IHN	IHP
1	0	1.10156	01	11
2	1.10156	∞	00	10

3.4 Inverse adaptive quantizers

3.4.1 Inverse adaptive quantizer in the lower sub-band ADPCM encoder

The lower sub-band output code, $I_L(n)$, is truncated by two bits to produce $I_{Lt}(n)$. The 4-bit code-word, $I_{Lt}(n)$, is converted to the truncated quantized difference signal, $d_{Lt}(n)$, using the $QL4^{-1}$ output values of Table 7, and scaled by the scale factor, $\Delta_L(n)$:

$$d_{Lt}(n) = QL4^{-1}\left[I_{Lt}(n)\right] \cdot \Delta_L(n) \cdot \mathrm{sgn}\left[I_{Lt}(n)\right] \tag{3-11}$$

where sgn $[I_{Lt}(n)]$ is derived from the sign of $e_L(n)$ defined in Equation 3-9.

There is a unique mapping, shown in Table 7, between four adjacent 6-bit quantizer intervals and the $QL4^{-1}$ output values. $QL4^{-1}[I_{Lt}(n)]$ is determined in two steps: first determination of the quantizer interval index, m_L, corresponding to $I_L(n)$ from Table 5, and then determination of $QL4^{-1}(m_L)$ by reference to Table 7.

Table 7 – Output values and multipliers for 6, 5 and 4-bit lower sub-band inverse quantizers

m_L	$QL6^{-1}$	$QL5^{-1}$	$QL4^{-1}$	W_L
1	0.03409	0.06817	0.0000	–0.02930
2	0.10460			
3	0.17746	0.21389		
4	0.25300		0.29212	–0.01465
5	0.33124	0.37035		
6	0.41259			
7	0.49706	0.53929		
8	0.58518		0.63107	0.02832
9	0.67697	0.72286		
10	0.77310			
11	0.87359	0.92383		
12	0.97934		1.03485	0.08398
13	1.09036	1.14587		
14	1.20788			
15	1.33191	1.39391		
16	1.46415		1.53439	0.16309
17	1.60462	1.67486		
18	1.75585			
19	1.91782	1.99880		
20	2.09443		2.19006	0.26270
21	2.28568	2.38131		
22	2.49806			

Table 7 – Output values and multipliers for 6, 5 and 4-bit lower sub-band inverse quantizers

m_L	$QL6^{-1}$	$QL5^{-1}$	$QL4^{-1}$	W_L
23	2.73157	2.84833		
24	2.99827		3.14822	0.58496
25	3.29816	3.44811		
26	3.65804			
27	4.07789	4.28782		
28	4.64140		4.99498	1.48535
29	5.34856	5.70214		
30	6.05572			

3.4.2 Inverse adaptive quantizer in the higher sub-band ADPCM encoder

The higher sub-band output code, $I_H(n)$ is converted to the quantized difference signal, $d_H(n)$, using the $Q2^{-1}$ output values of Table 8 and scaled by the scale factor, $\Delta_H(n)$:

$$d_H(n) = Q2^{-1}\left[I_H(n)\right] \cdot \Delta_H(n) \cdot \mathrm{sgn}\left[I_H(n)\right] \tag{3-12}$$

where $\mathrm{sgn}[I_H(n)]$ is derived from the sign of $e_H(n)$ defined in Equation (3-10), and where $Q2^{-1}[I_H(n)]$ is determined in two steps: first determine the quantizer interval index, m_H, corresponding to $I_H(n)$ from Table 6 and then determine $Q2^{-1}(m_H)$ by reference to Table 8.

Table 8 – Output values and multipliers for the 2-bit higher sub-band quantizer

m_H	$Q2^{-1}$	W_H
1	0.39453	–0.10449
2	1.80859	0.38965

3.5 Quantizer adaptation

This block defines $\Delta_L(n)$ and $\Delta_H(n)$, the scaling factors for the lower and higher sub-band quantizers. The scaling factors are updated in the log domain and subsequently converted to a linear representation. For the lower sub-band, the input is $I_{Lt}(n)$, the codeword truncated to preserve the four most significant bits. For the higher sub-band, the 2-bit quantizer output, $I_H(n)$, is used directly.

Firstly the log scaling factors, $\Delta_L(n)$ and $\Delta_H(n)$, are updated as follows:

$$\nabla_L(n) = \beta \cdot \nabla_L(n-1) + W_L\left[I_{Lt}(n-1)\right] \tag{3-13}$$

$$\nabla_H(n) = \beta \cdot \nabla_H(n-1) + W_H\left[I_H(n-1)\right] \tag{3-14}$$

where W_L and W_H are logarithmic scaling factors multipliers given in Tables 7 and 8, and B is a leakage constant equal to 127/128.

Then the log scaling factors are limited, according to:

$$0 \le \nabla_L(n) \le 9 \tag{3-15}$$

$$0 \le \nabla_H(n) \le 11 \tag{3-16}$$

Finally, the linear scaling factors are computed from the log scaling factors, using an approximation of the inverse $\log_2$ function:

$$\Delta_L(n) = 2^{[\nabla_L(n)+2]} \cdot \Delta_{\min} \tag{3-17}$$

$$\Delta_H(n) = 2^{\nabla_H(n)} \cdot \Delta_{\min} \tag{3-18}$$

where Δ_{min} is equal to half the quantizer step size of the 14 bit analogue-to-digital converter.

3.6 Adaptive prediction

3.6.1 Predicted value computations

The adaptive predictors compute predicted signal values, $s_L(n)$ and $s_H(n)$, for the lower and higher sub-bands respectively.

Each adaptive predictor comprises two sections: a second-order section that models poles, and a sixth-order section that models zeroes in the input signal.

The second order pole sections (coefficients $a_{L,i}$ and $a_{H,i}$) use the quantized reconstructed signals, $r_{Lt}(n)$ and $r_H(n)$, for prediction. The sixth order zero sections (coefficients $b_{L,i}$) and $b_{H,i}$) use the quantized difference signals, $d_{Lt}(n)$ and $d_H(n)$. The zero-based predicted signals, $s_{Lz}(n)$ *and* $s_{Hz}(n)$, are also employed to compute partially reconstructed signals as described in clause 3.6.2.

Firstly, the outputs of the pole sections are computed as follows:

$$s_{Lp} = \sum_{i=1}^{2} a_{L,i}(n-1) \cdot r_{Lt}(n-i) \tag{3-19}$$

$$s_{Hp} = \sum_{i=1}^{2} a_{H,i}(n-1) \cdot r_H(n-i) \tag{3-20}$$

Similarly, the outputs of the zero sections are computed as follows:

$$s_{Lz}(n) = \sum_{i=1}^{6} b_{L,i}(n-1) \cdot d_{Lt}(n-i) \tag{3-21}$$

$$s_{Hz}(n) = \sum_{i=1}^{6} b_{H,i}(n-1) \cdot d_H(n-i) \tag{3-22}$$

Then, the intermediate predicted values are summed to produce the predicted signal values:

$$s_L(n) = s_{Lp} + s_{Lz}(n) \tag{3-23}$$

$$s_H(n) = s_{Hp} + s_{Hz}(n) \tag{3-24}$$

3.6.2 Reconstructed signal computation

The quantized reconstructed signals, $r_{Lt}(n)$ and $r_H(n)$, are computed as follows:

$$r_{Lt}(n) = s_L + d_{Lt}(n) \tag{3-25}$$

$$r_H(n) = s_H + d_H(n) \tag{3-26}$$

The partially reconstructed signals, $p_{Lt}(n)$ and $p_H(n)$, used for the pole section adaptation, are then computed:

$$p_{Lt}(n) = d_{Lt}(n) + s_{Lz}(n) \tag{3-27}$$

$$p_H(n) = d_H(n) + s_{Hz}(n) \tag{3-28}$$

3.6.3 Pole section adaptation

The second order pole section is adapted by updating the coefficients, $a_{L,1}$, $a_{L,2}$, $a_{H,1}$, $a_{H,2}$, using a simplified gradient algorithm:

$$a_{L,1}(n) = (1 - 2^{-8})a_{L,1}(n-1) + 3 \cdot 2^{-8} \cdot p_A \tag{3-29}$$

$$a_{L,2}(n) = (1 - 2^{-7})a_{L,2}(n-1) + 2^{-7} \cdot p_B - 2^{-7} \cdot f \cdot p_A \tag{3-30}$$

where

$$p_A = \text{sgn2}\left[p_{Lt}(n)\right] \cdot \text{sgn2}\left[p_{Lt}(n-1)\right] \tag{3-31}$$

$$p_B = \text{sgn2}\left[p_{Lt}(n)\right] \cdot \text{sgn2}\left[p_{Lt}(n-2)\right] \tag{3-32}$$

with

$$\text{sgn2}(q) = \begin{cases} +1, & q \geq 0 \\ -1, & q < 0 \end{cases} \tag{3-33}$$

and

$$f = \begin{cases} 4a_{L,1}(n-1), & |a_{L,1}| \leq 1/2 \\ 2\,\text{sgn}\left[a_{L,1}(n-1)\right], & |a_{L,1}| > 1/2 \end{cases} \tag{3-34}$$

Then the following stability constraints are imposed:

$$|a_{L,2}| \leq 0{,}75 \tag{3-35}$$

$$|a_{L,1}| \leq 1 - 2^{-4} - a_{L,2} \tag{3-36}$$

$a_{H,1}(n)$ and $a_{H,2}(n)$ are similarly computed, replacing $a_{L,1}(n)$, $a_{L,2}(n)$ and $P_{Lt}(n)$, by $a_{H,1}(n)$, $a_{H,2}(n)$ and $P_H(n)$, respectively.

3.6.4 Zero section adaptation

The sixth order zero predictor is adapted by updating the coefficients $b_{L,i}$ and $b_{H,i}$ using a simplified gradient algorithm:

$$b_{L,i}(n) = (1 - 2^{-8})b_{L,i}(n-1) + 2^{-7}\,\text{sgn}3\left[d_{Lt}(n)\right] \cdot \text{sgn}2\left[d_{Lt}(n-i)\right] \tag{3-37}$$

for i = 1, 2 . . . 6

and with

$$\text{sgn3}(q) = \begin{cases} +1, & q > 0 \\ 0, & q = 0 \\ -1, & q < 0 \end{cases} \tag{3-38}$$

where $b_{L,i}(n)$ is implicitly limited to ± 2.

$b_{H,i}(n)$ are similarly updated, replacing $b_{L,i}(n)$ and $d_{Lt}(n)$ by $b_{H,i}(n)$ and $d_H(n)$ respectively.

4 SB-ADPCM decoder principles

A block diagram of the SB-ADPCM decoder is given in Figure 6 and block diagrams of the lower and higher sub-band ADPCM decoders are given respectively in Figures 7 and 8.

The input to the lower sub-band ADPCM decoder, $I_{L,r}$, may differ from I_L even in the absence of transmission errors, in that one or two least significant bits may have been replaced by data.

4.1 Inverse adaptive quantizer

4.1.1 Inverse adaptive quantizer selection for the lower sub-band ADPCM decoder

According to the received indication of the mode of operation the number of least significant bits which should be truncated from the input codeword I_{Lr}, and the choice of the inverse adaptive quantizer are determined, as shown in Table 2.

For operation in mode 1, the 6-bit codeword, $I_{Lr}(n)$, is converted to the quantized difference, $d_L(n)$, according to $QL6^{-1}$ output values of Table 7, and scaled by the scale factor, $\Delta_L(n)$*:*

$$d_L(n) = QL6^{-1}\left[I_{Lr}(n)\right] \cdot \Delta_L(n) \cdot \mathrm{sgn}\left[I_{Lr}(n)\right] \tag{4-1}$$

where sgn[$I_{Lr}(n)$] is derived from the sign of $I_L(n)$ defined in equation (3-9).

Similarly, for operations in mode 2 or mode 3, the truncated codeword (by one or two bits) is converted to the quantized difference signal, $d_L(n)$, according to $QL5^{-1}$ or $QL4^{-1}$ output values of Table 7 respectively.

There are unique mappings, shown in Table 7, between two or four adjacent 6-bit quantizer intervals and the $QL5^{-1}$ or $QL4^{-1}$ output values respectively.

In the computations above, the output values are determined in two steps: first determination of the quantizer interval index, m_L, corresponding to $I_{Lr}(n)$ from Table 5, and then determination of the output values corresponding to m_L by reference to Table 7.

The inverse adaptive quantizer, used for the computation of the predicted value and for adaptation of the quantizer and predictor, is described in clause 3.4.1, but with $I_L(n)$ replaced by $I_{Lr}(n)$.

4.1.2 Inverse adaptive quantizer for the higher sub-band ADPCM decoder

See clause 3.4.2.

4.2 Quantizer adaptation

See clause 3.5.

4.3 Adaptive prediction

4.3.1 Predicted value computation

See clause 3.6.1.

4.3.2 Reconstructed signal computation

See clause 3.6.2.

The output reconstructed signal for the lower sub-band ADPCM decoder, $r_L(n)$*,* is computed from the quantized difference signal, $d_L(n)$*,* as follows:

$$r_L(n) = s_L(n) + d_L(n) \tag{4-2}$$

4.3.3 Pole section adaptation

See clause 3.6.3.

4.3.4 Zero section adaptation

See clause 3.6.4.

4.4 Receive QMF

A 24-coefficient QMF is used to reconstruct the output signal, $x_{out}(j)$, from the reconstructed lower and higher sub-band signals, $r_L(n)$ and $r_H(n)$. The QMF coefficient values, h_i, are the same as those used in the transmit QMF and are given in Table 4.

The output signals, $x_{out}(j)$ and $x_{out}(j + 1)$, are computed in the following way:

$$x_{out}(j) = 2\sum_{i=0}^{11} h_{2i} \cdot x_d(i) \tag{4-3}$$

$$x_{out}(j+1) = 2\sum_{i=0}^{11} h_{2i+1} \cdot x_s(i) \tag{4-4}$$

where

$$x_d(i) = r_L(n-i) - r_H(n-i) \tag{4-5}$$

$$x_s(i) = r_L(n-i) + r_H(n-i) \tag{4-6}$$

5 Computational details for QMF

5.1 Input and output signals

Table 9 defines the input and output signals for the transmit and receive QMF. All input and output signals have 16-bit word lengths, which are limited to a range of –16384 to 16383 in 2's complement notation. Note that the most significant magnitude bit of the A/D output and the D/A input appears at the third bit location in XIN and XOUT, respectively.

Table 9 – Representation of input and output signals

Transmit QMF			
	Name	**Binary representation**	**Description**
Input	XIN	S, S, –2, –3, . . ., –14, –15	Input value (uniformly quantized)
Output	XL	S, S, –2, –3, . . ., –14, –15	Output signal for lower sub-band encoder
Output	XH	S, S, –2, –3, . . ., –14, –15	Outband signal for higher sub-band encoder
Receive QMF			
	Name	**Binary representation**	**Description**
Input	RL	S, S, –2, –3, . . ., –14, –15	Lower sub-band reconstructed signal
Input	RH	S, S, –2, –3, . . ., –14, –15	Higher sub-band reconstructed signal
Output	XOUT	S, S, –2, –3, . . ., –14, –15	Output value (uniformly quantized)

NOTE – XIN and XOUT are represented in a sign-extended 15-bit format, where the LSB is set to "0" for 14-bit converters.

5.2 Description of variables and detailed specification of sub-blocks

This section contains a detailed expansion of the transmit and receive QMF. The expansions are illustrated in Figures 17 and 18 with the internal variables given in Table 10, and the QMF coefficients given in Table 11. The word lengths of internal variables, XA, XB and WD must be equal to or greater than 24 bits (see Note). The other internal variables have a minimum of 16 bit word lengths. A brief functional description and the full specification is given for each sub-block.

The notations used in the block descriptions are as follows:

>> n denotes an *n*-bit arithmetic shift right operation (sign extension),

+ denotes arithmetic addition with saturation control which forces the result to the minimum or maximum representable value in case of underflow or overflow, respectively,

– denotes arithmetic subtraction with saturation control which forces the result to the minimum or maximum representable value in case of underflow or overflow, respectively.

* denotes arithmetic multiplication which can be performed with either truncation or rounding,

< denotes the "less than" condition as $x < y$; x is less than y,

> denotes the "greater than" condition, as $x > y$; x is greater than y,

= denotes the substitution of the right-hand variable for the left-hand variable.

NOTE 1 – Some freedom is offered for the implementation of the accumulation process in the QMF: the word lengths of the internal variables can be equal to or greater than 24 bits, and the arithmetic multiplications can be performed with either truncation or rounding. It allows a simplified implementation on various types of processors. The counterpart is that it excludes the use of digital test sequence for the test of the QMF.

Table 10 – Representation of internal processing variables and QMF coefficients

Transmit QMF		
Name	**Binary representation**	**Description**
XA	S, –1, –2, –3, . . ., –y+1, –y	Output signal of sub-block, ACCUMA
XB	S, –1, –2, –3, . . ., –y+1, –y	Output signal of sub-block, ACCUMB
XIN1, XIN2, . . ., XIN23	S, S, –2, –3, . . ., –14, –15	Input signal with delays 1 to 23
Receive QMF		
Name	**Binary representation**	**Description**
XD, XD1, . . ., XD11	S, –1, –2, –3, . . ., –14, –15	Input signal for sub-block, ACCUMC, with delays 0 to 11
XOUT1	S, S, –2, –3, . . ., –14, –15	8 kHz sampled output value
XOUT2	S, S, –2, –3, . . ., –14, –15	8 kHz sampled output value
XS, XS1, . . ., XS11	S, –1, –2, –3, . . ., –14, –15	Input signal for sub-block, ACCUMD, with delays 0 to 11
WD	S, –1, –2, –3, . . ., –y+1, –y	Partial sum
QMF coefficient		
Name	**Binary representation**	**Description**
H0, H1, . . ., H23	S, –2, –3, –4, . . ., –12, –13	Filter coefficient values
NOTE – y is equal to or greater than 23.		

Table 11 – QMF coefficient

Coefficient	Scaled values (see Note)
H0, H23	3
H1, H22	–11
H2, H21	–11
H3, H20	53
H4, H19	12
H5, H18	–156
H6, H17	32
H7, H16	362
H8, H15	–210
H9, H14	–805
H10, H13	951
H11, H12	3876
NOTE – QMF coefficients are scaled by 2^{13} with respect to the representation specified in Table 10.	

5.2.1 Description of the transmit QMF

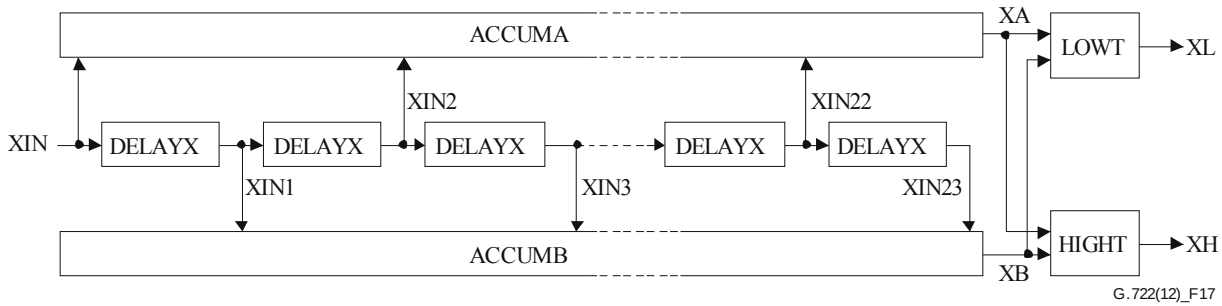


Figure 17 – Transmit QMF

DELAYX

Input: x

Output: y

NOTE – Index (j) indicates the current 16-kHz sample period, while index (j – 1) indicates the previous one.

Function: Memory block. For any input x, the output is given by:

$y(j) = x(j - 1)$

ACCUMA

Inputs: XIN, XIN2, XIN4, ..., XIN22

Output: XA

NOTE 1 – H0, H2, ..., H22 are obtained from Table 11.

NOTE 2 – The values XIN, XIN2, ..., XIN22 and H0, H2, ..., H22 may be shifted before multiplication, if so desired. The result XA must be rescaled accordingly, In performing these scaling operations the following rules must be obeyed:

1) the precision of XIN, XIN2, ..., XIN22 and H0, H2, ..., H22 as given in Table 9 and Table 10 must be retained,

2) the partial products and the output signal XA must be retained to a significance of at least 2^{-23},

3) no saturation should occur in the calculation of the function XA.

NOTE 3 – No order of summation is specified in accumulating the partial products.

Function: Multiply the even order QMF coefficients by the appropriately delayed input signals, and accumulate these products.

XA = (XIN ∗ H0) + (XIN2 ∗ H2) + (XIN4 ∗ H4) + ... + (XIN22 ∗ H22)

ACCUMB

Inputs: XIN1, XIN3, XIN5,. . . , XIN23

Output: XB

NOTE 1 – H1, H3, . . ., H23 are obtained from Table 11.

NOTE 2 – The values XIN1, XIN3, ..., XIN23 and HI, H3, ..., H23 may be shifted before multiplication, if so desired. The result XB must be rescaled accordingly. In performing these scaling operations the following rules must be obeyed:

1) the precision of XIN1, XIN3, ..., XIN23 and HI, H3, ..., H23 as given in Table 9 and Table 10 must be retained,

2) the partial products and the output signal X3 must be retained to a significance of at least 2^{-23},

3) no saturation should occur in the calculation of the function XB.

NOTE 3 – No order of summation is specified in accumulating the partial products.

Function: Multiply the odd order QMF coefficients by the appropriately delayed input signals, and accumulate these products.

XB = (XIN1 ∗ H1) + (XIN3 ∗ H3) + (XIN5 ∗ H5) + ... + (XIN23 ∗ H23)

LOWT

Inputs: XA, XB

Output: XL

Function: Compute the lower sub-band signal component.

XL = (XA + XB) >> (y – 15)

$$XL = \begin{cases} 16383, & \text{if XL} \\ -16384, & \text{if XI} \end{cases}$$

HIGHT

Inputs: XA, XB

Output: XH

Function Compute the higher sub-band signal component.

XH = (XA – XB) >> (y – 15)

$$XH = \begin{cases} 16383, & \text{if} \\ -16384, & \text{if} \end{cases}$$

5.2.2 Description of the receive QMF

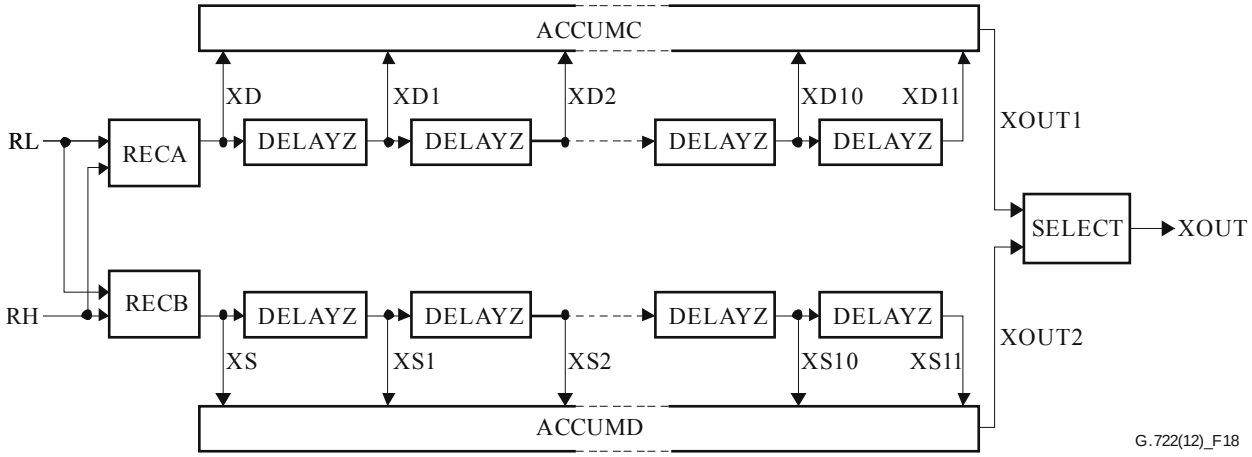


Figure 18 – Receive QMF

RECA

Inputs: RL, RH

Output: XD

Function: Compute the input signal to the receive QMF

XD = RL – RH

RECB

Inputs: RL, RH

Output: XS

Function: Compute the input signal to the receive QMF

XS = RL + RH

DELAYZ

Input: x

Output: y

NOTE – Index (n) indicates the current 8-kHz sample period, while index (n – 1) indicates the previous one.

Function: Memory block. For any input x, the output is given by:

$y(n) = x(n-1)$

ACCUMC

Inputs: XD, XDi (i = 1 to 11)

Output: XOUT1

NOTE 1 – H0, H2, ..., H22 are obtained from Table 11.

NOTE 2 – The values XD, XD1, ..., XD11 and H0, H2, ..., H22 may be shifted before multiplication, if so desired. The result WD must be rescaled accordingly. In performing these scaling operations the following rules must be obeyed:

1) the precision of XD, XD1, ..., XD11 and H0, H2, ..., H22 as given in Table 9 and Table 10 must be retained,

2) the partial products and the output signal WD must be retained to a significance of at least 2^{-23};

3) no saturation should occur in the calculation of the function WD.

NOTE 3 – No order of summation is specified in accumulating the partial products.

Function: Multiply the even order QMF coefficients by the appropriately delayed input signals, and accumulate these products.

$WD = (XD * H0) + (XD1 * H2) + (XD2 * H4) + ... + (XD11 * H22)$

$XOUT1 = WD >> (y - 16)$

$$XOUT1 = \begin{cases} 16383, \\ -16384, \end{cases}$$

ACCUMD

Inputs: XS, XSi (i = 1 to 11)

Output: XOUT2

NOTE 1 – H1, H3,..., H23 are obtained from Table 11.

NOTE 2 – The values XS, XS1, ..., XS11 and H1, H3, ..., H23 may be shifted before multiplication, if so desired. The result WD must be rescaled accordingly. In performing these scaling operations the following rules must be obeyed:

1) the precision of XS, XS1, ..., XS11 and H1, H3, ..., H23 as given in Table 9 and Table 10 must be retained,

2) the partial products and the output signal WD must be retained to a significance of at least 2^{-23};

3) no saturation should occur in the calculation of the function WD.

NOTE 3 – No order of summation is specified in accumulating the partial products.

Function: Multiply the odd order QMF coefficients by the appropriately delayed input signals, and accumulate these products.

WD = (XS ∗ H1) + (XS1 ∗ H3) + (XS2 ∗ H5) + ... + (XS11 ∗ H23)

XOUT2 = WD >> (y – 16)

$$\text{XOUT2} = \begin{cases} 16383, \\ -16384, \end{cases}$$

SELECT

Inputs: XOUT1, XOUT2

Output: XOUT

NOTE 1 – Index (j) indicates the current 16-kHz sample period, while index (j + 1) indicates the next one. With respect to the input sampling instant XOUT1 is selected first, followed by XOUT2.

Function: Select one of the 8 kHz sampled input signals alternately to produce the 16 kHz sampled output signal.

XOUT(j) = XOUT1

XOUT(j + 1) = XOUT2

---

6 Computational details for lower and higher sub-band ADPCM

6.1 Input and output signals

Table 12 defines the input and output signals for the lower and higher sub-band encoders and decoders. The signal RS represents a reset function that sets all internal memory elements to a specified condition, so that encoders or decoders can be forced into a known state. The signal MODE represents a mode indication. The three basic modes of operation are described in Table 1. The mode identification is performed in every 8 kHz sampling interval.

Table 12 – Input and output signals

Lower sub-band encoder		
	Name	**Description**
Input	XL	15-bit uniformly quantized input signal
Input	RS	Reset
Output	IL	6-bit ADPCM codeword
Higher sub-band encoder		
	Name	**Description**
Input	XH	15-bit uniformly quantized input signal
Input	RS	Reset
Output	IH	2-bit ADPCM codeword

Table 12 – Input and output signals

Lower sub-band decoder		
	Name	**Description**
Input	ILR	Received 6-bit ADPCM codeword
Input	MODE	Mode indication
Input	RS	Reset
Output	RL	15-bit uniformly quantized output signal
Higher sub-band decoder		
	Name	**Description**
Input	IH	2-bit ADPCM codeword
Input	RS	Reset
Output	RH	15-bit uniformly quantized output signal

6.2 Description of variables and detailed specification of sub-blocks

This section contains a detailed expansion of all blocks in Figures 4, 5, 7 and 8 described in clauses 3 and 4. The expansions are illustrated in Figures 19 to 31 with the internal processing variables in Table 13, the constant values in Tables 14 and 15, and conversion tables in Tables 16 to 21. All internal variables have 16-bit word lengths, and are represented in 2's complement notation. Constant values with 13-bit precision as given in Tables 14 and 15 are used in sub-blocks with 16-bit representation, extending the sign to the first three MSBs. A brief functional description and full specification is given for each sub-block.

The notations used in the block descriptions are as follows:

$<< n$ denotes an n-bit arithmetic shift left operation (zero fill);

$>> n$ denotes an n-bit arithmetic shift right operation (sign extension); if n is negative, $>> n$ means $<< (-n)$;

$>>> n$ denotes an n-bit logical shift right operation (zero fill);

$<<< n$ denotes an n-bit logical shift left operation (zero fill);

& denotes the logical "and" operation;

+ denotes arithmetic addition. (The result is set at +32767 when overflow occurs, or at −32768 when underflow occurs.);

– denotes arithmetic subtraction. (The result is set at +32767 when overflow occurs, or at −32768 when underflow occurs.).

∗ denotes the multiplication defined by the following arithmetic operation:

A ∗ B = (A times B) >> 15;

= = denotes the "equal to" condition;

! = denotes the "not equal to" condition;

< denotes the "less than" condition, as $x < y$; x is less than y;

> denotes the "greater than" condition, as $x > y$; x is greater than y;

= denotes the substitution of right-hand variable for the left-hand variable;

| delineates comments to equations.

Table 13 – Internal processing variables

Lower sub-band ADPCM		
Name	**Binary representation**	**Description**
AL1*, AL2*	S, 0, –1, –2, ..., –13, –14	Delayed second-order pole section coefficients
APL1,APL2	S, 0, –1, –2, ..., –13, –14	Second-order pole section coefficient
BL1*,..., BL6*	S, 0, –1, –2, ..., –13, –14	Delayed sixth-order zero section coefficients
BPL1,..., BPL6	S, 0, –1, –2, ..., –13, –14	Sixth-order zero section coefficients
DEPL	S, –4, –5, –6, ..., –17, –18	Quantizer scale factor
DETL*	S, –4, –5, –6, ..., –17, –18	Delayed quantizer scale factor
DLT	S, –1, –2, –3, ..., –14, –15	Quantized difference signal for the adaptative predictor with delay 0
DLT1*,..., DLT6*	S, –1, –2, –3, ..., –14, –15	Quantized difference signal for the adaptative predictor with delays 1 to 6
DL	S, –1, –2, –3, ..., –14, –15	Quantized difference signal for decoder output
EL	S, –1, –2, –3, ..., –14, –15	Difference signal
NBL*	S, 3, 2, 1,0, ..., –10, –11	Delayed logarithmic quantizer scale factor
NBPL	S, 3, 2, 1,0, ..., –10, –11	Logarithmic quantizer scale factor
PLT	S, –1, –2, –3, ..., –14, –15	Partially reconstructed signal with delay 0
PLT1*, PLT2*	S, –1, –2, –3, ..., –14, –15	Partially reconstructed signal with delays 1 and 2
YL	S, –1, –2, –3, ..., –14, –15	Output reconstructed signal
RLT	S, –1, –2, –3, ..., –14, –15	Reconstructed signal for the adaptive predictor with delay 0
RLT1*, RLT2*	S, –1, –2, –3, ..., –14, –15	Reconstructed signal for the adaptive predictor with delay 1 and 2
SL	S, –1, –2, –3, ..., –14, –15	Predictor output value
SPL	S, –1, –2, –3, ..., –14, –15	Pole section output signal
SZL	S, –1, –2, –3, ..., –13, –14	Zero section output signal
AH1*, AH2*	S, 0, –1, –2, ..., –13, –14	Delayed second-order pole section coefficients
APH1, APH2	S, 0, –1, –2, ..., –13, –14	Second-order pole section coefficients
BH1*,..., BH6*	S, 0, –1, –2, ..., –13, –14	Delayed sixth-order zero section coefficients
BPH1,..., BPH6	S, 0, –1, –2, ..., –13, –14	Sixth-order zero section coefficients
DEPH	S, –4, –5, –6, ..., –17, –18	Quantizer scale factor
DETH*	S, –4, –5, –6, ..., –17, –18	Delayed quantizer scale factor
DH	S, –1, –2, –3, ..., –14, –15	Quantizer difference signal with delay 0
DH1*,..., DH6*	S, –1, –2, –3, ..., –14, –15	Quantized difference signal with delays 1 to 6
EH	S, –1, –2, –3, ..., –14, –15	Difference signal
NBH*	S, 3, 2, 1, 0, ..., –10, –11	Delayed logarithmic quantizer scale factor
NBPH	S, 3, 2, 1, 0, ..., –10, –11	Logarithmic quantizer scale factor
PH	S, –1, –2, –3, ..., –14, –15	Partially reconstructed signal with delay 0
PH1*, PH2*	S, –1, –2, –3, ..., –14, –15	Partially reconstructed signal with delays 1 and 2
YH	S, –1, –2, –3, ..., –14, –15	Quantized reconstructed signal with delay 0
RH1*, RH2*	S, –1, –2, –3, ..., –14, –15	Quantized reconstructed signal with delays 1 and 2
SH	S, –1, –2, –3, ..., –14, –15	Predictor output value
SPH	S, –1, –2, –3, ..., –14, –15	Pole section output signal
SZH	S, –1, –2, –3, ..., –14, –15	Zero section output signal
NOTE – * indicates variables which should be initialized to a specific value when a reset condition is applied.		

Table 14 – Quantizer decision levels and output values

Quantizer constant representation		
Name	**Binary representation**	**Description**
Qi	S, 2, 1, 0, –1, ..., –8, –9	Quantizer decision level
QQi	S, 2, 1, 0, –1, ..., –8, –9	Inverse quantizer output
WL, WH	S, 0, –1, –2, ..., –10, –11	Logarithmic scaling factor multiplier

Lower sub-band quantizer					
Address	**Q6**	**QQ6**	**QQ5**	**QQ4**	**WL**
0					
1	35	17	35		
2	72	54	110	0	–60
3	110	91	190	150	–30
4	150	130	276	323	58
5	190	170	370	530	172
6	233	211	473	786	334
7	276	254	587	1121	538
8	323	300	714	1612	1198
9	370	347	858	2557	3042
10	422	396	1023		
11	473	447	1219		
12	530	501	1458		
13	587	558	1765		
14	650	618	2195		
15	714	682	2919		
16	786	750			
17	858	822			
18	940	899			
19	1023	982			
20	1121	1072			
21	1219	1170			
22	1339	1279			
23	1458	1399			
24	1612	1535			
25	1765	1689			
26	1980	1873			
27	2195	2088			
28	2557	2376			
29	2919	2738			
30		3101			

Higher sub-band quantizer			
Address	**Q2**	**QQ2**	**WH**
1	564	202	–214
2		926	798

Table 15 – Log-to-linear conversion table

Conversion table constants		
Name	**Binary representation**	**Description**
ILA	S, –5, –6, –7, ..., –15, –16	353-entry table constants
ILB	S, 0, –1, –2, ..., –15, –16	32-entry table constants

ILA

i \ j	0	1	2	3	4	5	6	7
0	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1
16	1	1	1	2	2	2	2	2
24	2	2	2	2	2	2	2	2
32	3	3	3	3	3	3	3	3
40	3	3	3	4	4	4	4	4
48	4	4	4	5	5	5	5	5
56	5	5	6	6	6	6	6	6
64	7	7	7	7	7	7	8	8
72	8	8	8	9	9	9	9	10
80	10	10	10	11	11	11	11	12
88	12	12	13	13	13	13	14	14
96	15	15	15	16	16	16	17	17
104	18	18	18	19	19	20	20	21
112	21	22	22	23	23	24	24	25
120	25	26	27	27	28	28	29	30
128	31	31	32	33	33	34	35	36
136	37	37	38	39	40	41	42	43
144	44	45	46	47	48	49	50	51
152	52	54	55	56	57	58	60	61
160	63	64	65	67	68	70	71	73
168	75	76	78	80	82	83	85	87
176	89	91	93	95	97	99	102	104
184	106	109	111	113	116	118	121	124
192	127	129	132	135	138	141	144	147
200	151	154	157	161	165	168	172	176
208	180	184	188	192	196	200	205	209
216	214	219	223	228	233	238	244	249
224	255	260	266	272	278	284	290	296
232	303	310	316	323	331	338	345	353
240	361	369	377	385	393	402	411	420
248	429	439	448	458	468	478	489	500
256	511	522	533	545	557	569	582	594
264	607	621	634	648	663	677	692	707
272	723	739	755	771	788	806	823	841
280	860	879	898	918	938	958	979	1001
288	1023	1045	1068	1092	1115	1140	1165	1190
296	1216	1243	1270	1298	1327	1356	1386	1416
304	1447	1479	1511	1544	1578	1613	1648	1684
312	1721	1759	1797	1837	1877	1918	1960	2003
320	2047	2092	2138	2185	2232	2281	2331	2382
328	2434	2488	2542	2598	2655	2713	2773	2833
336	2895	2959	3024	3090	3157	3227	3297	3370
344	3443	3519	3596	3675	3755	3837	3921	4007
352	4095							

Table 15 – Log-to-linear conversion table

ILB								
j i	0	1	2	3	4	5	6	7
0	2048	2093	2139	2186	2233	2282	2332	2383
8	2435	2489	2543	2599	2656	2714	2774	2834
16	2896	2960	3025	3091	3158	3228	3298	3371
24	3444	3520	3597	3676	3756	3838	3922	4008

NOTE 1 – A table address is obtained by adding i and j.

NOTE 2 – Either a 353-entry or a 32-entry table may be used in accordance with the choice of log-to-linear conversion method, Method 1 or Method 2 (see clauses 6.2.1.3 and 6.2.2.3).

Table 16 – Conversion from quantizer intervals to 6-bit output codewords

SIL	MIL	IL
–1	30	000100
–1	29	000101
–1	28	000110
–1	27	000111
–1	26	001000
–1	25	001001
–1	24	001010
–1	23	001011
–1	22	001100
–1	21	001101
–1	20	001110
–1	19	001111
–1	18	010000
–1	17	010001
–1	16	010010
–1	15	010011
–1	14	010100
–1	13	010101
–1	12	010110
–1	11	010111
–1	10	011000
–1	9	011001
–1	8	011010
–1	7	011011
–1	6	011100
–1	5	011101
–1	4	011110
–1	3	011111
–1	2	111110
–1	1	111111

SIL	MIL	IL
0	1	111101
0	2	111100
0	3	111011
0	4	111010
0	5	111001
0	6	111000
0	7	110111
0	8	110110
0	9	110101
0	10	110100
0	11	110011
0	12	110010
0	13	110001
0	14	110000
0	15	101111
0	16	101110
0	17	101101
0	18	101100
0	19	101011
0	20	101010
0	21	101001
0	22	101000
0	23	100111
0	24	100110
0	25	100101
0	26	100100
0	27	100011
0	28	100010
0	29	100001
0	30	100000

Table 17 – Conversion from 4-bit codewords to quantizer intervals

RIL	SIL	IL4
0000	0	0
0001	–1	7
0010	–1	6
0011	–1	5
0100	–1	4
0101	–1	3
0110	–1	2
0111	–1	1
1111	0	0
1110	0	1
1101	0	2
1100	0	3
1011	0	4
1010	0	5
1001	0	6
1000	0	7

NOTE – It is possible for the decoder to receive the codeword 0000 due to transmission errors.

Table 18 – Conversion from 6-bit codewords to quantizer intervals

RIL	SIL	IL6	RIL	SIL	IL6
000000	–1	1	111110	–1	2
000001	–1	1	111111	–1	1
000010	–1	1	111101	0	1
000011	–1	1	111100	0	2
000100	–1	30	111011	0	3
000101	–1	29	111010	0	4
000110	–1	28	111001	0	5
000111	–1	27	111000	0	6
001000	–1	26	110111	0	7
001001	–1	25	110110	0	8
001010	–1	24	110101	0	9
001011	–1	23	110100	0	10
001100	–1	22	110011	0	11
001101	–1	21	110010	0	12
001110	–1	20	110001	0	13
001111	–1	19	110000	0	14
010000	–1	18	101111	0	15
010001	–1	17	101110	0	16
010010	–1	16	101101	0	17
010011	–1	15	101100	0	18
010100	–1	14	101011	0	19
010101	–1	13	101010	0	20
010110	–1	12	101001	0	21
010111	–1	11	101000	0	22
011000	–1	10	100111	0	23
011001	–1	9	100110	0	24
011010	–1	8	100101	0	25
011011	–1	7	100100	0	26
011100	–1	6	100011	0	27
011101	–1	5	100010	0	28
011110	–1	4	100001	0	29
011111	–1	3	100000	0	30

NOTE – It is possible for the decoder to receive the codewords 000000, 000001, 000010 and 000011 due to transmission errors.

Table 19 – Conversion from 5-bit codewords to quantizer intervals

RIL	SIL	IL5		RIL	SIL	IL5
000000	–1	1		11111	–1	1
000001	–1	1		11110	0	1
000010	–1	15		11101	0	2
000011	–1	14		11100	0	3
000100	–1	13		11011	0	4
000101	–1	12		11010	0	5
000110	–1	11		11001	0	6
000111	–1	10		11000	0	7
001000	–1	9		10111	0	8
001001	–1	8		10110	0	9
001010	–1	7		10101	0	10
001011	–1	6		10100	0	11
001100	–1	5		10011	0	12
001101	–1	4		10010	0	13
001110	–1	3		10001	0	14
001111	–1	2		10000	0	15

NOTE – It is possible for the decoder to receive the codewords 00000 and 00001 due to transmission errors.

Table 20 – Conversion from quantizer intervals to 2-bit output codewords

SIH	MIH	IH
–1	2	00
–1	1	01
0	1	11
0	2	10

Table 21 – Conversion from 2-bit codewords to quantizer intervals

IH	SIH	IH2
00	–1	2
01	–1	1
11	0	1
10	0	2

6.2.1 Description of the lower sub-band ADPCM

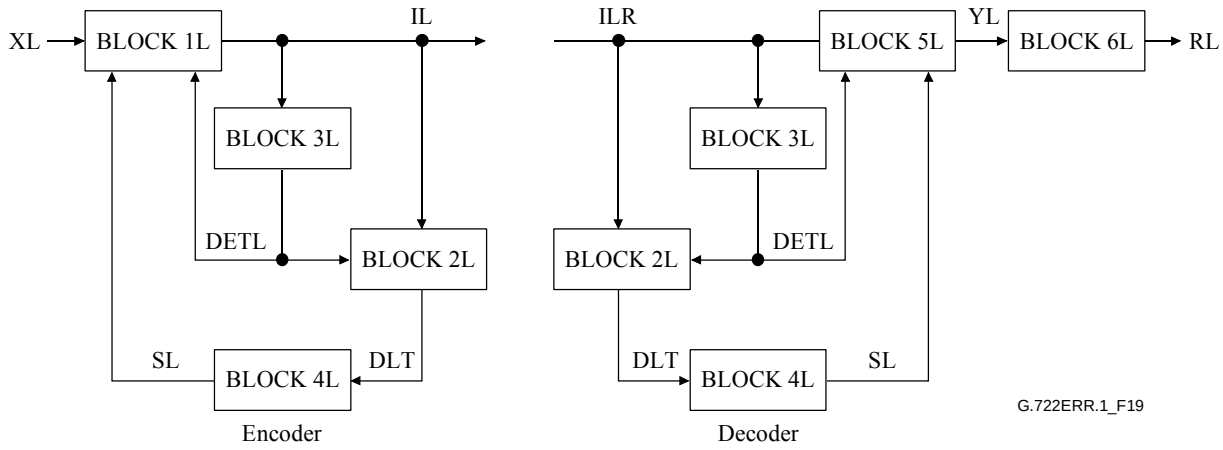


Figure 19 – Lower sub-band ADPCM encoder and decoder

6.2.1.1 Difference signal computation and quantization in the lower sub-band (BLOCK IL)

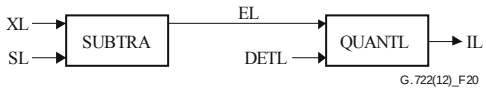


Figure 20 – Difference signal computation and quantization in the lower sub-band

---

SUBTRA

Inputs: XL, SL

Output: EL

Function: Compute the difference signal by subtracting from the input signal its predicted value.

EL = XL – SL

QUANTL

Inputs: EL, DETL

Output: IL

NOTE 1 – If WD falls exactly on a higher decision level, LDU, the larger adjacent MIL is used.

NOTE 2 – When both the lower and higher decision levels, LDL and LDU, are the same value, the value of MIL is excluded from that to be chosen.

Function: Quantize the difference signal in the lower sub-band.

SIL = EL >> 15 | Sign of EL

$$WD = \begin{cases} EL, & \text{if } SIL == 0 \\ 32767 - EL \;\&\; 32767, & \text{if } SIL == -1 \end{cases}$$

WD = EL, if SIL == 0 | Magnitude of EL

WD = 32767 – EL & 32767, if SIL == – 1 | (Magnitude of EL) – 1

Quantizer decision levels and corresponding MIL values:

WD		
Lower decision level (LDL)	**Higher decision level (LDU)**	**MIL**
0	(Q6 (1) << 3) * DETL	1
(Q6 (1) << 3) * DETL	(Q6 (2) << 3) * DETL	2
(Q6 (2) << 3) * DETL	(Q6 (3) << 3) * DETL	3
(Q6 (3) << 3) * DETL	(Q6 (4) << 3) * DETL	4
(Q6 (4) << 3) * DETL	(Q6 (5) << 3) * DETL	5
:	:	:
:	:	:
(Q6 (26) << 3) * DETL	(Q6 (27) << 3) * DETL	27
(Q6 (27) << 3) * DETL	(Q6 (28) << 3) * DETL	28
(Q6 (28) << 3) * DETL	(Q6 (29) << 3) * DETL	29
		30
otherwise		

Q6 is obtained from Table 14.

IL is obtained from Table 16 using SIL and MIL.

6.2.1.2 Inverse quantization of the difference signal in the lower sub-band (BLOCK 2L)

Figure 21 – Inverse quantization of the difference signal in the lower sub-band

---

INVQAL

Inputs: IL (ILR in the decoder), DETL

Output: DLT

Function: Compute the quantized difference signal for the adaptive predictor in the lower sub-band.

```
RIL = IL >>> 2                                  |  Delete
                                                |  2 LSB
SIL and IL4 are obtained from Table 17          |
using RIL. Use IL4 as an address for            |  Derive sign of DLT
QQ4 in Table 14                                 |

WD1 = QQ4(IL4) << 3

        ⎧ WD1          if SIL = =  0            |  Scale table
WD2 =   ⎨                                       |  constant
        ⎩ -WD1         if SIL = = –1            |  Attach sign
```

DLT = DETL ∗ WD2

6.2.1.3 Quantizer scale factor adaptation in the lower sub-band (BLOCK 3L)

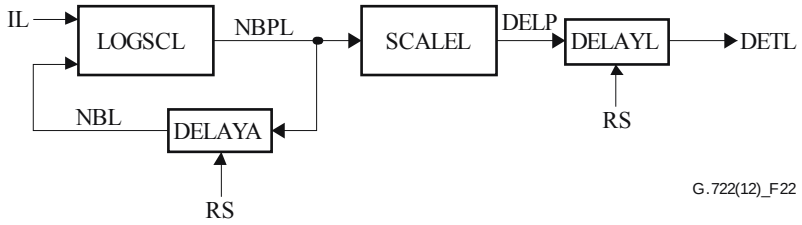


Figure 22 – Quantizer scale factor adaptation in the lower sub-band

LOGSCL

Inputs: IL (ILR in the decoder), NBL

Output: NBPL

Function: Update the logarithmic quantizer scale factor in the lower sub-band.

RIL = IL >>> 2			Delete 2 LSBs
IL4 are obtained from Table 17 using RIL. Use IL4 as an address for WL in Table 14			
WD = NBL ∗ 32512			Leakage factor of 127/128.
NBPL = WD + WL(IL4)			Add scale factor multiplier
NBPL =	0,	if NBPL < 0	Lower limit of 0,
	18432,	if NBPL > 18432	Upper limit of 9,

DELAYA

Inputs: x, RS

Output: y

Function : Memory block. For any input x, the output is given by:

y(n) =	x (n – 1),	if RS = = 0	
	0,	if RS = = 1	Reset to 0.

SCALEL

Inputs: NBPL

Output: DEPL

NOTE – Either Method 1 or Method 2 is used.

Function: Compute the quantizer scale factor in the lower sub-band.

Method 1 (using 353-entry table)

WD1 = (NBPL >> 6) & 511	Compute table address for ILA
WD2 = WD1 + 64	
Use WD2 as an address for ILA in Table 15	
DEPL = (ILA(WD2) + 1) << 2	Scaling by 2-bit shift

Method 2 (using 32-entry table)

WD1 = (NBPL >> 6) & 31	Fractional part of NBPL.
WD2 = NBPL >> 11	Integer part of NBPL.
Use WD1 as an address for ILB in Table 15.	
WD3 = ILB(WD1) >> (8 – WD2)	Scaling with integer part
DEPL WD3 << 2	Scaling by 2-bit shift

DELAYL

Inputs: x, RS

Output: y

Function: Memory block. For the input x, the output is given by:

$$y(n) = \begin{cases} x(n-1), & \text{if RS} = = 0 \\ 32, & \text{if RS} = = 1 \end{cases}$$

Reset to minimum value

6.2.1.4 Adaptive predictor and reconstructed signal calculator in the lower sub-band (BLOCK 4L)

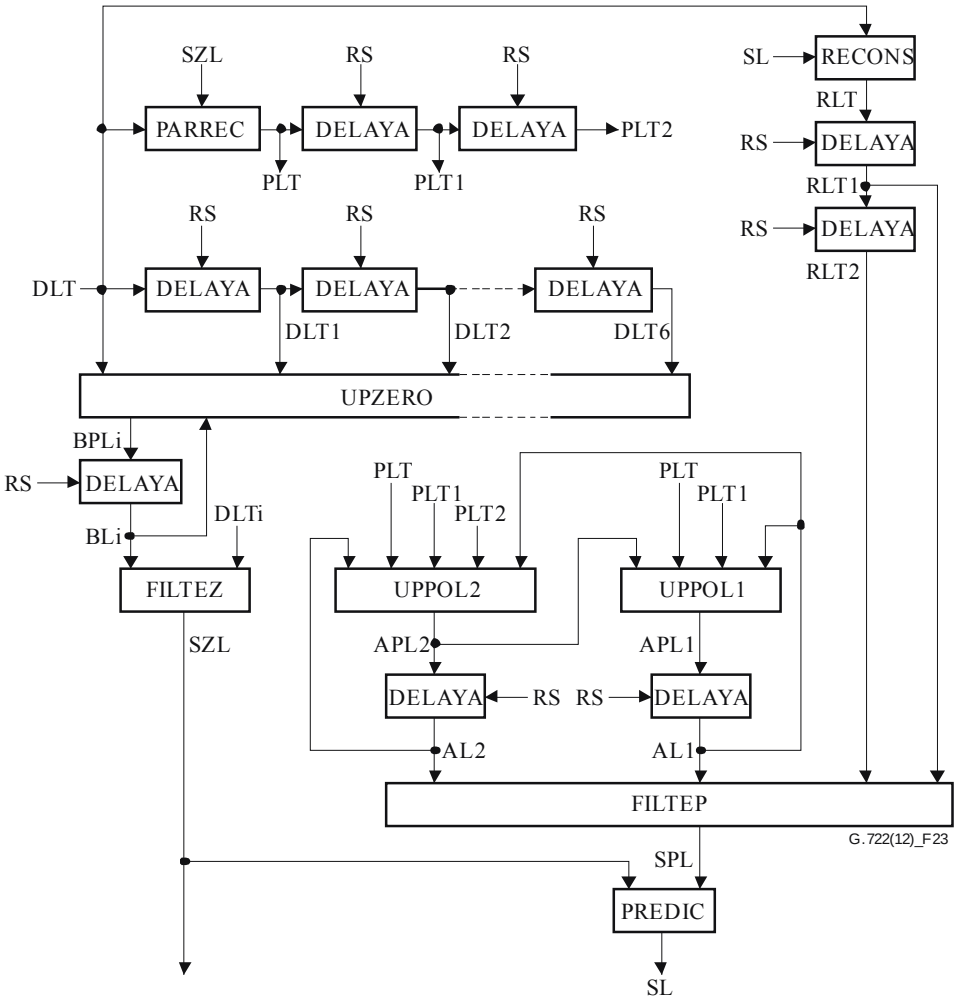


Figure 23 – Adaptive predictor and reconstructed signal calculator in the lower sub-band

---

DELAYA

See clause 6.2.1.3 for specification.

PARREC

Inputs: DLT, SZL

Output: PLT

Function: Compute partially reconstructed signal.

PLT = DLT + SZL

RECONS

Inputs: SL, DLT

Output: RLT

Function: Compute reconstructed signal for the adaptive predictor.

RLT = SL + DLT

UPZERO

Inputs: DLT, DLTi (i = 1 to 6), BLi (i = 1 to 6)

Output: BPLi (i = 1 to 6)

Function: Update sixth-order predictor (zero section) coefficients.

WD1 = 0,	if DLT = = 0	Gain of zero
WD1 = 128,	if DLT ! = 0	Gain of 1/128
SG0 = DLT >> 15		Sign of DLT

Repeat the following computations for i = 1 to 6:

SGi = DLTi >> 15

WD2 = WD1,	if SG0 = = SGi	Sign of DLTi
WD2 = -WD1,	if SG0 ! = SGi	Attach sign to WD1
WD3 = BLi ∗ 32640		Leak factor of 255/256
BPLi = WD2 + WD3		Update zero-section coefficients

UPPOL2

Inputs: ALi (i = 1 and 2), PLT, PLTi (i = 1 and 2)

Output: APL2

Function: Update second predictor coefficient (pole section).

SG0 = PLT >> 15		Sign of PLT
SG1 = PLT1 >> 15		Sign of PLT1
SG2 = PLT2 >> 15		Sign of PLT2
WD1 = AL1 + AL1		Compute f (AL1)
WD1 = WD1 + WD1		[Eq. (3-34) of clause 3.6.3]
WD2 = 0 – WD1	if SG0 = = SG1	Attach correct sign to f(AL1)
WD2 = WD1,	if SG0 ! = SG1	
WD2 = WD2 >> 7		Gain of 1/128

$$WD3 = \begin{cases} 128, & \text{if } SG0 == SG2 \\ -128, & \text{if } SG0 != SG2 \end{cases}$$ | Attach sign to the constant of 1/128

$WD4 = WD2 + WD3$ | Compute gain factor

$WD5 = AL2 * 32512$ | Leak factor of 127/128

$APL2 = WD4 + WD5$ | Update second pole section coefficient

$$APL2 = \begin{cases} 12288, & \text{if } APL2 > 12288 \\ -12288, & \text{if } APL2 < -12288 \end{cases}$$ | Upper limit of +0.75 / Lower limit of –0.75

UPPOL1

Inputs: AL1, APL2, PLT, PLT1

Output: APL1

Function: Update first predictor coefficient (pole section).

$SG0 = PLT >> 15$ | Sign of PLT

$SG1 = PLT1 >> 15$ | Sign of PLT1

$$WD1 = \begin{cases} 192, & \text{if } SG0 == SG1 \\ -192, & \text{if } SG0 != SG1 \end{cases}$$ | Gain 3/256

$WD2 = AL1 * 32640$ | Leak factor of 255/256

$APL1 = WD1 + WD2$ | Update first pole section coefficient

$WD3 = 15360 - APL2$ | Compute $(1 - 2^{-4} - APL2)$

$$APL1 = \begin{cases} WD3, & \text{if } APL1 > WD3 \\ -WD3, & \text{if } APL1 < -WD3 \end{cases}$$ | Upper limit of APL1 / Lower limit of APL1

FILTEZ

Inputs: DLTi (i = 1 to 6), BLi (i = 1 to 6)

Output: SZL

Function: Compute predictor output signal (zero section).

$WD1 = DLT1 + DLT1$ | Compute partial zero section output

$WD1 = BL1 * WD1$

$WD2 = DLT2 + DLT2$

$WD2 = BL2 * WD2$

: : :

: : :

: : :

$WD6 = DLT6 + DLT6$

$WD6 = BL6 * WD6$

$SZL = ((((WD6 + WD5) + WD4) + WD3) + WD2 + WD1$ | Sum the partial zero section outputs

FILTEP

Inputs: RLTi (i = 1 and 2), ALi (i = 1 and 2)

Output: SPL

Function: Compute predictor output signal (pole section).

WD1 = RLT1 + RLT1	Compute partial pole
WD1 = AL1 ∗ WD1	section output
WD2 = RLT2 + RLT2	Sum the partial pole
WD2 = AL2 ∗ WD2	Section outputs
SPL = WD1 + WD2	

PREDIC

Inputs: SPL, SZL

Output: SL

Function: Compute the predictor output value.

SL = SPL + SZL

6.2.1.5 Reconstructed signal calculator for the decoder output in the lower sub-band (BLOCK 5L)

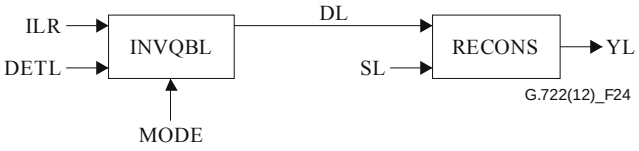


Figure 24 – Reconstructed signal calculator for the decoder output in the lower sub-band

INVQBL

Inputs: ILR, DETL, MODE

Output: DL

NOTE – DL may be substituted by output signal (DLT) of sub-block INVQAL in the case of Mode 3.

Function: Compute quantized difference signal for the decoder output in the lower sub-band.

RIL = ILR		6-bit codeword
SIL and IL6 are obtained from Table18/ using RIL Use IL6 as an address for QQ6 in Table 14	- if MODE = = 1	
WD1 QQ6(IL6) << 3		Scale table constant
RIL IRL >>> 1 SIL and IL5 are obtained		5-bit codeword

from Table 19 using RIL. Use IL5 as an address for QQ5 in Table 14	}	- if MODE = = 2	
WD1 = QQ5(IL5) << 3	)		Scale table constant
RIL = IRL >>> 2	)		4-bit codeword
SIL and IL4 are obtained from Table 17 using RIL. Use IL4 as an address for QQ4 in Table 14	}	- if MODE = = 3	
WD1 = QQ4(IL4) << 3	)		Scale table constant
WD2 = { WD1,		if SIL = = 0	Attach sign
WD2 = { -WD1,		if SIL = = –1	

DL = DETL ∗ WD2

RECONS

See clause 6.2.1.4 for specification. Substitute DL for DLT as input, YL for RLT as output.

6.2.1.6 Reconstructed signal saturation in the lower sub-band (BLOCK 6L)

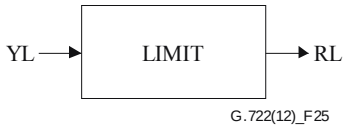


Figure 25 – Reconstructed signal saturation in the lower sub-band

---

LIMIT

Inputs: YL

Output: RL

Function: Limit the output reconstructed signal.

RL = YL

RL = { 16383,	if YL > 16383		Upper limit
RL = { –16384,	if YL < –16384		Lower limit

6.2.2 Description of the higher sub-band ADPCM

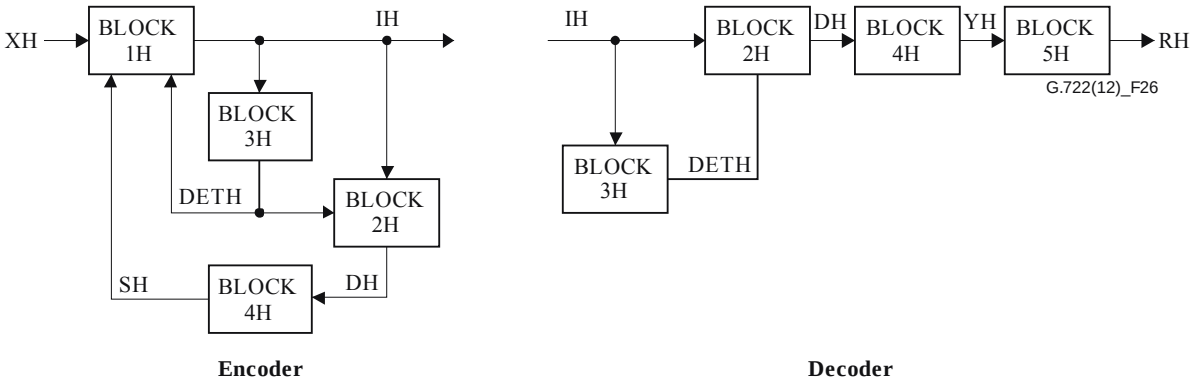


Figure 26 – Higher sub-band ADPCM encoder and decoder

6.2.2.1 Difference signal computation and quantization in the higher sub-band (BLOCK 1H)

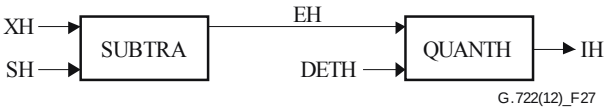


Figure 27 – Difference signal computation and quantization in the higher sub-band

---

SUBTRA

See clause 6.2.1.1 for specification. Substitute XH for XL and SH for SL as inputs, and EH for EL as output.

QUANTH

Inputs: EH, DETH

Output: IH

NOTE – If WD falls exactly on a higher decision level, HDU, the larger adjacent MIH is used.

Function: Quantize the difference signal in the higher sub-band.

SIH = EH >> 15		|	Sign of EH
WD =	EH,	if SIH = = 0 |	Magnitude of EH
	32767, – EH&32767	if SIH = = – 1 |	(Magnitude of EH) – 1

Quantizer decision levels and corresponding MIH values:

WD		MIH
Lower decision level (HDL)	Higher decision level (HDU)	
0	(Q2 (1) << 3) ∗ DETH	1
otherwise		2

Q2 is obtained from Table 14.

IH is obtained from Table 20 using SIH and MIH.

6.2.2.2 Inverse quantization of the difference signal in the higher sub-band (BLOCK 2H)

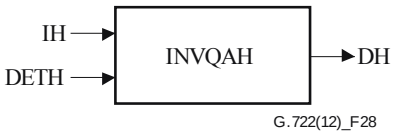


Figure 28 – Inverse quantization of the difference signal in the higher sub-band

INVQAH

Inputs: IH, DETH

Output: DH

Function: Compute the quantized difference signal in the higher sub-band.

SIH and IH2 are obtained from Table 21 using IH.	Derive sign of DH
Use 1H2 as an address for QQ2 in Table 14	
WD1 = QQ2(IH2) << 3	Scale table constant
$WD2 = \begin{cases} WD1, & \text{if SIH} = = 0 \\ -WD1, & \text{if SIH} = = -1 \end{cases}$	Attach sign

DH = DETH ∗ WD2

6.2.2.3 Quantizer scale factor adaptation in the higher sub-band (BLOCK 3H)

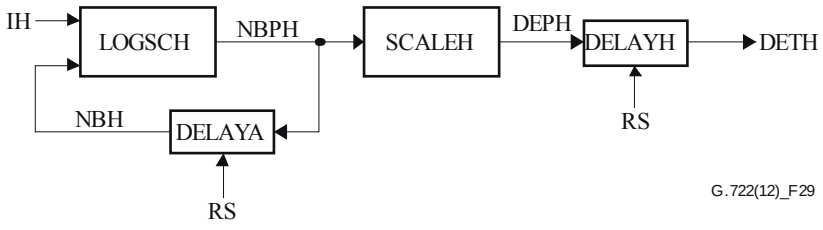


Figure 29 – Quantizer scale factor adaptation in the higher sub-band

LOGSCH

Inputs: IH, NBH

Output: NBPH

Function: Update the logarithmic quantizer scale factor in the higher sub-band.

IH2 is obtained from Table 21 in using IH.

Use IH2 as an address for WH
in Table 14.

WD = NBH ∗ 32512	Leakage factor of 127/128
NBPH = WD + WH (IH2)	Add scale factor multiplier
$NBPH = \begin{cases} 0, & \text{if } NBPH < 0 \\ 22528, & \text{if } NBPH > 22528 \end{cases}$	Lower limit of 0 Upper limit of 11

DELAYA

See clause 6.2.1.3 for specification.

SCALEH

Input: NBPH

Output: DEPH

NOTE – Either Method 1 or Method 2 is used.

Function: Compute the quantizer scale factor in the higher sub-band.

Method 1 (using 353-entry table)

WD = (NBPH >> 6) & 511	Compute table address for ILA

Use WD as an address for ILA
in Table 15.

DEPH = (ILA(WD) + 1) << 2	Scaling by 2-bit shift

Method 2 (using 32-entry table)

WD1 = (NBPH >> 6) & 31	Fractional part of NBPH
WD2 = NBPH >> 11	Integer part of NBPH

Use WD1 as an address for ILB
in Table 15.

WD3 = ILB (WD1) >> (10 – WD2)	Scaling with integer part
DEPH = WD3 << 2	Scaling by 2-bit shift

DELAYH

Inputs: x, RS

Output: y

Function: Memory block. For the input x, the output is given by:

$y(n) = \begin{cases} x(n-1), & \text{if } RS = = 0 \\ 8, & \text{if } RS = = 1 \end{cases}$	Reset to minimum value

6.2.2.4 Adaptive predictor and reconstructed signal calculator in the higher sub-band (BLOCK 4H)

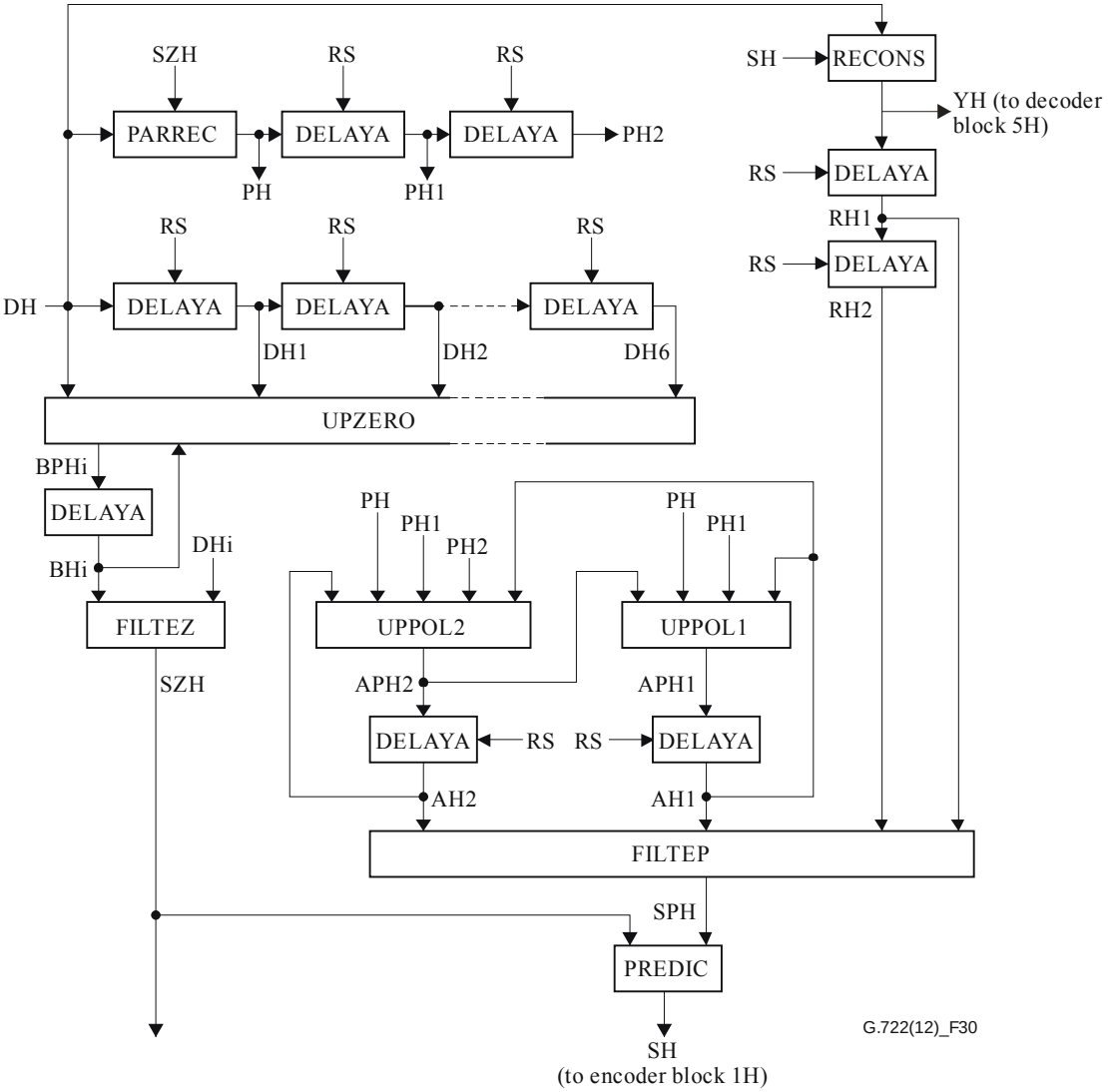


Figure 30 – Adaptive predictor and reconstructed signal calculator in the higher sub-band

---

DELAYA

See clause 6.2.1.3 for specification.

PARREC

See clause 6.2.1.4 for specification. Substitute DH for DLT and SZH for SZL as inputs, and PH for PLT as output.

RECONS

See clause 6.2.1.4 for specification. Substitute SH for SL and DH for DLT as inputs, and YH for RLT as output.

UPZERO

See clause 6.2.1.4 for specification. Substitute DH for DLT, DHi for DLTi, and BHi for BLi as inputs, and BPHi for BPLi as outputs.

UPPOL2

See clause 6.2.1.4 for specification. Substitute AHi for ALi, PH for PLT and PHi for PLTi as inputs, and APH2 for APL2 as output.

UPPOL1

See clause 6.2.1.4 for specification. Substitute AH1 for AL1, APH2 for APL2, PH for PLT and PH1 for PLT1 as inputs, and APH1 for APL1 as output.

FILTEZ

See clause 6.2.1.4 for specification. Substitute DHi for DLTi and BHi for BLi as inputs, and SZH for SZL as output.

FILTEP

See clause 6.2.1.4 for specification. Substitute RHi for RLTi and AHi for ALi as inputs, and SPH for SPL as output.

PREDIC

See clause 6.2.1.4 for specification. Substitute SPH for SPL and SZH for SZL as inputs, and SH for SL as output.

6.2.2.5 Reconstructed signal saturation in the higher sub-band (BLOCK 5H)

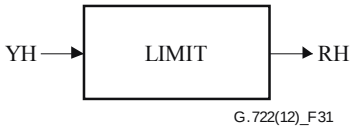


Figure 31 – Reconstructed signal saturation in the higher sub-band

LIMIT

See clause 6.2.1.6 for specification. Substitute YH for YL as input, and RH for RL as output.

Annex A

Testing signal-to-total distortion ratio for 7 kHz audio-codecs at 64 kbit/s Recommendation ITU-T G.722 connected back-to-back

(This annex forms an integral part of this Recommendation.)

The proposals described below are specifically not intended to supplant any requirements of this Recommendation, but rather to suggest the needs of acceptance testing for production quantities of equipments using ITU-T G.722 codes. They concern the measure of the signal-to-total distortion ratio in a loop with SB-ADPCM.

Thus, these specifications do not aim at taking the place of the test digital sequences of the ITU-T G.722 algorithm, but rather to ensure, once these sequences have been checked on a first model, that the quality of the equipments using these codecs is maintained.

The codecs should therefore first of all conform to this Recommendation as a whole; in particular, they should have successfully undergone the digital test sequences of the algorithm and conform to the linear signal-to-total distortion ratio masks through the audio parts (see Figures 14 and 15).

The measuring principle is illustrated in Figure A.3.

The three masks proposed below have been defined to be approximately 2 dB below the results obtained through computer simulation and on reference models.

The first two (see Figure A.1) are given for two frequencies: 1 020 Hz and 6 010 Hz.

The third mask (see Figure A.2) is given for two frequencies (3 900 Hz and 4 100 Hz) and enables us to ascertain the quality of the quadrature mirror filters, as there are no provisions for QMF tests in this Recommendation.

These three masks allow making simpler the evaluation of the mass-produced equipment using ITU-T G.722 codecs, and make easier checks carried out during installation.

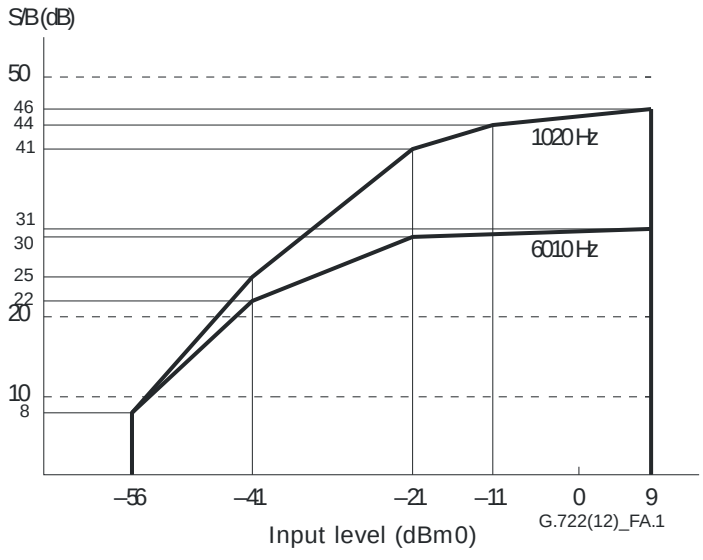


Figure A.1 – Mask for 1 020 Hz and 6 010 Hz frequencies

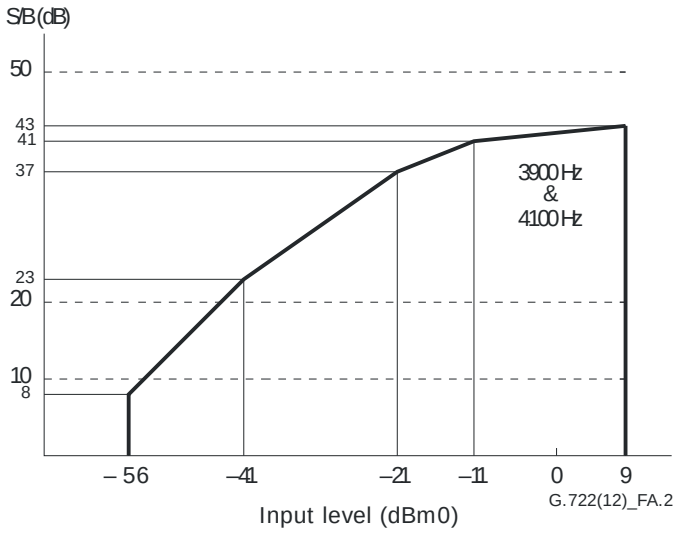


Figure A.2 – Mask for the evaluation of the quadrature mirror filters

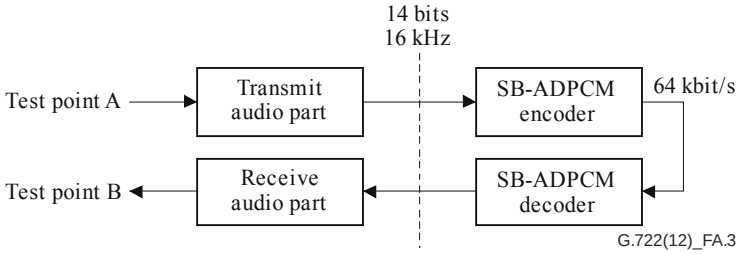


Figure A.3 – Measuring principle

Annex B

Superwideband embedded extension for ITU-T G.722

(This annex forms an integral part of this Recommendation.)

B.1 Scope

This annex[1] contains the description of an algorithm extending ITU-T G.722 for the scalable coding of superwideband speech and audio signals at bitrates from 64 to 96 kbit/s. Part of the bitstream generated by this ITU-T G.722 superwideband extension codec is interoperable with ITU-T G.722.

This annex is organized as follows. The references, definitions, abbreviations and acronyms, and conventions used throughout this annex are defined in clauses B.2, B.3, and B.4, respectively. Clause B.5 gives a general outline of the ITU-T G.722 superwideband extension algorithm. The ITU-T G.722 superwideband extension encoder and decoder principles are discussed in clauses B.6 and B.7, respectively. Clause B.8 describes the software that defines this coder in 16-32 bits fixed-point arithmetic.

B.2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T G.191] Recommendation ITU-T G.191 (2010), *Software tools for speech and audio coding standardization*.

B.3 Abbreviations and acronyms

The abbreviations and acronyms used in this annex are summarized in Table B.3-1.

Table B.3-1 – Glossary of abbreviations and acronyms

Abbreviation or acronym	Description
ADPCM	Adaptive Differential Pulse Code Modulation
AVQ	Algebraic Vector Quantization
BWE	Bandwidth Extension
DeMUX	Demultiplexer
FERC	Frame Erasure Concealment
FIR	Finite Impulse Response
HB	Higher Band (4-8 kHz)
HBE	Higher Band Enhancement
iMDCT	Inverse MDCT

[1] This annex includes an electronic attachment containing the ANSI C source code for the superwideband extension coder.

Table B.3-1 – Glossary of abbreviations and acronyms

Abbreviation or acronym	Description
LB	Lower Band (0-4 kHz)
LP	Linear Prediction
LPC	Linear Prediction Coding
LSB	Least Significant Bit
LTP	Long-Term Prediction
MDCT	Modified Discrete Cosine Transform
MSB	Most Significant Bit
MUX	Multiplexer
NB	Narrow Band
OLA	OverLap and Add
QMF	Quadrature-Mirror Filterbank
RMS	Root Mean Square
SHB	Super Higher Band (8-16 kHz)
SWB	Superwideband (0-16 kHz)
TDAC	Time Domain Aliasing Cancellation
VQ	Vector Quantization
WB	Wideband (0-8 kHz)
WMOPS	Weighted Million Operations Per Second

B.4 Conventions

The notational conventions are detailed below:

– Time-domain signals are denoted by their symbol and a sample index between parentheses, e.g., $s(n)$. The variable n is used as sample index.

– Frequency-domain transforms are denoted by converting the related time-domain signal to capital letters, e.g., $S(k)$ is the transform of $s(n)$. The variable k is used as coefficient index.

– Superscript indices between parentheses (e.g., $g^{(m)}$) are used to indicate time-dependency of variables. The variable m refers, depending on the context, to either a frame or sub-frame index.

– Recursion indices are identified by a superscript between square brackets (e.g., $E^{[k]}$).

– Subscript indices identify a particular element in a coefficient array.

– The symbol ^ identifies a quantized version of a parameter (e.g., $\hat{g}_c$).

– Parameter ranges are given between square brackets, and include the boundaries (e.g., [0.6, ..., 0.9]).

– The sign function gives the polarity of the value and is denoted as sgn(x), where $\mathrm{sgn}(x) = \begin{cases} 1 & if\ x \geq 0 \\ -1 & if\ x < 0 \end{cases}$.

– Integer operator $\lfloor x \rfloor$ denotes rounding of x towards minus infinity ($\lfloor x \rfloor = \max\{n \in Z \mid x \geq n\}$).

– Absolute value calculation of x, performed with saturation operation, are denoted with $|x|$.

– The function $round(x)$ denotes the rounding to the nearest integer, i.e., $round(x) = \text{sgn}(x)\lfloor |x| + 0.5 \rfloor$.

– In some parts, bit special operators are used, where $\otimes$ and $\oplus$ represent the AND bit-operator and the XOR bit-operator, respectively.

– The constants with "0x" prefix mean that the values are noted in hexadecimal.

– N -bit right-shift operations of a variable x are denoted as multiplications with floor of 2 to the power of $-N$, i.e., $\lfloor 2^{-N} x \rfloor$.

– The floating-point numbers used are rounded versions of the values used in the 16-bit fixed-point ANSI C implementation.

Table B.4-1 lists the most relevant symbols used throughout this annex.

Table B.4-1 – Glossary of most relevant symbols

Type	Name	Description
Signals	$s_{SWB}(n)$	Superwideband input signal
	$s_{WB}(n)$	Wideband input signal or wideband signal after QMF processing (decimated)
	$\tilde{s}_{SWB}(n)$	Pre-processed superwideband input signal
	$s_{LB}(n)$	Lower band signal after QMF processing (decimated)
	$s'_{LB}(n)$	Lower band signal with noise feedback
	$s_{SHB}(n)$	Super higher band signal after QMF processing (decimated)
	$s_{SHB}^{fold}(n)$	Spectral folded super higher band signal after QMF processing (decimated)
	$s_{HB}(n)$	Higher band signal after QMF processing (decimated)
	$S_{SHB}(k)$	Super higher band MDCT coefficients
	$S_{SHB}^{norm}(k)$	Normalized super higher band MDCT coefficients
	$S'_{\text{SWBL1}}(k)$	Ordered sub-band normalized super higher band MDCT coefficients for SWB layer 1
	$\hat{s}_{LB}(n)$	Lower band signal after decoding
	$\hat{s}'_{LB}(n)$	Lower band signal after FERC decoding
	$\hat{s}_{HB}(n)$	Higher band signal after decoding
	$\hat{s}'_{HB}(n)$	Higher band signal after FERC decoding
	$S_{SHB}^{err}(k)$	Super higher band MDCT error coefficients in SHB mode 1
	$\hat{S}_{SHB}^{AVQ}(k)$	Decoded sub-band de-normalized super higher band MDCT coefficients
	$\hat{S}_{exc\_base}(k)$	Super higher band MDCT excitation coefficients for BWE

Table B.4-1 – Glossary of most relevant symbols

Type	Name	Description
	$\hat{S}_{SHB}^{BWE}(k)$	BWE decoded super higher band MDCT coefficients
	$\hat{S}_{SHB}^{adp}(k)$	Decoded super higher band MDCT coefficients after BWE/AVQ adaptation
	$\hat{S}_{SHB}(k)$	Decoded super higher band MDCT coefficients before inverse MDCT
	$\hat{S}_{WB}(k)$	Decoded wideband MDCT coefficients
	$\hat{s}_{WB}(n)$	Decoded wideband signal from ITU-T G.722
	$\hat{s}_{SWB}(n)$	Decoded superwideband signal
Parameters	$h_0^{qmf}(i)$	QMF coefficient set 1
	$h_1^{qmf}(i)$	QMF coefficient set 2
	$w_{TDAC}(i)$	Higher band MDCT overlap window
	g_{HB}	RMS value of weighted MDCT coefficients $S_{HBw}(k)$
	$\hat{g}_{HB}$	Gain value of weighted normalized MDCT coefficients $\hat{S}_{HBw}(k)$ in decoder
	g_{adj}	Adjusted super higher band global gain
	$\hat{g}_{adj}$	Decoded adjusted super higher band global gain
	$\hat{f}_{env}(j)$	Decoded spectral envelope of the SHB spectrum
	$f_{rms}(j)$	Normalized spectral envelope of the SHB spectrum
	$\hat{f}_{rms}(j)$	Decoded normalized spectral envelope of the SHB spectrum
	$t_{rms}(j)$	Time envelope of the SHB signal
	$\hat{t}'_{rms}(j)$	Decoded time envelope of the SHB signal
	c_{det}	Problematic zero sub-bands detection counter in SHB
	$f_0, f_1, f_{0,s1}, f_{0,s2}$	Problematic zero sub-bands detection flags in SHB
	I_{HB}^{EL0}	G722EL0 bitstream
	I_{HB}^{EL1}	G722EL1 bitstream
	F_{class}	Super wide band signal class
	w_{bws}	Attenuation factor of bandwidth switching from wideband to superwideband
	w_{bws1}	Additional attenuation factor of bandwidth switching
	w_{bws2}	Weighting factor for the spectral envelope of bandwidth switching from superwideband to wideband
	w_{bws3}	Attenuation factor of bandwidth switching from superwideband to wideband
	E_{avg}, E_{max}	Average and maximum energies of zero sub-bands in SHB

Table B.4-1 – Glossary of most relevant symbols

Type	Name	Description
	$rat(j)$, $rat_{\max}$	Energy ratio in SHB spectrum and its maximum value
	$R_{\max 1}$, $R_{\max 2}$	Maximum correlations in zero sub-bands filling in SHB for SWB layers 1 and 2
	δ_1, δ_2	Lags with the maximum correlation in zero sub-bands filling in SHB
	φ_1, φ_2	Energy correction factors in zero sub-bands filling in SHB
	C_{AVQ}	Base codebook in AVQ coding
	$\mathbf{c}_j$	Lattice point in Gosset lattice
	$\mathbf{z}_j$	Point in the RE_8 base codebook
	$\mathbf{v}_j$	Voronoi extension in AVQ coding
	n_j	AVQ codebook number
	I_j, $\mathbf{I}_j^v$	Vector index in base codebook and 8-dimensional Voronoi index in AVQ coding
	r_j^v	Voronoi extension order in AVQ coding
	M_j^v	Scaling factor in AVQ coding
	β_{avq}	Low energy MDCT coefficients constant for AVQ
	$\Omega_b(j)$	Perceptual importance ordering vector

B.5 General description of the coder

The ITU-T G.722 superwideband extension coder has a monaural superwideband encoding/decoding capability and three scalable operational bitrate modes: ITU-T G.722 core R1sm/R2sm/R3sm. Here, "Rx" specifies the rate and R1, R2 and R3 correspond to 64, 80, and 96 kbit/s modes, respectively. The notation "sm" after the rate specifier indicates that the modes are in "superwideband monaural". The ITU-T G.722 core of R1sm works at 56 kbit/s, while the core of R2sm/R3sm works at 64 kbit/s.

The underlying algorithm includes three main parts: higher band enhancements based on higher resolution scalable 3 and 4 bit quantizers, bandwidth extension and transform coding in MDCT domain based on algebraic vector quantization.

B.5.1 Coder modes and bit allocation

ITU-T G.722 R1sm, R2sm and R3sm include wideband enhancement, and bandwidth extension. Meanwhile, ITU-T G.722 R2sm and R3sm also include algebraic vector quantization (AVQ) and bandwidth extension/algebraic vector quantization adaptation. Signal class is identified in the bandwidth extension module, i.e., TRANSIENT or non-TRANSIENT. Table B.5-1 illustrates the mode definition of the wideband and superwideband coder modes.

Table B.5-1 – Mode definition

Coder Mode	Description
MODE_R0wm	ITU-T G.722 R0wm, 56 kbit/s wide band
MODE_R1wm	ITU-T G.722 R1wm, 64 kbit/s wide band
MODE_R1sm	ITU-T G.722 R1sm, 64 kbit/s superwideband (R0wm core)
MODE_R2sm	ITU-T G.722 R2sm, 80 kbit/s superwideband (R1wm core)
MODE_R3sm	ITU-T G.722 R3sm, 96 kbit/s superwideband (R1wm core)

Table B.5-2 and Table B.5-3 give the bitrate of the layers and the coder modes. Layers G722EL0 and SWBL0 are combined to form R1sm bitstream. SWBL1 is added on top of R1sm to construct R2sm bitstream. Extra layers, G722EL1 and SWBL2, are added on top of R2sm to construct R3sm bitstream. Here, G722EL0 and G722EL1 are the layers that contain the enhanced ITU-T G.722 higher band bitstream. SWBL0 is the layer for bandwidth extension, and SWBL1/SWBL2 are used for algebraic vector quantization. Note that G722EL0 with 3.8 kbit/s is only used in non-TRANSIENT frames. The bitrates for SWBL0 are 8 kbit/s and 4.2 kbit/s for TRANSIENT and non-TRANSIENT frames, respectively.

Table B.5-2 – Layer bit allocation

Layer name	Bits per frame		Bitrate [kbit/s]	
	TRANSIENT	Non-TRANSIENT	TRANSIENT	Non-TRANSIENT
G722EL0	0	19	0.0	3.8
SWBL0	40	21	8.0	4.2
SWBL1	40		8.0	
G722EL1	40		8.0	
SWBL2	40		8.0	

Table B.5-3 – The used layers in given coder modes

Coder mode	Core layer (kbit/s)	SWBL0 (TRANSIENT)		G722EL1	SWBL1	SWBL2	Overall bitrate (kbit/s)
		G722EL0 (non-TRANSIENT)	SWBL0 (non-TRANSIENT)				
R1sm	56	x		–	–	–	64
R2sm	64	x		–	x	–	80
R3sm	64	x		x	x	x	96

Note that this bitstream structure is fully scalable, lower bitrates can be obtained by simply omitting certain parts of the bitstream at a higher bitrate.

B.5.2 Input/output sampling rate

This coder operates with a 16-bit linear PCM digital signal sampled at 32 kHz as input to the encoder. Similarly, the format of the decoder output is 16-bit linear PCM with a sampling frequency of 32 kHz. Other input/output characteristics should be converted to 16-bit linear PCM with a 32 kHz sampling rate before encoding, or from 16-bit linear PCM to the appropriate format after decoding. The bitstream from the encoder to the decoder is defined within this annex.

B.5.3 Algorithmic delay

The ITU-T G.722 superwideband extension coder has an algorithmic delay of 12.3125 ms. The delay contributions are listed below:

– 5 ms for input frame;

– 5 ms for the MDCT overlap-add;

– 1.375 ms for the QMF analysis-synthesis filterbank in ITU-T G.722 core (wideband);

– 0.9375 ms for QMF analysis-synthesis filterbank for superwideband layers.

B.5.4 Computational complexity and storage requirements

The observed worst-case complexity and storage requirements in 16-bit words of the ITU-T G.722 superwideband extension coder (encoder plus decoder) are based on the basic operators of the ITU-T STL2009 Software Tool Library in [ITU-T G.191] and are detailed in Table B.5-4, in function of the mode.

Table B.5-4 – Complexity of the ITU-T G.722-SWB coder

Mode		**R1sm**	**R3sm**
Worst case complexity (in WMOPS)	Encoder	7.926	10.935
	Decoder, no FER	9.923	R2sm: 11.362 R3sm: 11.801
	Decoder, FER (Note)	10.613	R2sm: 11.400 R3sm: 11.825
	Overall	18.539	22.760
Dynamic RAM and Static RAM		4.634 kwords	
Data ROM		2.973 kwords	
Program ROM		4 905 operators	
NOTE – Calculated 10.7% with frame erasure rate.			

B.6 Functional description of the encoder

B.6.1 Encoder

The encoder block diagram for the ITU-T G.722 superwideband (SWB) extension is shown in Figure B.6-1. A pre-processing high-pass filter is applied to the 32-kHz-sampled input signal $s_{SWB}(n)$ to remove 0-50 Hz components. The pre-processed signal $\tilde{s}_{SWB}(n)$ is divided into two 16-kHz-sampled wideband (0-8 kHz) and super higher band (8-16 kHz) signals, $s_{WB}(n)$ and $s_{SHB}(n)$, using a 32-tap quadrature mirror filterbank (QMF) applied to the 5 ms input frame size. The wideband signal $s_{WB}(n)$ is encoded with an ITU-T G.722 enhanced core encoder which produces an ITU-T G.722 bitstream. The super higher band signal $s_{SHB}(n)$ is encoded with a super higher band (SHB) encoder: after transformation into modified discrete cosine transform (MDCT) domain, the frequency domain coefficients $S_{SHB}(k)$ are encoded by bandwidth extension (BWE) and algebraic vector quantization (AVQ) algorithms with a scalable bitstream.

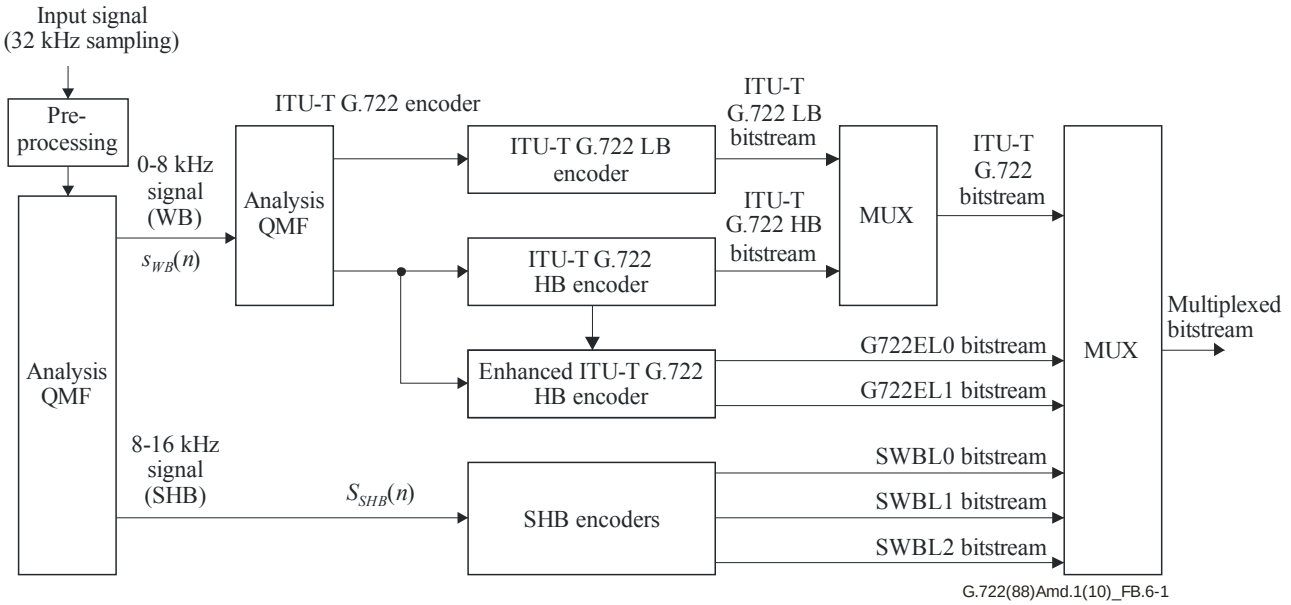


Figure B.6-1 – High-level encoder block diagram of ITU-T G.722 superwideband extension

B.6.2 Pre-processing high-pass filter

The pre-processing filter applied to the 32-kHz sampled input signal $s_{SWB}(n)$ is defined as

$$\tilde{s}_{SWB}(n) = 0.984375 \cdot \tilde{s}_{SWB}(n-1) + s_{SWB}(n) - s_{SWB}(n-1) \qquad n = 0,\cdots,159 \qquad \text{(B.6-1)}$$

where $\tilde{s}_{SWB}(n)$ is the filter output. The pre-processing high-pass filter is designed with a cut-off frequency at 50 Hz.

B.6.3 Analysis QMF

An analysis QMF, h^{qmfA}, is applied to the high-pass filtered input signal $\tilde{s}_{SWB}(n)$ in order to split it into two 16-kHz-sampled signals; wideband signal $s_{WB}(n)$ and super higher band signal $s_{SHB}(n)$. The 32-kHz-sampled wideband signal $s_{SWW}(n)$ is obtained by filtering the 32 kHz sampled pre-processed signal $\tilde{s}_{SWB}(n)$ through a symmetric FIR low-pass filter with 32 coefficients given by:

$$s_{SWW}(n) = \sum_{i=0}^{31} h_L^{qmfA}(i)\,\tilde{s}_{SWB}(n-i) \qquad n = 0,\cdots,159 \qquad \text{(B.6-2)}$$

where $h_L^{qmfA}(i)$ are the filter coefficients. The 16 kHz sampled WB signal $s_{WB}(n)$ is then obtained by decimating $s_{SWW}(n)$ by a factor of 2:

$$s_{WB}(n) = s_{SWW}(2n+1) \qquad n = 0,\cdots,79 \qquad \text{(B.6-3)}$$

Similarly, the 16-kHz-sampled super higher band signal $s_{SHB}(n)$ is obtained by filtering the 32 kHz sampled pre-processed signal $\tilde{s}_{SWB}(n)$ through a FIR high-pass filter with 32 coefficients, then decimating the filter output signal $s_{SWSH}(n)$ by a factor of 2:

$$s_{SWSH}(n) = \sum_{i=0}^{31} h_H^{qmfA}(i)\,\tilde{s}_{SWB}(n-i) \qquad n = 0,\cdots,159 \qquad \text{(B.6-4)}$$

$$s_{SHB}(n) = s_{SWSH}(2n+1) \qquad n = 0,\cdots,79 \qquad \text{(B.6-5)}$$

The high-pass and low-pass filter coefficients $h_H^{qmfA}(i)$ and $h_L^{qmfA}(i)$ have the following relationship:

$$h_H^{qmfA}(i) = (-1)^i h_L^{qmfA}(i) \qquad i = 0,\cdots,31 \qquad \text{(B.6-6)}$$

Therefore, $s_{WB}(n)$ and $s_{SHB}(n)$ can be directly computed as follows:

$$s_{WB}(n) = \sum_{i=0}^{15} h_0^{qmf}(i)\tilde{s}_{SWB}(2(n-i)) + \sum_{i=0}^{15} h_1^{qmf}(i)\tilde{s}_{SWB}(2(n-i)+1) \qquad n = 0,\cdots,79 \qquad \text{(B.6-7)}$$

$$s_{SHB}(n) = -\sum_{i=0}^{15} h_0^{qmf}(i)\tilde{s}_{SWB}(2(n-i)) + \sum_{i=0}^{15} h_1^{qmf}(i)\tilde{s}_{SWB}(2(n-i)+1) \qquad n = 0,\cdots,79 \qquad \text{(B.6-8)}$$

where:

$$\begin{aligned} h_0^{qmf}(i) &= h_H^{qmfA}(2i) \\ h_1^{qmf}(i) &= h_L^{qmfA}(2i+1) \end{aligned} \qquad i = 0,\cdots,15 \qquad \text{(B.6-9)}$$

Table B.6-1 gives the values of the coefficients h_0^{qmf} and h_1^{qmf}.

The super higher band signal $s_{SHB}(n)$ is spectrally folded as follows:

$$s_{SHB}^{fold}(n) = (-1)^n s_{SHB}(n) \qquad n = 0,\cdots,159 \qquad \text{(B.6-10)}$$

Table B.6-1 – QMF coefficients

i	$h_0^{qmf}(i)$	$h_1^{qmf}(i)$
0	0.00064087	–0.00134277
1	–0.00125122	0.00415039
2	0.00143433	–0.0093689
3	–0.00018311	0.0178833
4	–0.00411987	–0.03115845
5	0.01446533	0.05291748
6	–0.03924561	–0.09979248
7	0.128479	0.46646118
8	0.46646118	0.128479
9	–0.09979248	–0.03924561
10	0.05291748	0.01446533
11	–0.03115845	–0.00411987
12	0.0178833	–0.00018311
13	–0.0093689	0.00143433
14	0.00415039	–0.00125122
15	–0.00134277	0.00064087

B.6.4 ITU-T G.722 core, G722EL0 and G722EL1 layer encoder

The ITU-T G.722 core layer coder is an improved version of ITU-T G.722. As shown in the block diagram of Figure B.6-1, the ITU-T G.722 core and extension coder comprises the analysis QMF, the ITU-T G.722 LB and HB encoder, and extension coders: the enhanced ITU-T G.722 HB encoders (EL0 and EL1). These blocks are described in the following clauses.

B.6.4.1 Analysis QMF

The same as clause 3.1 of this Recommendation. Note that the analysis QMF is adapted to operate with 5 ms frames.

The wideband input $s_{WB}(n)$ is decomposed into a lower band output signal $s_{LB}(n)$ and a higher band output signal $s_{HB}(n)$. Note that these signals, $s_{WB}(n)$, $s_{LB}(n)$, and $s_{HB}(n)$, are respectively denoted x_{in}, x_L, and x_H in the main body of this Recommendation.

B.6.4.2 ITU-T G.722 LB encoder (5 or 5+1 bit/sample)

The ITU-T G.722 LB encoder is a bitstream interoperable, optimized version of the lower band embedded ADPCM encoder described in clauses 3.2-3.6 of ITU-T G.722. A block diagram of the ITU-T G.722 LB encoder is shown in Figure B.6-2. The legacy ITU-T G.722 LB encoder is modified to perceptually shape the embedded ADPCM coding noise.

The lower band ADPCM coding in clauses 3.2-3.6 of ITU-T G.722 operates at only 48 kbit/s (6 bit/sample), with the possibility to skip afterwards one or two bits per sample to reduce the bitrate to 40 or 32 kbit/s, whereas the ITU-T G.722 LB encoder of this Annex B can operate at either 40 kbit/s (5 bit/sample) for the G722R1sm mode or 48 kbit/s (6 bit/sample) for the G722R2sm and G722R3sm modes.

To shape the coding noise at both bitrates (40 and 48 kbit/s) in a coherent and optimal way, the embedded ADPCM coding in clauses 3.2-3.6 of ITU-T G.722 is separated in two stages, allowing the inclusion of a noise feedback loop in the first stage and the use of analysis by synthesis in the second stage. Hence:

– A noise feedback loop is used with the ADPCM coder operating at a reduced bitrate of 40 kbit/s (5 bit/sample).
– An enhancement encoder using analysis by synthesis and operating at 1 bit/sample brings the bitrate to 48 kbit/s (6 bit/sample).

Both stages use the same noise-shaping filter $F_{LB}(z)-1$.

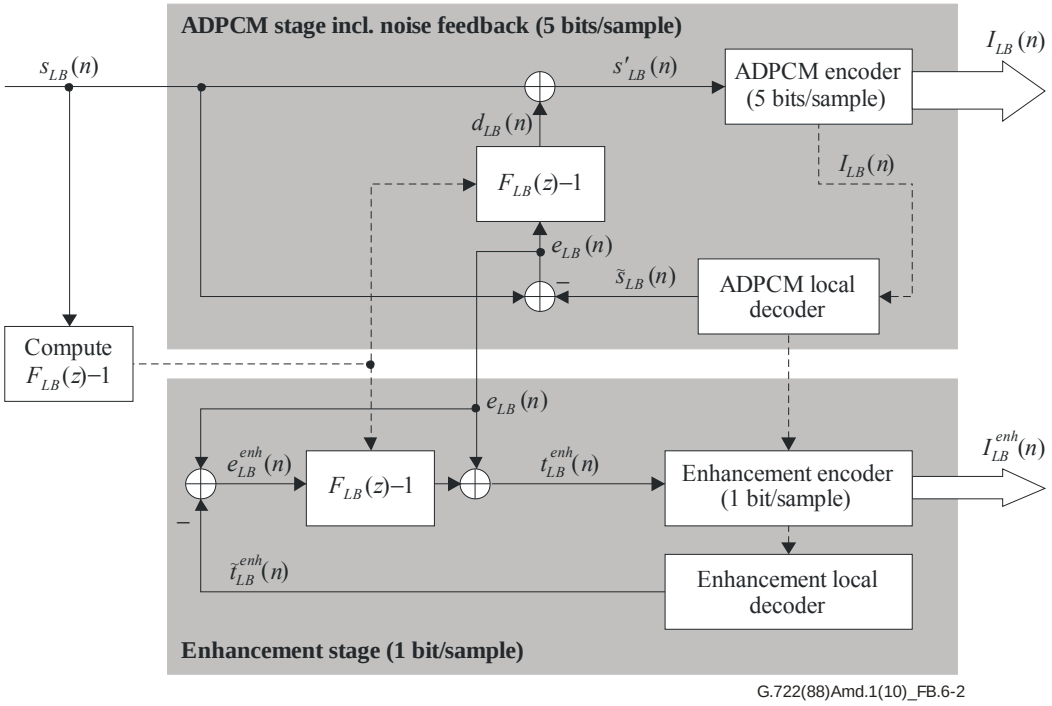


Figure B.6-2 – ITU-T G.722 LB encoder

B.6.4.2.1 Computation of noise-shaping filter $F_{LB}(z)-1$

The perceptual noise-shaping filter is given by $F_{LB}(z)-1$ where $F_{LB}(z)$ is derived from the linear prediction (LP) filter $A_{LB}(z)$ defined below. It can be shown that using the noise feedback loop in Figure B.6-2 the spectrum of the ADPCM quantization noise is shaped by $1/F_{LB}(z)$.

Pre-emphasis and linear predictive analysis

The LP filter $A_{LB}(z)$ is computed after pre-emphasizing the signal $s_{LB}(n)$, $n = -40,\cdots,39$, where the negative indices refer to the past signal – (the ranges $[-40,\cdots,-1]$ and $[0,\cdots,39]$ represent the previous and current frames, respectively). To control the tilt in noise shaping and improve the perceptual quality of lower band encoding, an adaptive pre-emphasis factor is used. The pre-emphasis filter is a first-order filter with a transfer function $P(z) = 1 - \beta_e z^{-1}$, where β_e is signal-dependent and is calculated as:

$$\beta_e = 0.38275 + 0.007813\, c_{zc1} \qquad \text{(B.6-11)}$$

where c_{zc1} is a zero-crossing rate. The zero-crossing rate on the previous and current frames is calculated as:

$$c_{zc1} = \frac{1}{2}\sum_{n=-39}^{39} \left| \operatorname{sgn}\left(s_{LB}(n-1)\right) + \operatorname{sgn}\left(s_{LB}(n)\right) \right| \qquad \text{(B.6-12)}$$

This results in $0.38 < \beta_e < 1.0$. The pre-emphasized signal $s_{LB}^{pre}(n)$ is obtained as:

$$s_{LB}^{pre}(n) = s_{LB}(n) - \beta_e s_{LB}(n-1) \qquad n = -39,\cdots,39 \qquad \text{(B.6-13)}$$

A 4th-order LP analysis is performed on the pre-emphasized signal once per frame using an asymmetric window. The window is divided in two parts: the length of the first part is 60 samples and the length of the second part is 20 samples. The window function is given by:

$$w_{LP}(n)=\begin{cases}0 & n=0\\ \frac{1}{2}\cos\left(\frac{\pi}{2L_1}\left(n+\frac{1}{2}\right)-\frac{\pi}{2}\right)+\frac{1}{2}\cos^2\left(\frac{\pi}{2L_1}\left(n+\frac{1}{2}\right)-\frac{\pi}{2}\right) & n=1,...,L_1-1\\ \frac{1}{2}\cos\left(\frac{\pi}{2L_2}\left(n-L_1+\frac{1}{2}\right)\right)+\frac{1}{2}\cos^2\left(\frac{\pi}{2L_2}\left(n-L_1+\frac{1}{2}\right)\right) & n=L_1,...,L_1+L_2-1\end{cases} \quad \text{(B.6-14)}$$

where L_1=60 and L_2=20. The pre-emphasized signal $s_{LB}^{pre}(n)$ is multiplied with this window to obtain the signal $s_{LB}^{w}(n)$:

$$s_{LB}^{w}(n)=w_{LP}(n)s_{LB}^{pre}(n-40) \qquad n=0,\cdots,79 \qquad \text{(B.6-15)}$$

The autocorrelation function of the windowed signal $s_{LB}^{w}(n)$ is computed by:

$$r_{LB}(k)=\varepsilon_{LB}(k)+\sum_{n=k}^{79}s_{LB}^{w}(n)s_{LB}^{w}(n-k) \qquad k=0,\cdots,4 \qquad \text{(B.6-16)}$$

where $\varepsilon_{LB}(k)$ is an initialization value added to each correlation coefficient to ensure a proper shape of the noise-shaping filter for signals with low energy, given by $\varepsilon_{LB}(k)=0.95^k\times100^2$. A 120 Hz bandwidth expansion is then applied by lag-windowing the autocorrelation function. The lag windowing is a multiplication of $w_{lag}(k)$ with the correlation function. That is:

$$r'_{LB}(k)=\begin{cases}r_{LB}(k) & k=0\\ w_{lag}(k)r_{LB}(k) & k=1,...,4\end{cases} \quad \text{(B.6-17)}$$

The windowing function is defined as:

$$w_{lag}(i)=\frac{1}{1.0001}\exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right] \qquad i=1,\cdots,4 \qquad \text{(B.6-18)}$$

where f_0=120 Hz is the bandwidth expansion and f_s = 8 000 Hz is the sampling frequency. The multiplication factor 1/1.0001 is a white noise correction to stabilize LP filter coefficients calculation, and this is equivalent to adding a noise floor at –40 dB below the windowed signal level.

The bandwidth-expanded autocorrelations $r'_{LB}(k)$ are used to obtain the LP filter coefficients a_{LBj}, with the Levinson-Durbin algorithm. This algorithm is performed in the following recursive steps:

1) Set iteration number $i=1$, $a_{LB0}^{[0]}=1.0$, and $E^{[0]}=r'_{LB}(0)$

2) Compute $k_i=-\frac{1}{E^{[i-1]}}\left(r'_{LB}(i)+\sum_{j=1}^{i-1}a_{LBk}^{[i-1]}r'_{LB}(i-j)\right)$

3) Set $a_{LBi}^{[i]}=k_i$

4) Compute $a_{LBj}^{[i]}=a_{LBj}^{[i-1]}+k_i a_{LB(i-j)}^{[i-1]}$ for $j=1,...,i-1$

5) Compute $E^{[i]} = (1 - k_i^2)E^{[i-1]}$

6) Increment i by 1 and go back to step 2, until i reaches 4

The final solution is given as $a_{LBj} = a_{LBj}^{[4]}$, j=1,⋯,4.

The result of the LP analysis is a filter with the transfer function:

$$A_{LB}(z) = 1 + a_{LB1}z^{-1} + a_{LB2}z^{-2} + a_{LB3}z^{-3} + a_{LB4}z^{-4} \qquad \text{(B.6-19)}$$

where a_{LBi}, i=1,⋯,4, are the LP coefficients obtained from the lower band signal.

Derivation of noise shaping filter

The noise-shaping filter with transfer function

$$F_{LB}(z) = 1 + a'_{LB1}z^{-1} + a'_{LB2}z^{-2} + a'_{LB3}z^{-3} + a'_{LB4}z^{-4} \qquad \text{(B.6-20)}$$

is derived from $A_{LB}(z)$ as explained below and illustrated in Figure B.6-3 with M=4.

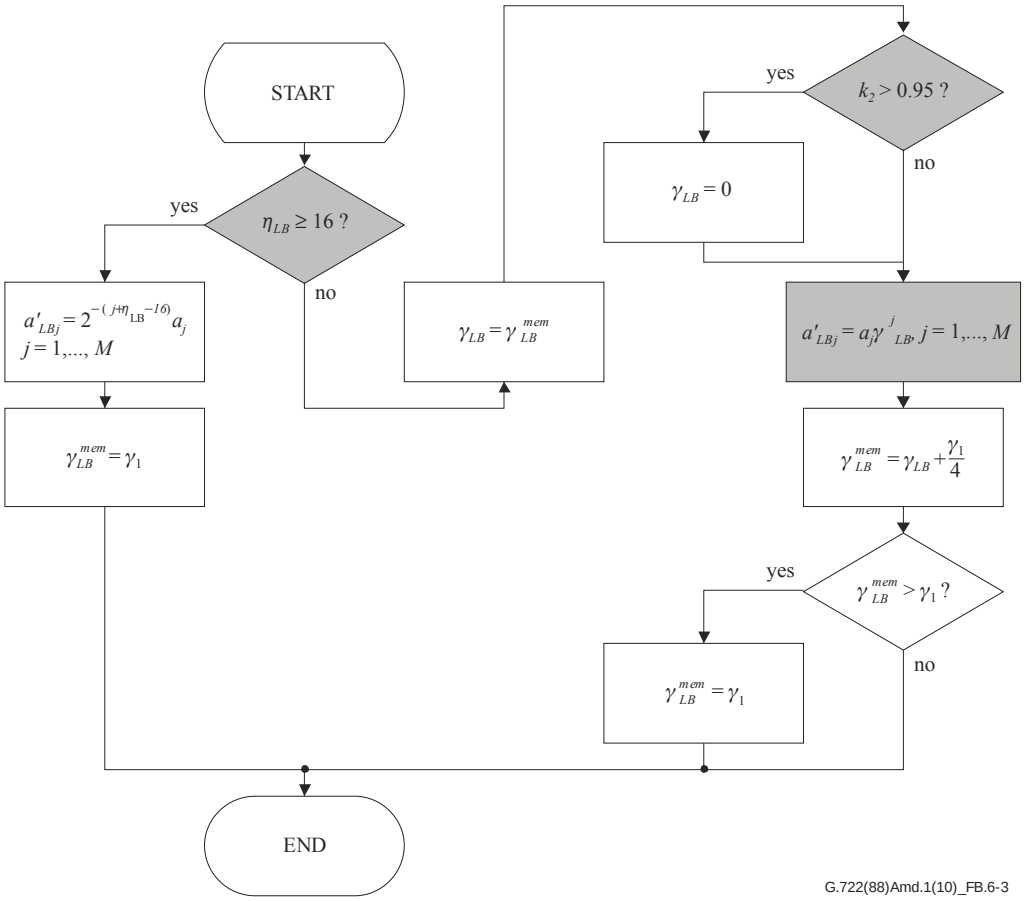


Figure B.6-3 – Computation of $F_{LB}(z) - 1$ in ITU-T G.722 LB encoder

By default, $F_{LB}(z)$ is a weighted version of $A_{LB}(z)$, as follows: $F_{LB}(z) = A_{LB}(z/\gamma_1)$ with $\gamma_1 = 0.92$. The $F_{LB}(z)$ filter coefficients are calculated from $A_{LB}(z)$ coefficients as follows:

$$a'_{LBj} = \gamma_1^j a_{LBj}, j = 1,..., M \qquad \text{(B.6-21)}$$

This default operation (shown in the shaded rectangular box in Figure B.6-3) is changed in two special cases:

– low level input signal;

– risk of instability in noise feedback loop.

These two cases are detailed below.

Attenuation of the perceptual filter for signals with very low level

In order to detect low level signals, the normalization factor η_{LB} is calculated with:

$$\eta_{LB} = 30 - \lfloor \log_2 (r_{LB}(0)) \rfloor \quad \text{(B.6-22)}$$

where $r_{LB}(0)$ is the first autocorrelation coefficient calculated in Equation (B.6-16).

When the input signal has a very low energy, the noise-shaping may fail to properly mask the ADPCM coding noise. This special case of very low energy signal is detected when the normalization factor η_{LB} fulfils the condition:

$$\eta_{LB} \geq 16 \quad \text{(B.6-23)}$$

In this case, the attenuation for very low level signal is performed and the attenuated perceptual weighting filter of Equation (B.6-20) becomes:

$$a'_{LBj} = 2^{-(j+\eta_{LB}-16)} a_{LBj}, j = 1,...,M \quad \text{(B.6-24)}$$

Attenuating the perceptual noise-shaping filter for very low level input signals avoids the case where the noise feedback loop would increase the objective noise level without bringing the benefit of having a perceptually lower noise floor.

Deactivation of noise feedback loop in case of risk of instability and progressive reactivation

The noise feedback loop may become unstable when the noise masking filter is highly resonant and the step size and predictors in ADPCM coding are adapted too slowly to catch up with quick variations between segments of high spectral dynamics (e.g., sinusoids). ITU-T G.722 ADPCM coding may exhibit problems of mistracking (i.e., temporary divergence of adaptation), which would be amplified by the noise feedback loop. This can cause audible artefacts lasting for several consecutive frames until the noise-shaping loop and ADPCM states converge.

To prevent such problems, the noise-shaping feedback is deactivated whenever a signal with high spectral dynamics, giving a risk of instability, is detected in the encoder.

This detection is based on the second reflection coefficient k_2 obtained in the Levinson-Durbin algorithm. If condition:

$$k_2 > 0.95 \quad \text{(B.6-25)}$$

is fulfilled, the perceptual weighting filter is used with the weighting factor $\gamma_{LB} = 0$ (i.e., $F_{LB}(z) = 1$).

The normal operation is restored progressively as soon as the condition $k_2 > 0.95$ is not fulfilled in the current frame; in that case, the weighting factor γ_{LB} is incremented by steps of $\frac{\gamma_1}{4}$ in a predetermined number of successive frames (here 4) until the value γ_1 is reached.

Note that to enable the progressive reactivation of the noise feedback loop, the value of γ_{LB} is stored in a memory γ_{LB}^{mem} which is loaded at the beginning and updated at the end of every frame.

Furthermore, when low level signals are detected, the value in γ_{LB}^{mem} is set to the default value γ_1 to restore the normal state of operation.

B.6.4.2.2 ADPCM encoder (5 bit/sample) with noise feedback

In the first stage of ITU-T G.722 LB coding (core coding), noise feedback is performed for each sample *n* by combining the signal $s_{LB}(n)$ with the filtered ADPCM coding noise, $d_{LB}(n)$, as follows:

$$s'_{LB}(n) = s_{LB}(n) + d_{LB}(n) \qquad n = 0, \cdots, 39 \qquad \text{(B.6-26)}$$

where $d_{LB}(n)$ is the result of the filtering operation:

$$d_{LB}(n) = \sum_{i=1}^{M} a'_{LBi}\, e_{LB}(n-i) \qquad n = 0, \cdots, 39 \qquad \text{(B.6-27)}$$

$e_{LB}(n-i)$ being the ADPCM coding noise at the past sample $n-i$ (see Equation (B.6-30) below).

The signal $s'_{LB}(n)$ including noise feedback is encoded by an ADPCM encoder operating at 5 bit/sample. This encoder is equivalent (bit-exact) to the embedded ADPCM encoder in clauses 3.2-3.6 of ITU-T G.722, in which the LSB of the output 6-bit index $I_L(n)$ (see clause 3.3 of ITU-T G.722) is forced to zero:

$$I_{LB}(n) = \left\lfloor \frac{I_L(n)}{2} \right\rfloor \qquad n = 0, \cdots, 39 \qquad \text{(B.6-28)}$$

Therefore the index of the first stage scalar quantizer, $I_{LB}(n)$, is represented with 5 bits for each sample of the ITU-T G.722 lower band.

A local ADPCM decoder operating at 5 bit/sample reconstructs the decoded sample $\tilde{s}_{LB}(n)$ from the index $I_L(n)$:

$$\tilde{s}_{LB}(n) = d_L(n) + s_L(n) \qquad n = 0, \cdots, 39 \qquad \text{(B.6-29)}$$

where $d_L(n)$ is the quantized difference signal (using 5 bits) and $s_L(n)$ is the predicted signal, as defined in clauses 3.2-3.6 of ITU-T G.722.

The coding noise $e_{LB}(n)$ is then computed as follows:

$$e_{LB}(n) = s_{LB}(n) - \tilde{s}_{LB}(n) \qquad n = 0, \cdots, 39 \qquad \text{(B.6-30)}$$

B.6.4.2.3 Enhancement encoder (1 bit/sample) using analysis by synthesis

In the second stage of ITU-T G.722 LB coding, the LSB of the 6-bit index of clauses 3.2-3.6 is searched using an analysis by synthesis method for each sample *n*, based on the same noise masking filter $F_{LB}(z)-1$ as in the first stage of ITU-T G.722 LB coding.

The target sample $t_{LB}^{enh}(n)$ is computed as follows:

$$t_{LB}^{enh}(n) = e_{LB}(n) - \sum_{i=1}^{M} a'_{LBi}\, e_{LB}^{enh}(n-i) \qquad n = 0, \cdots, 39 \qquad \text{(B.6-31)}$$

where $e_{LB}^{enh}(n-i)$ is the coding noise of the second stage at the past sample $n-i$ (see Equation (B.6-34) below).

Each target sample $t_{LB}^{enh}(n)$, $n = 0,\cdots,39$, is then quantized by a 1-bit scalar quantizer, minimizing the mean square error between $t_{LB}^{enh}(n)$ and two possible quantized values $\left\{\xi_{LB}^{i}(n)\right\}_{i=0,1}$ defined as:

$$\xi_{LB}^{i}(n) = \Delta_{\mathrm{L}}(n)\left(\mathrm{sgn}\left(2I_{LB}(n)+i\right)\mathrm{QL6}^{-1}\left[2I_{LB}(n)+i\right] - \mathrm{sgn}\left(I_{LB}(n)\right)\mathrm{QL5}^{-1}\left[I_{LB}(n)\right]\right) \qquad \text{(B.6-32)}$$

where $\mathrm{QL6}^{-1}[.]$ and $\mathrm{QL5}^{-1}[.]$ are given in Table 7 of ITU-T G.722. $\Delta_{\mathrm{L}}(n)$ is the step size of the ITU-T G.722 lower band ADPCM encoder given in Equation (3-17) of ITU-T G.722.

The index of the second stage, $I_{LB}^{enh}(n)$, is calculated as:

$$I_{LB}^{enh}(n) = \arg\min_{i=0,1}\left|t_{LB}^{enh}(n) - \xi_{LB}^{i}(n)\right|^2 \qquad n = 0,\cdots,39 \qquad \text{(B.6-33)}$$

The local decoder in the enhancement stage reconstructs $\tilde{t}_{LB}^{enh}(n) = \xi_{LB}^{I_{LB}^{enh}(n)}(n)$ in order to update the coding noise $e_{LB}^{enh}(n)$ for the next sample:

$$e_{LB}^{enh}(n) = e_{LB}(n) - \tilde{t}_{LB}^{enh}(n) \qquad n = 0,\cdots,39 \qquad \text{(B.6-34)}$$

B.6.4.3 ITU-T G.722 HB encoder (2 bit/sample) and enhanced ITU-T G.722 HB encoders (ITU-T G.722 EL0 and EL1 layers)

Block diagrams of the ITU-T G.722 HB and enhanced ITU-T G.222 HB encoders are shown in Figure B.6-4.

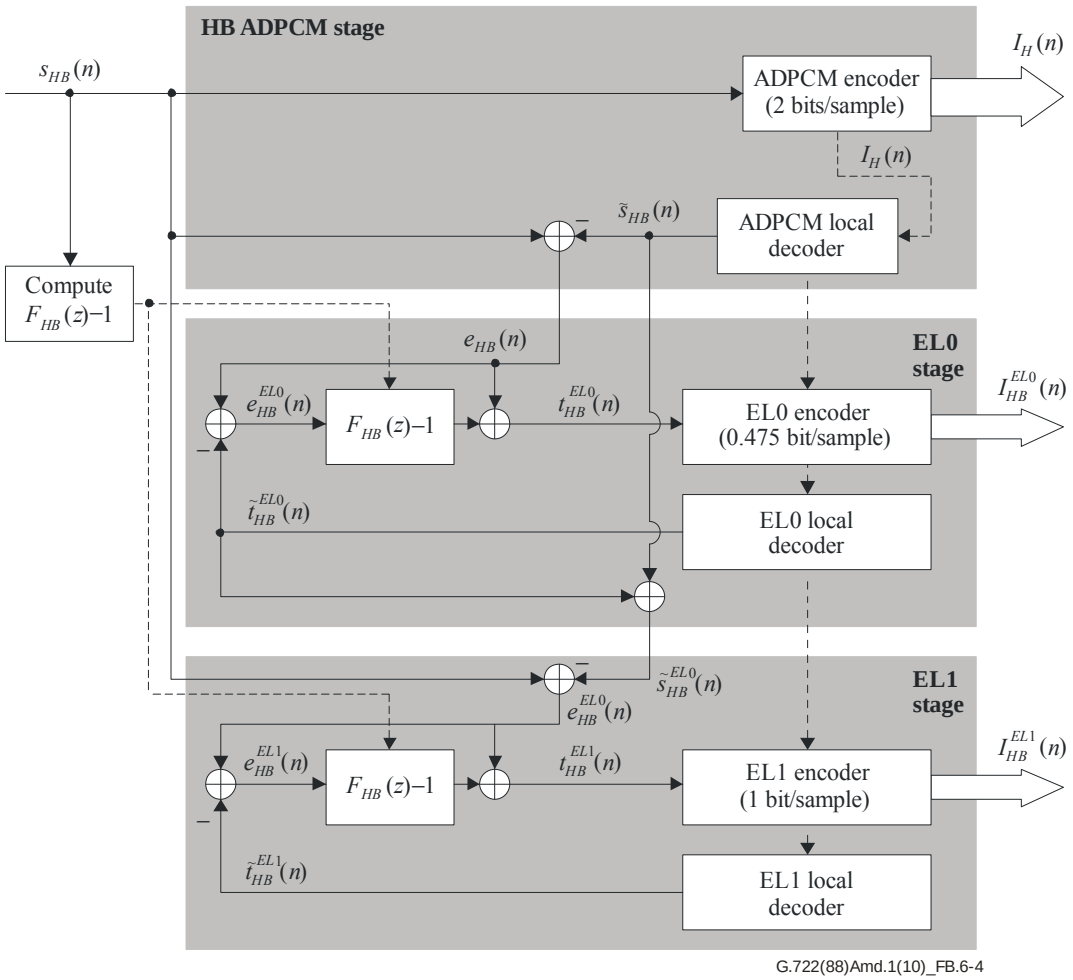


Figure B.6-4 – ITU-T G.722 HB encoder and enhanced ITU-T G.722 HB encoder

The ITU-T G.722 HB encoder corresponds to the legacy higher band adaptive differential pulse coded modulation (ADPCM) encoder described in clauses 3.2-3.6 of ITU-T G.722. This legacy ADPCM encoder operates at 16 kbit/s (2 bit/sample).

The ADPCM signal-to-noise ratio in the higher band is usually quite low due to the allocated bitrate of 2 bit/sample, as opposed to the bitrate of 5 or 6 bit/sample for ITU-T G.722 LB ADPCM coding. This difference in bitrate explains that:

– Contrary to the ITU-T G.722 lower band, no noise feedback is used in higher band with ADPCM coding. For noise feedback to be efficient, a bitrate higher than 2 bit/sample is desirable.

– If additional bandwidth is available to improve the quality of the ITU-T G.722 core encoder, extra bits are entirely allocated to the higher band. The quantization resolution can be improved by the enhanced ITU-T G.722 HB encoders, which rely on embedded scalar quantizers of 3 and 4 bit/sample extending the ITU-T G.722 HB scalar quantizer of 2 bit/sample.

The ITU-T G.722 HB ADPCM encoder is extended in a bitstream scalable fashion with two embedded stages. The first extension stage (G722EL0) operates at 3.8 kbit/s, where only 19 of the 40 samples are enhanced using one bit per selected sample; this forms the G722EL0 layer, which is used in all superwideband layers. Note that the G722EL0 layer is disabled in case of transient signal segments where the spared 19 bits are allocated to the bandwidth extension module described in clause B.6.6. The second extension stage (G722EL1) operates at 8 kbit/s (1 bit/sample) to further refine the quantization of the ITU-T G.722 higher band, forming the G722EL1 layer. This layer is only used in the highest bitrate mode G722R3sm.

B.6.4.3.1 ITU-T G.722 HB encoder (2 bit/sample)

The same as the higher band ADPCM encoder in clauses 3.2-3.6 of ITU-T G.722. For each sample *n* this encoding results in an index $I_H(n)$, using the same notation as in clauses 3.2-3.6 of ITU-T G.722.

From the index $I_H(n)$ the local decoder reconstructs:

$$\tilde{s}_{HB}(n) = d_H(n) + s_H(n) \qquad n = 0,...,39 \qquad \text{(B.6-35)}$$

where $d_H(n)$ is the quantized difference signal (obtained from the index $I_H(n)$) and $s_H(n)$ is the predicted signal, as defined in clauses 3.2-3.6 of ITU-T G.722.

B.6.4.3.2 Computation of noise-shaping filter $F_{HB}(z)-1$

Pre-emphasis and linear predictive analysis

The same as the corresponding section in clause B.6.4.2.1, except that the lower band input signal $s_{LB}(n)$ is replaced by the higher band input signal $s_{HB}(n)$.

This procedure results in the LP analysis giving a filter with the transfer function:

$$A_{HB}(z) = 1 + a_{HB1}z^{-1} + a_{HB2}z^{-2} + a_{HB3}z^{-3} + a_{HB4}z^{-4} \qquad \text{(B.6-36)}$$

where a_{HBi}, i=1,⋯,4, are the LP coefficients obtained from the higher band signal, similarly obtained as in the lower band (see clause B.6.4.2.1).

Derivation of noise shaping filter

The noise-shaping filter with transfer function

$$F_{HB}(z) = 1 + a'_{HB1}z^{-1} + a'_{HB2}z^{-2} + a'_{HB3}z^{-3} + a'_{HB4}z^{-4} \qquad \text{(B.6-37)}$$

is derived from $A_{HB}(z)$ as explained below and illustrated in Figure B.6-5 with *M*=4.

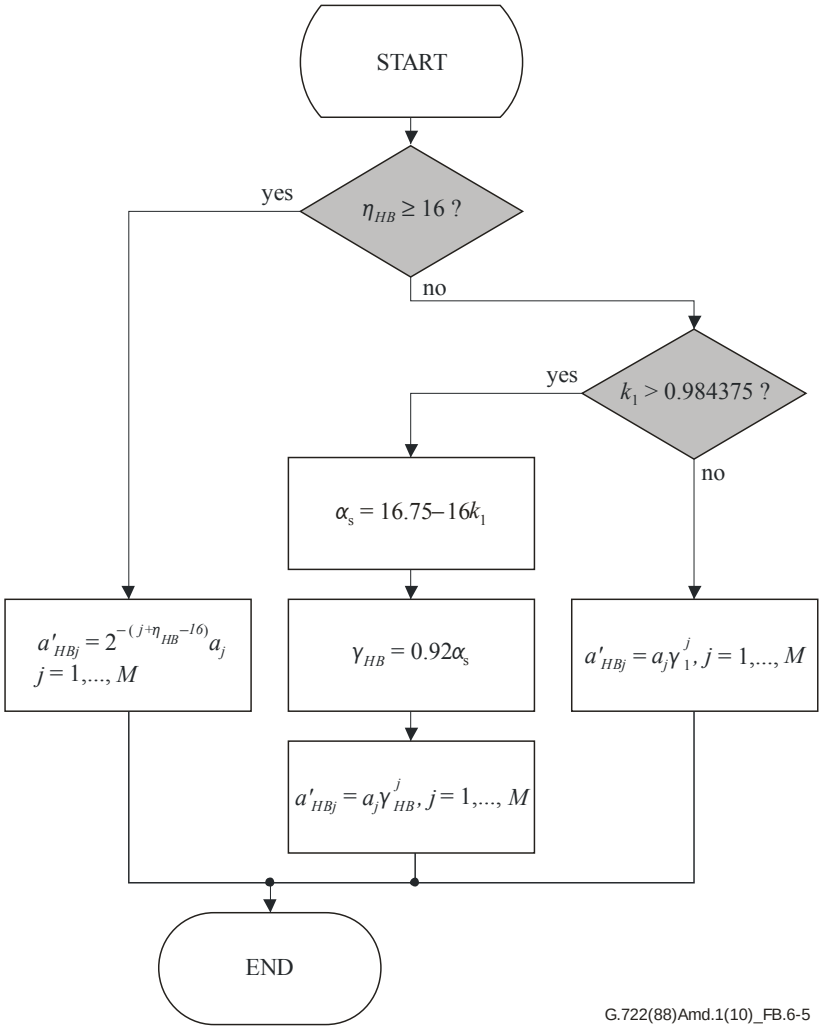


Figure B.6-5 – Computation of $F_{HB}(z)-1$ in ITU-T G.722 HB encoder

By default, $F_{HB}(z)$ is a weighted version of $A_{HB}(z)$, as follows: $F_{HB}(z) = A_{HB}(z/\gamma_1)$ with $\gamma_1 = 0.92$, i.e., $a'_{HBj} = \gamma_1^j a_{HBj}, j = 1,...,M$. This default operation is changed in the special case of low level input signal, and of signal with energy concentrated in higher frequencies. These two cases are detailed below.

Attenuation of the perceptual filter for signals with very low level

Same as corresponding clause in B.6.4.2.1.

Attenuation of the perceptual filter for signals with energy concentrated in higher frequencies

The noise-shaping filter is attenuated whenever a signal whose energy is concentrated in higher frequencies is detected in the encoder. To determine the spectral tilt of the signal, the reflection coefficient k_1 is used and the following condition must be fulfilled:

$$k_1 > 0.984375 \qquad \text{(B.6-38)}$$

In this case, the perceptual weighting filter is used with the weighting factor γ_{HB} defined by:

$$\gamma_{HB} = 0.92\alpha_s \qquad \text{(B.6-39)}$$

where the attenuation factor α_s is a function of k_1 given by:

$$\alpha_s = 16.75 - 16k_1 \qquad \text{(B.6-40)}$$

B.6.4.3.3 Encoding G722EL0 layer (0.475 bit/sample)

G.722EL0 layer is the higher band enhancement layer with 19 bits used only in non-TRANSIENT frames. Nineteen samples are selected for the higher band enhancement according to the available bits and the ITU-T G.722 higher band encoding. One bit is coded for each selected sample.

To obtain 19 more perceptual important samples, a moving average value $\bar{d}_H$ of the quantized difference signal d_H (defined in Table 3 of ITU-T G.722) is computed as follows:

$$\bar{d}_H(i) = \frac{1}{i}\sum_{n=0}^{i-1}\left|d_H(n)\right| \qquad i = 1,...,39 \qquad \text{(B.6-41)}$$

The higher band samples are selected according to the following conditions:

– $i = 0$, the first higher band sample is always selected and the number of the enhanced samples is set to 1; or

– $\left|d_H(i)\right| > \bar{d}_H(i), \quad i = 1,\ldots,39$ and the number of the enhanced samples is less than 19; or

– the number of non-enhanced samples is equal to 21.

The number of the enhanced samples is incremented when a sample is selected. If $i = 39$, the number of enhanced samples is set to zero and $\bar{d}_H(i)$ is also set to zero.

For each selected sample *n*, one bit is coded according to the target signal $t_{HB}^{EL0}(n)$, which is computed as follows:

$$t_{HB}^{EL0}(n) = e_{HB}(n) - \sum_{i=1}^{M} a_{HBi}\, e_{HB}^{EL0}(n-i) \qquad n = 0,...,39 \qquad \text{(B.6-42)}$$

where $e_{LB}^{EL0}(n-i)$ is the coding noise of the G722EL0 layer at the past sample $n-i$; see Equation (B.6-47) below.

The target sample $t_{HB}^{EL0}(n)$ is then quantized by minimizing the mean square error between $t_{HB}^{EL0}(n)$ and two possible quantized values:

$$\xi_{EL0}^{i}(n) = \Delta_{\mathrm{H}}(n)\left(Q3\left[2I_H(n)+i\right] - Q2\left[I_H(n)\right]\right) \qquad n = 0,...,39,\ i = 0,1 \qquad \text{(B.6-43)}$$

where $Q3[\cdot]$ and $Q2[\cdot]$ are respectively defined in Table B.6-3 and Table B.6-2. Again, $\Delta_H(n)$ is the step size of the ITU-T G.722 higher band ADPCM encoder given in Equation (3-18) of ITU-T G.722.

Table B.6-2 – ITU-T G.722 HB 2-bit normalized codebook

j	*Q2*[*j*]
0	–0.22607421875
1	–0.04931640625
2	0.22607421875
3	0.04931640625

Table B.6-3 – ITU-T G.722 HB 3-bit normalized codebook

j	*Q*3[*j*]
0	–0.293701171875
1	–0.182373046875
2	–0.09326171875
3	–0.026611328125
4	0.182373046875
5	0.293701171875
6	0.026611328125
7	0.09326171875

For each selected sample *n*, the analysis by synthesis consists of finding the bit $I_{HB}^{EL0}(n)$ of G722EL0 layer minimizing:

$$I_{HB}^{EL0}(n) = \arg\min_{i=0,1} \left| t_{HB}^{EL0}(n) - \xi_{EL0}^{i}(n) \right|^2 \qquad n = 0,...,39 \qquad \text{(B.6-44)}$$

This error minimization is equivalently realized by comparing $t_{HB}^{EL0}(n)$ to the corresponding decision threshold from Table B.6-4, where $T(I_H(n))$ is the mid value:

$$T(I_H(n)) = \frac{\left(Q3\left[2I_H(n)+1\right] - Q2\left[I_H(n)\right]\right) + \left(Q3\left[2I_H(n)\right] - Q2\left[I_H(n)\right]\right)}{2} \qquad n = 0,...,39 \qquad \text{(B.6-45)}$$

If $t_{HB}^{EL0}(n) > \Delta_H(n) \cdot T(I_H(n))$, $I_{HB}^{EL0}(n)$ is set to "1", otherwise it is set to "0". Then all the $I_{HB}^{EL0}(n)$ for the selected samples produce the G722EL0 bitstream. Table B.6-4 gives the thresholds $T(I_H(n))$.

Table B.6-4 – Thresholds for G722EL0

j	*T*[*j*]
0	–0.011963
1	–0.01062
2	0.011963
3	0.01062

The local decoder in the G722EL0 layer reconstructs:

$$\tilde{t}_{HB}^{EL0}(n) = \begin{cases} \xi_{EL0}^{I_{HB}^{EL0}(n)}(n), & \text{if } n \text{ is selected} \\ 0, & \text{otherwise} \end{cases} \qquad n = 0,...,39 \qquad \text{(B.6-46)}$$

in order to update the coding noise $e_{HB}^{EL0}(n)$ for the next sample:

$$e_{HB}^{EL0}(n) = e_{HB}(n) - \tilde{t}_{HB}^{EL0}(n) \qquad n = 0,...,39 \qquad \text{(B.6-47)}$$

In addition the local decoder in the G722EL0 layer reconstructs the enhanced quantized difference signal:

$$d_{HB}^{EL0}(n) = \begin{cases} Q3\left[2I_H(n) + I_{HB}^{EL0}(n)\right] \cdot \Delta_{\mathrm{H}}(n), & \text{if } n \text{ is selected} \\ Q2\left[I_H(n)\right] \cdot \Delta_{\mathrm{H}}(n), & \text{otherwise} \end{cases} \qquad n = 0,...,39 \qquad \text{(B.6-48)}$$

B.6.4.3.4 Encoding G722EL1 layer (1 bit/sample)

For each sample n, one bit is coded according to the target signal $t_{HB}^{EL1}(n)$, which is computed as follows:

$$t_{HB}^{EL1}(n) = e_{HB}^{EL0}(n) - \sum_{i=1}^{M} a_{HBi}\, e_{HB}^{EL1}(n-i) \qquad n = 0,...,39 \qquad \text{(B.6-49)}$$

where $e_{LB}^{EL1}(n-i)$ is the coding noise of the G722EL0 layer at the past sample $n-i$ (see Equation (B.6-55) below).

The target sample $t_{HB}^{EL1}(n)$ is then quantized by minimizing the mean square error between $t_{HB}^{EL1}(n)$ and two possible quantized values that depend on whether the sample *n* is selected or not in the G722EL0 layer.

If the sample *n* is selected in G722EL0 layer, the two possible quantized values are:

$$\xi_{EL1}^{i}(n) = \Delta_{\mathrm{H}}(n) \cdot \left(Q4\left[4I_H(n) + I_{HB}^{EL0}(n) + i\right] - Q3\left[2I_H(n) + I_{HB}^{EL0}(n)\right]\right) \qquad i = 0,1 \qquad \text{(B.6-50)}$$

where $Q4[\cdot]$ and $Q3[\cdot]$ are respectively defined in Table B.6-5 and Table B.6-3. Otherwise (if the sample *n* is not selected in EL0), the two possible quantized values are:

$$\xi_{EL1}^{i}(n) = \Delta_{\mathrm{H}}(n) \cdot \left(Q3\left[2I_H(n) + i\right] - Q2\left[I_H(n)\right]\right) \qquad i = 0,1 \qquad \text{(B.6-51)}$$

where $Q3[\cdot]$ and $Q2[\cdot]$ are respectively defined in Table B.6-3 and Table B.6-2.

Equivalently, the two possible quantized values are calculated using the enhanced quantized difference signal $d_{HB}^{EL0}(n)$ from the local decoder in G722EL0 layer:

$$\xi_{EL1}^{i}(n) = \begin{cases} \Delta_{\mathrm{H}}(n) \cdot Q4\left[4I_H(n) + I_{HB}^{EL0}(n) + i\right] - d_{HB}^{EL0}(n), & \text{if } n \text{ is selected in EL0} \\ \Delta_{\mathrm{H}}(n) \cdot Q3\left[2I_H(n) + i\right] - d_{HB}^{EL0}(n), & \text{otherwise} \end{cases} \qquad i = 0,1 \qquad \text{(B.6-52)}$$

Table B.6-5 – ITU-T G.722 HB 4-bit normalized codebook

j	Q4[j]
0	–0.444091796875
1	–0.267578125
2	–0.20849609375
3	–0.160400390625
4	–0.115234375
5	–0.07666015625
6	–0.043212890625
7	–0.013427734375

Table B.6-5 – ITU-T G.722 HB 4-bit normalized codebook

j	Q4[j]
8	0.160400390625
9	0.20849609375
10	0.267578125
11	0.444091796875
12	0.013427734375
13	0.043212890625
14	0.07666015625
15	0.115234375

The analysis by synthesis consists in finding the bit $I_{HB}^{EL1}(n)$ of the G722EL1 layer with the following minimization:

$$I_{HB}^{EL1}(n) = \arg\min_{i=0,1} \left| t_{HB}^{EL1}(n) - \xi_{EL1}^{i}(n) \right|^2 \qquad n = 0,...,39 \qquad \text{(B.6-53)}$$

The local decoder in the G722EL1 layer reconstructs:

$$\tilde{t}_{HB}^{EL1}(n) = \xi_{EL1}^{I_{HB}^{EL1}(n)}(n) \qquad n = 0,...,39 \qquad \text{(B.6-54)}$$

in order to update the coding noise $e_{HB}^{EL1}(n)$ for the next sample:

$$e_{HB}^{EL1}(n) = e_{HB}(n) - \tilde{t}_{HB}^{EL1}(n) \qquad n = 0,...,39 \qquad \text{(B.6-55)}$$

B.6.5 MDCT

The super higher band signals $s_{SHB}^{fold}(n)$ is transformed into frequency domain by modified discrete cosine transform (MDCT) with a frame length of 5 ms and an analysis window of 10 ms length. The MDCT coefficients $S_{SHB}(k)$ of the signal $s_{SHB}^{fold}(n)$ are given by:

$$S_{SHB}(k) = \sqrt{\frac{2}{80}} \sum_{n=0}^{159} w_{TDAC}(n) \cos\left(\frac{\pi}{80}(n+40.5)(k+0.5) \right) s_{SHB}^{fold}(n) \qquad k = 0,...,79 \qquad \text{(B.6-56)}$$

where $w_{TDAC}(n)$ is the analysis weighting window given by:

$$w_{TDAC}(n) = \sin\left(\frac{\pi}{160}(n+0.5) \right) \qquad n = 0,...,159 \qquad \text{(B.6-57)}$$

B.6.6 SWBL0 layer encoder

This layer is encoded with bandwidth extension (BWE) algorithm based on the adaptive spectral envelope coding and time envelope coding. The bit budget of 40 bits is shared with HBE module of ITU-T G.722 core, depending on whether a frame is a TRANSIENT (TS) frame or a non-TRANSIENT frame. If the input super higher band signal of the previous frame or the current frame is detected as TRANSIENT, then the current frame is classified as TRANSIENT. In case of a TRANSIENT frame, all 40 bits are allocated to BWE which includes two bits for the signal class, and 38 bits for encoding the global gain, four spectral envelopes and four time envelopes. For other cases (non-TRANSIENT frame), only 21 bits are allocated to BWE which also includes two bits for the signal class and 19 bits for encoding the global gain and eight spectral envelopes. In this case,

no time envelope is encoded. The remaining 19 bits are allocated to HBE (see clause B.6.4.3.3). Non-TRANSIENT frames can be further classified as HARMONIC (HM), NORMAL (NM) or NOISE (NS) according to the frequency fluctuation. For BWE encoding, the first 64 (out of 80) MDCT coefficients of super higher band (SHB), $S_{SHB}(k)$, k=0,⋯,63, are coded. The last 16 MDCT coefficients, $S_{SHB}(k)$, k=64,⋯,79, that are associated with the frequency range between 14.4-16 kHz are discarded.

B.6.6.1 Time envelope calculation and transient detection

The transient detection uses parameters computed from time envelopes of three consecutive frames, the current frame and the previous two frames.

The time envelope which represents the temporal energy of SHB signal is computed as a set of root-mean square (RMS) calculated from 20 samples of time-domain folded SHB signal. This results in four time envelope coefficients per frame.

$$t_{rms}(j+8) = \frac{1}{2}\log_2\left(\frac{1}{20}\sum_{n=0}^{19} s_{SHB}^{fold}(20j+n)^2\right) \qquad j = 0,\cdots,3 \qquad \text{(B.6-58)}$$

where $t_{rms}(j), \quad j = 0,...,7$, are the time envelopes of the two previous frames. It results in 12 time envelopes corresponding to 12 time sub-frames. For the first frame, the $t_{rms}(j), \quad j = 0,...,7$ are set to zero.

The total energy of the current frame is calculated as:

$$t_{en}(2) = \sum_{n=0}^{79} s_{SHB}^{fold}(n)^2 \qquad \text{(B.6-59)}$$

with $t_{en}(0)$ and $t_{en}(1)$ being the total energies of the previous two frames. For the first frame, $t_{en}(0)$ and $t_{en}(1)$ are set to zero.

Three time parameters are calculated to perform the transient detection:

– the total RMS of the time envelopes t_{rms_total}, for three consecutive frames:

$$t_{rms\_total} = \frac{1}{2}\log_2\left(\frac{\sum_{j=0}^{2} t_{en}(j)}{240} + \varepsilon_{rms}\right) \qquad \text{(B.6-60)}$$

where: $\varepsilon_{rms} = 10^{-3}$.

– the maximum difference between the consecutive time envelopes d_{tenv},

$$d_{tenv} = \max_{j=0,...,10}(t_{rms}(j+1) - t_{rms}(j)) \qquad \text{(B.6-61)}$$

– the maximum difference between the time envelopes and the total RMS of the time envelopes d_{tenv_max},

$$d_{tenv\_max} = \max_{j=0,...,11}(t_{rms}(j) - t_{rms\_total}) \qquad \text{(B.6-62)}$$

If three conditions, $d_{tenv\_max} > 3.3$, $d_{tenv} > 2.4$ and $t_{rms\_total} > 8$ are fulfilled, the current frame is identified as a TRANSIENT frame. The signal class F_{class} is set to TRANSIENT ($F_{class} = TS$).

The sub-frame index with the maximum time envelope is denoted as $idx_{tenv\_\max}$, with $idx_{tenv\_\max} = \arg\max_{j=0,\ldots,11}\left(t_{rms}(j)\right)$. When the current frame or the previous frame is classified as TRANSIENT, a time envelope adjustment is performed based on two parameters, r_{avg1} and r_{avg2}. These parameters are the time envelope ratio between the maximum time envelope and the average time envelope of the sub-frames before and after the sub-frame $idx_{tenv\_\max}$ respectively,

$$r_{avg1} = \begin{cases} \dfrac{t_{rms}(idx_{tenv\_\max}) \cdot idx_{tenv\_\max}}{\sum_{j=0}^{idx_{tenv\_\max}-1} t_{rms}(j)}, & \text{if } idx_{tenv\_\max} \geq 4 \\ 0, & \text{otherwise} \end{cases} \tag{B.6-63}$$

and

$$r_{avg2} = \begin{cases} \dfrac{t_{rms}(idx_{tenv\_\max}) \cdot (11 - idx_{tenv\_\max})}{\sum_{j=idx_{tenv\_\max}+1}^{11} t_{rms}(j)}, & \text{if } 4 \leq idx_{tenv\_\max} < 8 \\ 0, & \text{otherwise} \end{cases} \tag{B.6-64}$$

If F_{class} = *TS*, the time envelopes to be encoded at current frame, $\bar{t}_{rms}(j)$, $0 \leq j \leq 3$, are set to the previous frame time envelopes with an adjustment based on r_{avg1}, r_{avg2}, and $idx_{tenv\_\max}$:

$$\bar{t}_{rms}(j) = \begin{cases} t_{rms}(j+4) + 0.5, & \text{if } j+4 = idx_{tenv\_\max} \text{ and } r_{avg1} > 2 \text{ and } r_{avg2} > 2 \\ t_{rms}(j+4) - 1.0, & \text{else if } j+4 < idx_{tenv\_\max} \\ t_{rms}(j+4), & \text{otherwise} \end{cases} \tag{B.6-65}$$

Otherwise ($F_{class} \neq$ TS), the time envelopes to be encoded at current frame, $\bar{t}_{rms}(j)$, $0 \leq j \leq 3$, are set to the previous frame time envelopes without adjustment:

$$\bar{t}_{rms}(j) = t_{rms}(j+4) \qquad j = 0, \cdots, 3 \tag{B.6-66}$$

The obtained time envelopes $\bar{t}_{rms}(j)$, $0 \leq j \leq 3$, are further bounded in the range [0,...,15]: $0 \leq \bar{t}_{rms}(j) \leq 15$, $0 \leq j \leq 3$. The rounded time envelopes $t'_{rms}(j) = round(\bar{t}_{rms}(j))$ are quantized into four bits by uniform scalar quantization in case of TRANSIENT frames.

To further reduce the pre-echo for the transients, additional time envelope adjustment is performed at the decoder side for the sub-frame with the maximum time envelope $\bar{t}_{rms}(idx_{trans})$, with $idx_{trans} = \arg\max_{j=0,\ldots,3}\left(\bar{t}_{rms}(j)\right)$. The time envelope adjustment flag bit, F_{tenv}, is set at the encoder side according to e_0 and e_1, the energies of the first half and the second half of this maximum time envelope sub-frame:

$$\begin{cases} F_{tenv} = 1, & \text{if } \quad e_1 > e_0 \\ F_{tenv} = 0, & \text{otherwise} \end{cases} \tag{B.6-67}$$

where $e_0 = \sum_{n=0}^{9} s_{SHB}^{fold(-1)}(20 \cdot idx_{trans} + n)$, $e_1 = \sum_{n=0}^{9} s_{SHB}^{fold(-1)}(20 \cdot idx_{trans} + 10 + n)$ and $s_{SHB}^{fold(-1)}(n)$ being the folded super higher band signal of the previous frame.

The following buffers are updated after the time envelope calculation and transient detection,

$$t_{rms}(j) = t_{rms}(j+4) \qquad j = 0,\cdots,7 \qquad \text{(B.6-68)}$$

$$t_{en}(k) = t_{en}(k+1) \qquad k = 0,1 \qquad \text{(B.6-69)}$$

Finally, if the current frame is not classified as TRANSIENT and the saved signal class for the previous frame is TRANSIENT, the current frame is also identified as a TRANSIENT frame. Otherwise, it is identified as a non-TRANSIENT frame.

B.6.6.2 Global gain

The global gain g_{glob} is calculated in the frequency domain and encoded using five bits by uniform scalar quantization.

$$g_{glob} = round\left(\log_2 \sqrt{\frac{1}{64}\left(\sum_{k=0}^{63} S_{SHB}(k)^2 + \varepsilon_{rms} \right)} \right) \qquad \text{(B.6-70)}$$

where g_{glob} is further bounded in the range [0,...,31].

The global gain g_{glob} is locally converted into the linear domain as follows:

$$\hat{g}_{glob} = 2^{g_{glob}} \qquad \text{(B.6-71)}$$

The locally decoded (converted) global gain of the previous frame $\hat{g}_{glob}^{(-1)}$ is saved for frequency sharpness measurement in clause B.6.6.6.

B.6.6.3 Sub-band division

The 64 MDCT coefficients in the 8000-14400 Hz frequency range are split into four sub-bands for TRANSIENT frames (16 coefficients per sub-band) or eight sub-bands for non-TRANSIENT frames (eight coefficients per sub-band). Table B.6-6 and Table B.6-7 define the sub-band boundaries and size for TRANSIENT frames and non-TRANSIENT frames respectively. The *j*-th sub-band comprises $N_{swbcf}(j)$ coefficients $S_{SHB}(k)$ with $b_{swb}(j) \le k < b_{swb}(j+1)$.

Table B.6-6 – Sub-band boundaries and number of coefficients per sub-band in BWE (TRANSIENT frame)

j	$b_{swb}(j)$	$N_{swbcf}(j)$
0	0	16
1	16	16
2	32	16
3	48	16
4	64	–

Table B.6-7 – Sub-band boundaries and number of coefficients per sub-band in BWE (Non-TRANSIENT frame)

j	$b_{swb}(j)$	$N_{swbcf}(j)$
0	0	8
1	8	8
2	16	8
3	24	8
4	32	8
5	40	8
6	48	8
7	56	8
8	64	–

B.6.6.4 Normalized spectral envelope calculation

If the current frame is a TRANSIENT frame, the normalized spectral envelope is computed with four RMS coefficients calculated using 16 samples of frequency-domain SHB signal each. For a non-TRANSIENT frame, the normalized spectral envelope consists of eight RMS coefficients calculated using eight frequency-domain samples each. The normalized spectral envelope is defined as the RMS of the sub-bands:

$$f_{rms}(j) = \frac{1}{\hat{g}_{glob}} \sqrt{\frac{1}{N_{swbcf}(j)} \left(\sum_{k=b_{swb}(j)}^{b_{swb}(j+1)-1} S_{SHB}{}^2(k) + \varepsilon_{rms} \right)} \qquad \text{(B.6-72)}$$

where $j = 0,\cdots,3$ for TRANSIENT frame and $j = 0,\cdots,7$ for non-TRANSIENT frame.

B.6.6.5 Normalized spectral envelope coding

In case of TRANSIENT frame, the normalized spectral envelope is multiplied by five and rounded to the nearest integer with a ceiling of 15. Then, the spectral envelope index $f_{rms\_idx\_t}(j)$ is calculated as follows:

$$f_{rms\_idx\_t}(j) = \min(round(5 f_{rms}(j)),15) \qquad j = 0,\cdots,3 \qquad \text{(B.6-73)}$$

The obtained spectral envelope index $f_{rms\_idx\_t}(j)$ is quantized with four bits per sub-band using a uniform scalar quantization.

For non-TRANSIENT frames, the lower four sub-bands and the higher four sub-bands are quantized into six bits using a 4-dimensional vector quantizer. For this vector quantization, two sets of codebook of size 64 are available. If either condition, $\sum_{j=0}^{3} f_{rms}{}^2(j) > 1.69$ or $\sum_{j=4}^{7} f_{rms}{}^2(j) > 1.69$ is fulfilled, the first codebook, cb_{fenv1}, is used. Otherwise, the second codebook, cb_{fenv2}, is used. Two flag bits, $F_{cb}(i), i = 0,1$, are transmitted to identify the selected codebooks for the lower four sub-bands and the higher four sub-bands, respectively.

The normalized spectral envelope is locally decoded as $\hat{f}_{rms}(j)$ which will be used to sub-band ordering by perceptual importance (see clause B.6.7.2).

– If the current frame is TRANSIENT, the normalized spectral envelope $\hat{f}_{rms}(j)$ is decoded as:

$$\hat{f}_{rms}(j) = 0.2 f_{rms\_idx\_t}(j) \qquad j = 0,\cdots,3 \qquad \text{(B.6-74)}$$

– If the current frame is non-TRANSIENT, the normalized spectral envelope $\hat{f}_{rms}(j)$ is decoded as follows:

$$\hat{f}_{rms}(j+4i) = \begin{cases} cb_{fenv1}(4 f_{rms\_idx\_nt}(i) + j), & \text{if } F_{cb}(i) = 1 \\ cb_{fenv2}(4 f_{rms\_idx\_nt}(i) + j), & \text{otherwise} \end{cases} \qquad \begin{matrix} j = 0,\cdots,3 \\ i = 0,1 \end{matrix} \qquad \text{(B.6-75)}$$

where $f_{rms\_idx\_nt}(j)$, i=0, 1 is the vector quantization index.

B.6.6.6 Frequency sharpness

For non-TRANSIENT frames, frequency sharpness is computed to measure the spectrum fluctuation of the frequency coefficients of the super higher band signal and those frames are categorized in three classes:

a) HARMONIC: if frequency sharpness is high.

b) NOISE: if frequency sharpness is low.

c) NORMAL: if frequency sharpness is moderate.

The first 60 MDCT coefficients in the 8000-14000 Hz frequency range are split into ten sharpness bands (six coefficients per band). The frequency sharpness, $\kappa(j)$, is defined as the ratio between peak magnitude and average magnitude in a sharpness band:

$$\kappa(j) = \begin{cases} \dfrac{5 \cdot A_{sharp}(j)}{\sum_{k=6j}^{6j+5} |S_{SHB}(k)| - A_{sharp}(j)} & \text{if } \sum_{k=6j}^{6j+5} |S_{SHB}(k)| \neq 0 \\ 0, & \text{otherwise} \end{cases} \qquad j = 0,\cdots,9 \qquad \text{(B.6-76)}$$

where the maximum magnitude of spectral coefficients in a sharpness band, denoted $A_{sharp}(j)$, is:

$$A_{sharp}(j) = \max_{k=6j,\ldots,6j+5} |S_{SHB}(k)| \qquad j = 0,\ldots 9 \qquad \text{(B.6-77)}$$

Then, three further sharpness parameters are determined: the maximum sharpness, $\kappa_{\max}$, and two counters, the sharpness band counter, c_{sharp}, and the noise band counter, c_{noise}.

The maximum sharpness, $\kappa_{\max}$, in all sharpness bands is computed as:

$$\kappa_{\max} = \max_{j=0,\ldots,9} \left(A_{sharp}(j) \right) \qquad \text{(B.6-78)}$$

The counter c_{sharp} is computed from the ten frequency sharpness parameters, $\kappa(j)$, and from the ten maximum magnitudes, $A_{sharp}(j)$, as follows: initialized to zero, c_{sharp} is incremented by one for each j, $j = 0,\ldots 9$, if $\kappa(j) > 4$ and $A_{sharp}(j) > 10$.

The counter c_{noise} is computed from the ten frequency sharpness parameters $\kappa(j)$ as follows: initialized to zero, c_{noise} is incremented by one for each j, $j = 0,\ldots 9$, if $\kappa(j)$ is less than 2.5.

The class of non-TRANSIENT frames is determined from these three sharpness parameters, $\kappa_{\max}$, c_{sharp}, and c_{noise} and two other paremeters – the previous frame saved class, $F_{class}^{(-1)}$, and the ratio between the locally decoded global gains of the current and previous frames.

The initial sharpness i_{sharp} is set according to the saved signal class of previous frame $F_{class}^{(-1)}$:

$$i_{sharp} = \begin{cases} 4, & \text{if} \quad F_{class}^{(-1)} = HARMONIC \\ 7, & \text{else if} \quad F_{class}^{(-1)} = TRANSIENT \\ 5, & \text{otherwise} \end{cases} \tag{B.6-79}$$

– If $c_{sharp} \geq i_{sharp}$ and $0.5 < \frac{\hat{g}_{glob}}{\hat{g}_{glob}^{(-1)}} < 1.8$, the current frame is classified as HARMONIC frame (F_{class} = HARMONIC) and the counter for signal class c_{class} is incremented by one when c_{class} is less than eight. Otherwise, c_{class} is decremented by one when c_{class} is larger than zero.

– Then, if $c_{class} \geq 2$, the current frame is also classified as HARMONIC frame (F_{class} = HARMONIC).

– Finally, if the saved previous frame signal class $F_{class}^{(-1)}$ was already HARMONIC, the current frame is also classified as HARMONIC.

– For other cases, depending on the noise counter c_{noise} and the maximum sharpness $\kappa_{\max}$, the current frame is classified as NORMAL or NOISE: if $c_{noise} > 6$ and $\kappa_{\max} < 3.5$, the current frame is classified as NOISE frame (F_{class} = NOISE), otherwise, the current frame is classified as NORMAL frame (F_{class} = NORMAL).

B.6.6.7 SHB signal class coding and saving

Two bits are transmitted for SHB signal class coding. Table B.6-8 gives the coded bits for each class.

Table B.6-8 – SHB signal class coding

Signal class F_{class}	Coded bits
NORMAL	00
NOISE	01
HARMONIC	10
TRANSIENT	11

The signal class F_{class} of the current frame is saved in $F_{class}^{(-1)}$ for the next frame. Nevertheless, to get accurate classification for the next frame, the saved signal class for the next frame, $F_{class}^{(-1)}$, is set to NORMAL in the following cases:

– If the current frame is not TRANSIENT and the saved signal class of the previous frame is TRANSIENT, then the saved signal class is NORMAL.

– If the current frame is not HARMONIC and the saved signal class of the previous frame is HARMONIC, then the saved signal class is NORMAL.

Then, the signal class is finally determined.

B.6.6.8 Bit allocation for BWE

Table B.6-9 illustrates the BWE bit allocation for TRANSIENT and non-TRANSIENT frames.

Table B.6-9 – BWE bit allocation

Signal class	Signal class bits	Time envelope	Time envelope adjustment flag	Global gain	Spectral envelope	Total bits
Parameters	F_{class}	$\bar{t}_{rms}(j)$	F_{tenv}	g_{glob}	$f_{rms}(j)$	
TRANSIENT	2	16 (=4×4)	1	5	16 (=4×4)	40
Non-TRANSIENT	2	0	0	5	14 (=2×(6+1))	21

B.6.7 SWBL1 and SWBL2 layer encoder

While SWBL0 only transmits the spectral (and time) envelope of the SHB signal, SWBL1 and SWBL2 layers convey the fine structure (or "excitation" as in the previous clause) of the SHB MDCT coefficients $S_{SHB}(k)$, k=0,⋯,63. The same sub-band division as performed in clause B.6.6.3 in case of non-TRANSIENT frame is performed: the 64 MDCT coefficients are divided into eight sub-bands with eight coefficients in each sub-band, normalized and quantized using the algebraic vector quantization (AVQ). The SWBL1 and SWBL2 bit-budget does not allow encoding all sub-bands using the AVQ, thus the spectrum in one or more sub-bands is derived from the BWE spectrum or other (AVQ coded) sub-bands. The sub-bands where AVQ is not applied are called "zero sub-bands" as AVQ output vector is zero for these sub-bands.

The SWBL1 is encoded using 40 bits per frame where one bit is used to indicate the SHB coding mode and three bits are used to refine the global gain coded in SWBL0. The remaining 36 bits can be used by the AVQ. The actual bit-budget needed to encode AVQ parameters varies from frame to frame and the difference between the allocated 36 bits and the number of spent bits is further called "Unused AVQ bits". The unused AVQ bits are further used to refine the zero sub-bands.

In SWBL2 all available bits (40 bits per frame) are allocated for the AVQ. Similarly to SWBL1, if there are any "Unused AVQ bits", they are used to refine the zero sub-bands.

B.6.7.1 Spectrum normalization

The first step in encoding the SHB signal in MDCT domain $S_{SHB}(k)$, $k = 0,\cdots,63$, is the normalization. The local decoded global gain, $\hat{g}_{glob}$, computed and transmitted in SWBL0 is used to obtain the normalized spectrum

$$S_{SHB}^{norm}(k) = S_{SHB}(k) / \hat{g}_{glob}, \quad k = 0,...,63 \tag{B.6-80}$$

B.6.7.2 Sub-band ordering by perceptual importance

The perceptual importance of the eight sub-bands, $ip(j)$, $j = 0,\ldots,7$, are simply estimated from the locally decoded normalized spectral envelope $\hat{f}_{rms}(j)$ (see clause B.6.6.5, Equations (B.6-74) and (B.6-75)):

$$\begin{aligned} &\text{If } F_{class}\text{= TRANSIENT} && ip(2j) = ip(2j+1) = \hat{f}_{rms}(j), && j = 0,\ldots,3 \\ &\text{Otherwise} && ip(j) = \hat{f}_{rms}(j), && j = 0,\ldots,7 \end{aligned} \tag{B.6-81}$$

The sub-bands are then sorted in order of descending perceptual importance. This results for each sub-band in an index $\Omega_b(j)$, $0 \le \Omega_b(j) < 8$, $j = 0,\ldots,7$ which indicates that $\Omega_b(j)$-th sub-band has the (j+1)-th largest perceptual importance (i.e., $\Omega_b(0)$-th sub-band has the highest perceptual

importance whereas $\Omega_b(7)$-th sub-band has the lowest perceptual importance). This ranking is used for choosing the most perceptually important sub-bands to be coded in SWBL1 while the less perceptually important sub-bands will be coded in SWBL2 for algebraic vector quantization (see clause B.6.7.4).

B.6.7.3 Selection of the SHB mode

The encoder for SWBL1 and SWBL2 layers classifies the super-highband signal into three encoding modes, SHB modes 0, 1 and 2. The first two modes are selected on the condition that the previous and the current frames are classified either NORMAL or NOISE. For frames that do not satisfy this condition, SHB mode 2 is selected. In SHB modes 0 and 1, one bit is transmitted as f_{shb_mode}, indicating either SHB mode 0 or 1. For SHB mode 2, no bit is transmitted. The distinction between SHB mode 0 and 1 is whether the coefficients are sparse or not, respectively, and the SHB mode flag $f_{shb\_mode}^{(m)}$ in the current frame *m* is computed as:

$$f_{shb\_mode}^{(m)} = \begin{cases} 1, & \text{if } \varphi_{shb}^{(m)} \le 15 \\ f_{shb\_mode}^{(m-1)}, & \text{if } 15 < \varphi_{shb}^{(m)} < 20 \\ 0, & \text{otherwise} \end{cases} \tag{B.6-82}$$

where $\varphi_{shb}^{(m)}$ is the sparseness parameter. The sparseness parameter of the current frame *m*, $\varphi_{shb}^{(m)}$, is smoothed across the pervious frame as:

$$\varphi_{shb}^{(m)} = 0.3c_{shb} + 0.7\varphi_{shb}^{(m-1)} \tag{B.6-83}$$

where $\varphi_{shb}^{(m-1)}$ is the parameter in the previous frame. Here, c_{shb} is computed as,

$$\begin{aligned} c_{shb} &= \sum_{k=0}^{63} f_{sparse}(k), \\ f_{sparse}(k) &= \begin{cases} 1 & \text{if } \left|S_{SHB}^{norm}(k)\right| < 0.5 \text{ and } \Omega_b(\lfloor k/8 \rfloor) < 4 \\ 0 & \text{otherwise} \end{cases}, \quad k = 0,\ldots,63 \end{aligned} \tag{B.6-84}$$

where $f_{sparse}(k)$ is a flag indicating if the amplitude of the normalized coefficient *k* is lower than the threshold 0.5. For the first decoded frame, $\varphi_{shb}^{(-1)}$ is initialized to 0.

B.6.7.4 Encoding the SHB in mode 0

The encoding in SHB mode 0 is based on quantization of the normalized spectrum $S_{SHB}^{norm}(k)$ using the AVQ. Before performing the AVQ, the spectrum is normalized and ordered per sub-bands using the perceptual importance ordering, $\Omega_b(j)$, determined in clause B.6.7.2.

Therefore, the spectrum to be quantized is computed in one step as:

$$S'(8j+i) = \frac{S_{SHB}^{norm}\left(8\Omega_b(j)+i\right)}{\beta_{avq} \cdot ip\left(\Omega_b(j)\right)}, \quad j = 0,\ldots 7, i = 0,\ldots 7. \tag{B.6-85}$$

where $\beta_{avq} = 10^{-3}$ is a constant for dealing with low energy MDCT coefficients.

The normalization by the normalized spectral envelope represented by $ip(j)$ makes the spectrum flat as much as possible. The AVQ is then able to encode more sub-bands because the AVQ codebook number (see later in clause B.6.7.10) differs from sub-band to sub-band less than in the case of non-normalized spectrum. Consequently, the number of zero sub-bands is reduced.

The AVQ coding is done in two steps that correspond to the encoding of the SWBL1 content and SWBL2 content. Giving the available bit-budget allocated for the AVQ (36 bits in SWBL1 and 40 bits in SWBL2), the AVQ is able to encode maximally three sub-bands in SWBL1 and four sub-bands in SWBL2. Thus at least one sub-band remains the zero sub-band and is dealt with differently. The signal flow in SHB mode 0 is shown in Figure B.6-6.

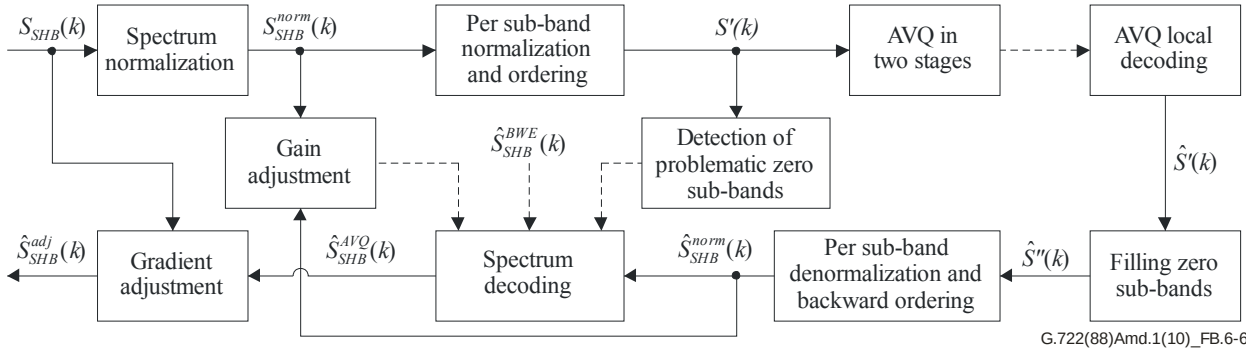


Figure B.6-6 – SHB mode 0 block diagram

The spectrum $S'(8j+i)$ contains coefficients to be quantized with the most perceptually important sub-band corresponding to $j = 0$ and the least perceptually important sub-band corresponding to $j = 7$. The AVQ in SWBL1 quantizes the first three ranked sub-bands ($j = 0, 1, 2$) as described in clause B.6.7.10.

B.6.7.4.1 Local decoding of MDCT coefficients

The AVQ computed in SWBL1 returns three 8-dimensional quantized sub-bands $\hat{\mathbf{S}}'(8j)$, $j = 0, 1, 2$. If none of these sub-bands are AVQ zero sub-bands (i.e., none of quantized sub-bands contains only zero coefficients), the input spectrum for the SWBL2 AVQ consists of $\hat{\mathbf{S}}'(8j)$, $j = 3, 4, 5, 6$. If one or two SWBL1 output sub-bands are AVQ zero sub-bands, these AVQ zero sub-bands are placed at the first positions of the input spectrum for the SWBL2 AVQ encoding. The AVQ encoding used in SWBL2 is the same as described in clause B.6.7.10 with only one difference: the number of sub-bands to be quantized is four.

The AVQ computed in SWBL2 returns four quantized sub-bands that are joined to the quantized coefficients from SWBL1 and form the AVQ locally decoded spectrum $\hat{S}'(k)$. The remaining sub-bands of the spectrum $\hat{S}'(k)$ that are not coded using the AVQ neither in SWBL1 nor SWBL2 are replaced by zero MDCT coefficients and form the zero sub-bands.

To form the full SHB spectrum, the coefficients in AVQ zero sub-bands need to be determined as well. They are derived either from the SWBL0 output spectrum or from the other AVQ coded coefficients. This is dealt with in the next clauses.

B.6.7.4.2 Detection of problematic zero sub-bands

The AVQ quantizes the most perceptually important sub-bands and one of the techniques to represent the coefficients of zero sub-bands is to derive them from the SWBL0 output spectrum. However, there are signals for which the reconstruction at the decoder is likely to be very inaccurate in certain sub-bands. These sub-bands are called problematic zero sub-bands and need to be detected. The detector is based on detection of zero sub-bands where the spectral envelope is not quantized too close to its original, basically due to the high quantization error in SWBL0 spectral envelope quantization. At the same time, a distribution of energy in zero sub-bands of the spectrum $S'(8j+i)$ is tested. The following assumption is used: If a sub-band contains a peak (the energy of the maximum sample in the sub-band is substantial compared to the average energy in this

sub-band), the coding of such sub-band is usually covered by the AVQ. But if this sub-band is not coded by the AVQ (i.e., it is a zero sub-band) and the AVQ selects other sub-bands (usually with peaks) to be encoded, this zero sub-band has a low importance. If there is a high number of such zero sub-bands, it is advantageous at the decoder to fill the zero sub-bands with significantly restrained spectral envelope. The detection of problematic zero sub-bands is performed only when both SWBL1 and SWBL2 are transmitted.

The detection itself relies on the value of the counter c_{det}, $c_{det} = 0,\cdots,20$, that is updated on a frame basis. If the counter $c_{det} > 0$, the detection flag for the current frame is $f_0 = 1$, otherwise it is $f_0 = 0$. The value of the counter c_{det} in the current frame depends on its value in the previous frame, on the SHB mode and also on two detection sub-flags $f_{0,s1}$ and $f_{0,s2}$. The sub-flag $f_{0,s1}$ value can be 0 or 1 and depends on the detection of the problematic zero sub-bands where the spectral envelope is poorly quantized. The following ratio is computed for all sub-bands

$$rat(j) = \frac{\hat{f}_{rms}(j) - f_{rms}(j)}{f_{rms}(j)}, \quad j = 0,..,7 \tag{B.6-86}$$

for non-TRANSIENT frames

$$rat(j) = \frac{\hat{f}_{rms}\left(\lfloor j/2 \rfloor\right) - f_{rms}\left(\lfloor j/2 \rfloor\right)}{f_{rms}\left(\lfloor j/2 \rfloor\right)}, \quad j = 0,..,7 \tag{B.6-87}$$

for TRANSIENT frames, where $f_{rms}(j)$ is the normalized spectral envelope and $\hat{f}_{rms}$ is its quantized representation known from SWBL0. Then a maximum ratio rat_{max} is searched within zero sub-bands. If $rat_{max} > 4$, $f_{0,s1} = 1$, otherwise $f_{0,s1} = 0$.

The value of sub-flag $f_{0,s2}$ can be 0, 1 or 2 and depends on the energy distribution in zero sub-bands of spectrum $S'(k)$. Initially it is set to $f_{0,s2} = 0$. Then, energy $E_{\max}$ of the maximum energy coefficient is searched and compared to the average energy E_{avg} of all coefficients in each zero sub-band. If $E_{\max} > 6E_{\text{avg}}$, then $f_{0,s2} = 2$. If there are at least 5 zero sub-bands and $E_{\max} > 4E_{\text{avg}}$ at least in one zero sub-band, then $f_{0,s2} = 1$. The sub-flag value is computed until it holds $f_{0,s2} = 2$ or all zero sub-bands are searched.

Finally the update of the detection counter c_{det} is performed as shown in Figure B.6-7. Note that the updated counter value is checked in every frame to be in the defined range [0,...,20].

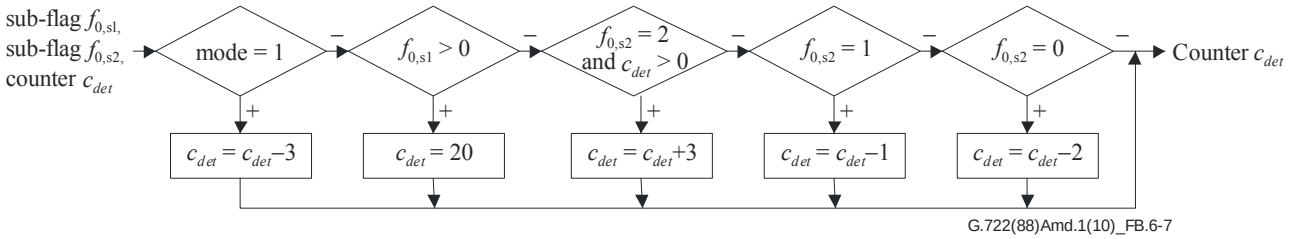


Figure B.6-7 – Computation of detection counter c_{det}

The detection flag f_0 is transmitted to the decoder in SWBL1 when there is at least one unused AVQ bit in this layer and both SWBL1 and SWBL2 are transmitted. The number of unused AVQ bits in SWBL1 is thus lowered by one in the current frame. If the detection flag is transmitted and holds $f_0 = 1$ and both SWBL1 and SWBL2 are transmitted, all zero sub-bands in the current frame are filled using restrained spectral envelope: the spectral envelope coefficients are multiplied by a factor of 0.1 with sign corresponding to the sign of the SWBL0 output coefficient. If the detection flag $f_0 = 0$, all zero sub-bands are replaced by SWBL0 decoded coefficients, or filled by coefficients derived from the AVQ coded coefficients (see further in clause B.6.7.4.3). Note that the change of

the detection flag f_0 value from one state to the other is done only in frames where there is at least one AVQ unused in SWBL. In frames with no "Unused AVQ bits", the value of the detection flag corresponds to its value in the previous frame. This keeps the synchronization between the encoder and the decoder. Also note that the detection flag holds $f_0 = 0$ and is not sent in the bitstream if only SWBL1 is transmitted.

B.6.7.4.3 Filling zero sub-bands

If both SWBL1 and SWBL2 are transmitted, the detection flag holds $f_0 = 0$ (frame with non-problematic zero sub-bands) and there are at least four unused AVQ bits in at least one of SWBL1 and SWBL2, the spectrum coefficients in one or two zero sub-bands are searched. To better match both the spectrum energy and distribution of amplitudes of the MDCT coefficients between the original spectrum and the reconstructed spectrum, the zero sub-band coefficients are derived from the AVQ coefficients in non-zero sub-bands using a correlation. The maximum correlation lag is sent to the decoder when four bits are available after the AVQ encoding and the maximum correlation is positive. This is applied for the first two zero sub-bands, one lag is sent in SWBL1 and the other lag in SWBL2 (when bits are available).

The search of best coefficients to be filled into a zero sub-band is based on finding the maximum correlation between the original normalized and ordered spectrum $S'(k)$ in a zero sub-band and the spectrum $\hat{S}'_{base}(k)$ referred further as a "base spectrum". The base spectrum $\hat{S}'_{base}(k)$ is extracted from the AVQ locally decoded spectrum $\hat{S}'(k)$ such that the zero sub-bands of $\hat{S}'(k)$ are omitted. Thus the length of the spectrum $\hat{S}'_{base}(k)$ is 24, if there are less than three non-zero sub-bands in $\hat{S}'(k)$ the filling is not performed.

Let us define M-dimensional vector $S'_{0sb1}(i)$, $i = 0,\cdots,7$, that corresponds to the coefficients of spectrum $S'(k)$ in the first zero sub-band. Similarly, a vector $S'_{0sb2}(i)$ corresponds to the coefficients of spectrum $S'(k)$ in the second zero sub-band (if it exists). Giving the fact that sub-bands are ordered according to the perceptual importance, vectors $S'_{0sb1}(i)$ and $S'_{0sb2}(i)$ represent $S'(k)$ spectrum coefficients of the two perceptually most important sub-bands not coded by the AVQ.

Therefore, if there are at least three non-zero sub-bands and there are at least four unused AVQ bits (after the AVQ coding and a 1-bit detection flag f_0 writing) in SWBL1, the maximum correlation $R_{\max 1}$ between the base spectrum $\hat{S}'_{base}(k)$ and the vector $S'_{0sb1}(i)$ is searched as:

$$R_{\max 1} = \max_{l}\left(\sum_{i=0}^{7} \hat{S}'_{base}(l+i)S'_{0sb1}(i)\right), \; l = 0,\cdots,14 \tag{B.6-88}$$

If $R_{\max 1}$ is positive, the lag δ_1 corresponding to the lag with the maximum correlation $R_{\max 1}$ is written into the SWBL1 bitstream and sent to the decoder. The reconstructed vector to be filled into the first zero sub-band is then computed as

$$\hat{S}'_{0sb1}(i) = \varphi_1 \cdot \hat{S}'_{base}(\delta_1 + i), \; i = 0,\cdots,7 \tag{B.6-89}$$

where φ_1 is an energy correction factor in the first zero sub-band that is computed as

$$\varphi_1 = \min\left(1,\; 1\Big/\sqrt{\sum_{i=0}^{7}\left(\hat{S}'_{base}(\delta_1 + i)\right)^2}\right) \tag{B.6-90}$$

If $R_{\max 1}$ is negative, a value of 15 is written to the SWBL1 bitstream to indicate that this first zero-sub-band is not filled by the base spectrum. In this case the filling of such zero sub-band is done using the SWBL0 output coefficients (details in clause B.7.3.6).

Similarly, if there are at least three non-zero sub-bands, and there are at least four unused AVQ bits in SWBL2, the maximum correlation $R_{\max 2}$ between the base spectrum $\hat{S}'_{base}(k)$ and the vector $S'_{0sb2}(i)$ is searched as

$$R_{\max 2} = \max_{l}\left(\sum_{i=0}^{7}\hat{S}'_{base}(l+i)S'_{0sb2}(i)\right), l = 0,\cdots,14 \quad \text{(B.6-91)}$$

In the case when δ_1 cannot be written into the SWBL1 bitstream (i.e., there are not at least four unused AVQ bits in SWBL1), the vector $S'_{0sb2}(i)$ is replaced by the vector $S'_{0sb1}(i)$ in the previous equation. This ensures the encoding of the most important zero sub-band coefficients. If $R_{\max 2}$ is positive, the lag δ_2 corresponding to the lag with the maximum correlation $R_{\max 2}$ is written into the SWBL2 bitstream and sent to the decoder. The reconstructed vector to be filled into this (first or second) zero sub-band is obtained as

$$\hat{S}'_{0sb2}(i) = \varphi_2 \cdot \hat{S}'_{base}(\delta_2 + i), i = 0,\cdots,7 \quad \text{(B.6-92)}$$

where φ_2 is an energy correction factor that corresponds to this zero sub-band and computed accordingly as φ_1 from Equation (B.6-90).

If $R_{\max 2}$ is negative, a value of 15 is written to the SWBL2 bitstream to indicate that the filling procedure by the base spectrum is not applied in this zero sub-band. In this case the filling of such zero sub-band is done using the SWBL0 output coefficients.

B.6.7.4.4 Backward reordering and denormalization

Vectors $\hat{S}'_{0sb1}(i)$ and $\hat{S}'_{0sb2}(i)$ found as described in clause B.6.7.4.3 are used to fill zero sub-bands in the spectrum $\hat{S}'(k)$ to form the optimized spectrum $\hat{S}''(k)$, see Figure B.6-6. Similarly to the normalization per sub-band and ordering described at the beginning of clause B.6.7.4, a reverse operation is needed. Thus sub-bands of the spectrum $\hat{S}''(k)$ are ordered back to the initial ordering and denormalized per sub-band to form the spectrum $\hat{S}^{norm}_{SHB}(k)$. Note that if there are more than two zero sub-bands, or there are not enough unused AVQ bits to encode lags δ_1 and δ_2, the zero sub-bands are replaced by the SWBL0 output spectrum to form the full coded SHB spectrum.

B.6.7.5 Encoding the SHB in mode 1

The input MDCT coefficients, of which the state is determined as "non sparse", are quantized based on the signal flow in SHB mode 1, as shown in Figure B.6-8. In order to achieve high quality for the "non sparse" input, as many as possible MDCT coefficients should be encoded. In the SHB mode 1, with regard to the coefficients, of which the absolute amplitude is higher than the spectral envelope from SWBL0, the difference between its magnitude and the offset calculated from spectral envelope is quantized using AVQ. At the decoder side, the decoded spectrum is obtained by adding the decoded error, which has not zero amplitude, and its offset, and then the remaining zero spectrum are filled with the spectrum envelope with the sign given in random.

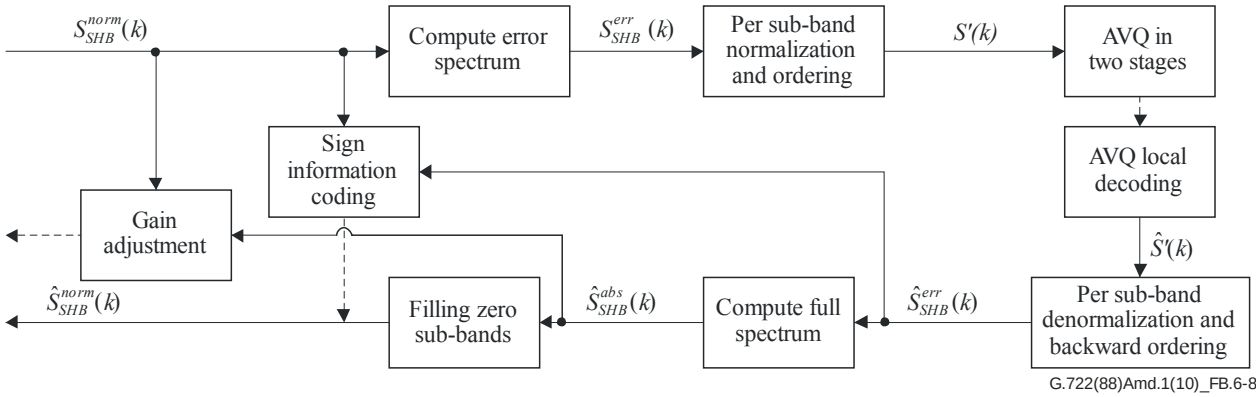


Figure B.6-8 – SHB mode 1 encoding block diagram

B.6.7.5.1 Compute error spectrum

The error spectrum, $S_{SHB}^{err}(k)$, $k = 0,...,63$, to be quantized in the SHB mode 1, is computed from the normalized spectrum, $S_{SHB}^{norm}(k)$, as follows:

$$S_{SHB}^{err}(8j+i) = \mathrm{sgn}\left(S_{SHB}^{norm}(8j+i)\right)\max\left(\left|S_{SHB}^{norm}(8j+i)\right| - 0.5\hat{f}_{rms}(j), 0\right) \quad \text{(B.6-93)}$$

where $i = 0,...,7$ is the coefficient index within a sub-band $j = 0,...,7$. The spectral envelope of the error spectrum in a sub-band j, $\hat{f}_{rms}^{err}(j)$, which is also required for the AVQ encoding, is estimated from $\hat{f}_{rms}(j)$ as:

$$\hat{f}_{rms}^{err}(j) = 0.6\hat{f}_{rms}(j) \quad \text{(B.6-94)}$$

Then the spectrum to be fed to the AVQ, shown in Equation (B.6-85) in case of the SHB mode 0 (see clause B.6.7.4), is replaced by:

$$S'(8j+i) = \frac{S_{SHB}^{err}\left(8\Omega_b(j)+i\right)}{\beta_{avq}\cdot\hat{f}_{rms}^{err}\left(\Omega_b(j)\right)},\ j = 0,...,7,\ i = 0,...,7 \quad \text{(B.6-95)}$$

The obtained spectrum is quantized using the AVQ with the procedure described in clause B.6.7.10.

B.6.7.5.2 Local decoding of MDCT coefficients

See clause B.6.7.4.1.

B.6.7.5.3 Backward reordering and denormalization

Same as clause B.6.7.4.4, except that the decoded error spectrum, $\hat{S}_{SHB}^{err}(k)$, $k = 0,...,63$, is obtained from the AVQ locally decoded spectrum $\hat{S}'(k)$ in the SHB mode 1 instead of the spectrum $\hat{S}_{SHB}^{norm}(k)$ obtained from spectrum $\hat{S}''(k)$ in the SHB mode 0.

B.6.7.5.4 Replacing zero coefficients in AVQ coded sub-bands with decoded envelope

In the SHB mode 1, regarding the decoded error spectrum, which results in zero amplitude in AVQ coding, the zero coefficients will be filled with the recalculated decoded envelope $\hat{f}_{rms}'(i)$ and their signs will be randomly set at the decoder. Meanwhile, at the encoder, the absolute value of local decoded MDCT spectrum is computed and the encoding process hereafter will be performed based on the obtained absolute value in the SHB mode 1. The recalculation of the decoded envelopes is performed as follows:

$$\hat{f}'_{rms}(j) = \sqrt{\frac{8\left[\hat{f}_{rms}(j)\right]^2 - \sum_{i=0}^{7}\left[\hat{S}^{tmp}_{SHB}(8j+i)\right]^2}{\sum_{i=0}^{7} f_{zero}(8j+i)}}, \quad j = 0,...,7 \tag{B.6-96}$$

where $\hat{S}^{tmp}_{SHB}(8j+i)$ is the temporary reconstructed spectrum from $\hat{S}^{err}_{SHB}(k)$,

$$\hat{S}^{tmp}_{SHB}(8j+i) = \begin{cases} 0 & \text{if} \quad \hat{S}^{err}_{SHB}(8j+i) = 0 \\ \left|\hat{S}^{err}_{SHB}(8j+i)\right| + 0.5\hat{f}_{rms}(i) & \text{otherwise} \end{cases} \tag{B.6-97}$$

and $f_{zero}(j)$ is a flag to indicate the zero coefficients in AVQ decoded sub-band *j*:

$$f_{zero}(8j+i) = \begin{cases} 1 & \text{if } \hat{S}^{err}_{SHB}(8j+i) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{B.6-98}$$

Then the absolute value of decoded normalized spectrum, $\hat{S}^{abs}_{SHB}(8j+i)$, is computed by

$$\hat{S}^{abs}_{SHB}(8j+i) = \begin{cases} \hat{f}'_{rms}(j), & \text{if } \hat{S}^{err}_{SHB}(8j+i) = 0 \\ \hat{S}^{tmp}_{SHB}(8j+i), & \text{otherwise} \end{cases} \tag{B.6-99}$$

B.6.7.5.5 Detection of problematic zero sub-bands

To detect problematic zero sub-bands in SHB mode 1, another classifier from the one described in clause B.6.7.4.2 is used. In this mode, we keep in mind that MDCT coefficients to be quantized are classified as to be not sparse and that the error MDCT spectrum is quantized by the AVQ. Similar to the technique described in clause B.6.7.4.2, a detection of zero sub-bands where the spectral envelope is not quantized too close to its original is performed. However, in this mode, the energy distribution in zero sub-bands is not tested. The detection is performed only when both SWBL1 and SWBL2 are transmitted.

Similar to Equations (B.6-86) and (B.6-87), the ratio *rat*(*j*) is computed. Then, a maximum ratio $r_{\max}$ is searched within zero sub-bands and quantized using a 1- or 2-bit quantizer. The number of quantization levels depends on the number of unused AVQ bits in SWBL1.

Let f_1 be the detection flag with value depending on $r_{\max}$ according to the following conditions:

$$f_1 = \begin{cases} 3, & \text{if } r_{\max} > 8, \\ 2, & \text{else if } r_{\max} > 4, \\ 1, & \text{else if } r_{\max} > 2, \\ 0, & \text{otherwise.} \end{cases} \tag{B.6-100}$$

The 2-bit detection flag is sent in the SWBL1 bitstream if there exist "Unused AVQ bits". In case that there are no "Unused AVQ bits", the flag f_1 is supposed to be 0. If there are only one unused AVQ bit, or five unused AVQ bits in SWBL1, the flag f_1 is upper bounded by 1 and its 1-bit value is sent to the decoder. The number of unused AVQ bits in SWBL1 is consequently reduced by one or two bits. Note that the detection flag holds $f_1 = 0$ and is not sent in the bitstream if only SWBL1 is transmitted.

Another difference between processing the SHB spectrum in SHB mode 1 compared to other SHB modes is that even in case that problematic frames are detected, the zero sub-bands filling described in clause B.6.7.4.3 is performed in SHB mode 1.

B.6.7.5.6 Filling zero sub-bands

In case of a sufficient number of unused AVQ bits and a transmission of both SWBL1 and SWBL2, the zero sub-bands are filled similarly as described in clause B.6.7.4.3, thus only the differences are emphasized here. The best vector to be filled into the zero sub-bands is found as follows:

– The base spectrum $\hat{S}'_{base}(k)$ is obtained by normalizing per sub-band the first three non-zero sub-bands from the decoded normalized SHB spectrum $\hat{S}^{abs}_{SHB}(k)$. Note that at the decoder side, the coefficients originally coded by the AVQ have the same signs as in the encoder, while the other coefficients (replaced by a modified spectral envelope) have signs often different from these at the encoder (this is due to the lack of this information at the decoder, where signs are deducted from the SWBL0 output spectrum).

– The 8-dimensional vectors $S'_{0sb1}(i)$ and $S'_{0sb2}(i)$ are obtained by normalizing per sub-band the coefficients of the spectrum $S^{norm}_{SHB}(k)$ in the first two zero sub-bands. Note that in SHB mode 1, the vectors $S'_{0sb1}(i)$ and $S'_{0sb2}(i)$ and the base spectrum are derived from the non ordered sub-bands spectrum (see Figure B.6-8).

– Lags δ_1 and δ_2 that correspond to the maximum correlation between the base vector and vectors $S'_{0sb1}(k)$ and $S'_{0sb2}(k)$, respectively, are found.

– The reconstructed vectors $\hat{S}'_{0sb1}(i)$ and $\hat{S}'_{0sb2}(i)$ to be filled into the zero sub-bands are reconstructed from the denormalized per sub-band base vector, i.e.,

$$\begin{aligned} \hat{S}'_{0sb1}(i) &= \varphi_1 \cdot \hat{f}_{env}(j_1) \cdot \hat{S}'_{base}(\delta_1 + i) \\ \hat{S}'_{0sb2}(i) &= \varphi_2 \cdot \hat{f}_{env}(j_2) \cdot \hat{S}'_{base}(\delta_2 + i) \end{aligned}, \qquad \text{(B.6-101)}$$

where $i = 0,\cdots,7$, and j_1 and j_2 correspond to the first and second zero sub-band, respectively. φ_1 and φ_2 are the energy correction factors for zero sub-band j_1 and j_2, respectively, computed as in Equation (B.6-90).

Finally, the vectors $\hat{S}'_{0sb1}(i)$ and $\hat{S}'_{0sb2}(i)$ are used to fill zero sub-bands in the spectrum $\hat{S}^{abs}_{SHB}(k)$ to form the normalized optimized decoded SHB spectrum $\hat{S}^{norm}_{SHB}(k)$, see Figure B.6-8. Note that if there are more than two zero sub-bands, or there are not enough unused AVQ bits to encode lags δ_1 and δ_2, the zero sub-bands are replaced by the signed normalized spectral envelope to form the full coded SHB spectrum (see details in clause B.7.4.5).

B.6.7.5.7 Send sign information

In SHB mode 1, when the locally decoded AVQ output has zero amplitude, it will be replaced by a coefficient with magnitude equivalent to the decoded spectral envelope, as previously described. The basic assumption is that the polarity, i.e., sign, of this coefficient will be randomly generated at the decoder. However, in order to enhance quality, the sign information of the input MDCT coefficients is transmitted to the decoder by making use of the remaining bits.

Assume that B_{swbl1_unused} is the number of unused bits in SWBL1 layer after AVQ, and $I_{swbl1\_sign}(u),\ u = 0,...,B_{swbl1\_unused} - 1$ are the indices to those unused bits. Then, the sign encoding comprises the following steps:

1) Start with $k = 0$ and $u = 0$.

2) If k lies within a sub-band j coded in SWBL1, and $\hat{S}_{SHB}^{err}(k) = 0$, then set the sign index as $I_{swbl1\_sign}(u) = \begin{cases} 0 & \text{if } S_{SHB}^{norm}(k) < 0 \\ 1 & \text{otherwise} \end{cases}$, and increment u by one.

3) Increment k, and if the coefficient runs out, i.e., $k = 63$, or the number of remaining bits runs out, i.e., $u = B_{swbl1\_unused}$, then stop the iteration. Otherwise, go to Step 2.

Similarly, the sign information of the coefficients are assigned to the unused number of bits in SWBL2 using the above computation steps, except that the number of unused bits in this case is B_{swbl2_unused} and the indices are $I_{swbl2\_sign}(v), v = 0,..., B_{swbl2\_unused} - 1$.

B.6.7.6 Encoding the SHB in mode 2

In the SHB mode 2, the input MDCT coefficients are encoded following the same procedure as the SHB mode 0 described in clause B.6.7.4, except that the one bit associated to the sparseness decision flag is not transmitted and, further, it is handled as an unused AVQ bit.

B.6.7.7 Gain adjustment

The quantized global gain in linear domain, $\hat{g}_{glob} = 2^{g_{glob}}$, is expressed by using the power of two as shown in Equation (B.6-71). After the AVQ coding stage, the global gain is adjusted by a gain correction factor and then this factor is transmitted to correct the magnitude of the decoded MDCT coefficients in SWBL1 layer.

The unquantized values of the adjusted global gain, g_{adj}, and the gain correction factor, $g_{correct}$, are represented by:

$$g_{adj} = g_{correct}\hat{g}_{glob}, \tag{B.6-102}$$

where

$$g_{correct} = \begin{cases} \dfrac{\sum_{j}\sum_{i=0}^{7} \left| S_{SHB}^{norm}(8j+i) \right| \hat{S}_{SHB}^{abs}(8j+i)}{\sum_{j}\sum_{i=0}^{7} (\hat{S}_{SHB}^{abs}(8j+i))^2}, & \text{if } f_{shb\_mode} = 1 \\ \dfrac{\sum_{j}\sum_{i=0}^{7} S_{SHB}^{norm}(8j+i)\hat{S}_{SHB}^{norm}(8j+i)}{\sum_{j}\sum_{i=0}^{7} (\hat{S}_{SHB}^{norm}(8j+i))^2}, & \text{otherwise} \end{cases} \tag{B.6-103}$$

for a j-th sub-band coded in SWBL1. The index, I_{gc}, of the quantized gain correction factor is selected as follows:

$$I_{gc} = \arg\min_{i=0,...,7} \left(\left| g_{correct} - \hat{g}_{correct}(i) \right|^2 \right), \tag{B.6-104}$$

where

$$\hat{g}_{correct}(i) = \begin{cases} 0.2i, & \text{if } g_{glob} = 0 \text{ and } 0 \le i \le 4 \\ 2^{\frac{k_i}{8}},\ k_i = \{-7,-5,-3,-2,-1,0,1,3\}, & \text{otherwise} \end{cases} \tag{B.6-105}$$

Thus the quantized value of the adjusted gain for SWBL1 layer is obtained by

$$\hat{g}_{adj} = \hat{g}_{correct}(I_{gc}) \cdot \hat{g}_{glob} \tag{B.6-106}$$

B.6.7.8 Local decoding of MDCT coefficients

The quantized SHB MDCT coefficients $\hat{S}_{SHB}(k)$ are locally decoded when all SWB layers are encoded and the SHB mode is different from 1 as

$$\hat{S}_{SHB}^{AVQ}(8j+i) = \begin{cases} \hat{g}_{adj} \cdot \hat{S}_{SHB}^{norm}(8j+i), & \text{for } j\text{-th sub-bands coded in SWBL1,} \\ \hat{g}_{glob} \cdot \hat{S}_{SHB}^{norm}(8j+i), & \text{for } j\text{-th sub-bands coded in SWBL2,} \\ \hat{S}_{SHB}^{BWE}(8j+i), & \text{otherwise,} \end{cases} \tag{B.6-107}$$

for $i = 0,\cdots,7$ and $j = 0,\cdots,7$. Note that sub-bands coded in SWBL2 also comprises the sub-bands filled using the technique described in clause B.6.7.4.3 and $\hat{S}_{SHB}^{BWE}(k)$ is the BWE decoded super higher band MDCT coefficients as described in clause B.7.3.6.

B.6.7.9 Gradient adjustment of the spectrum

The gradient adjustment of the spectrum is performed only when both SWBL1 and SWBL2 are transmitted to the decoder. In case there are still unused bits after AVQ coding and zero-band filling and the SHB mode is not 1, the information that adjusts the gradient of the spectrum magnitude is sent. The magnitude of the quantized SHB MDCT coefficients in each sub-band coded in each SWB layer, $\hat{S}_{SHB}^{AVQ}(8j+i),\ i = 0,...,7,$ is adjusted by multiplying it by gradient adjustment factors, $\gamma_{grd}(q,i),\ q$=0,...,3, $i = 0,...,7$. The gradient adjustment factors are used at the decoder to obtain the adjusted spectrum $\hat{S}_{SHB}^{adj}(8j+i)$. Each gradient adjustment factor is quantized using one or two bits per sub-band so as to minimize the error between the input SHB spectrum, $S_{SHB}(k)$, and its quantized spectrum $\hat{S}_{SHB}^{AVQ}(k)$.

For the SWBL1 layer, the gradient of the spectrum is adjusted by using the following steps:

1) Define the number of sub-bands coded in the SWBL1 layer as n_{SWBL1}, and the remaining number of bits in SWBL1 layer as B_{swbl1_unused}. Initialize the iteration number j = 0, remaining number of bits as $u = B_{swbl1\_unused}$, the remaining number of sub-bands for gradient adjustment as $v = n_{SWBL1}$, and the assigned number of bits per sub-band as $b_{grd}(i) = 0,\ \ i = 0,...,7$.

2) If $u \geq v+1$, it means that $\Omega_b(j)$-th sub-band can be assigned with two bits for gradient adjustment, while there are enough number of remaining bits *u* to assign at least one bit per remaining sub-bands. This means that two bits can be assigned to this $\Omega_b(j)$-th sub-band, otherwise only one bit can be assigned:

$$b_{grd}\left(\Omega_b(j)\right) = \begin{cases} 2, & \text{if } u > v \\ 1, & \text{otherwise} \end{cases} \tag{B.6-108}$$

3) Based on the assigned number of bits, the following VQ is performed and the gradient index is obtained as:

$$I_{swbl1\_grd}(j) = \underset{0 \leq q < 2^{b_{grd}(\Omega_b(j))}}{\arg\min} \left(\sum_{i=0}^{7} \left| S_{SHB}(8\Omega_b(j)+i) - \gamma_{grd}(q,i)\hat{S}_{SHB}(8\Omega_b(j)+i) \right| \right), \quad i = 0,...,7 \tag{B.6-109}$$

where $\gamma_{grd}(q,i)$ is the gradient adjustment factor stored in the following table,

$$\begin{Bmatrix} \gamma_{grd}(0,0) & \cdots & \gamma_{grd}(0,7) \\ \gamma_{grd}(1,0) & \cdots & \gamma_{grd}(1,7) \\ \gamma_{grd}(2,0) & \cdots & \gamma_{grd}(2,7) \\ \gamma_{grd}(3,0) & \cdots & \gamma_{grd}(3,7) \end{Bmatrix} = \begin{Bmatrix} 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 1.350 & 1.250 & 1.150 & 1.050 & 0.950 & 0.850 & 0.750 & 0.650 \\ 1.175 & 1.125 & 1.075 & 1.025 & 0.975 & 0.925 & 0.875 & 0.825 \\ 0.650 & 0.750 & 0.850 & 0.950 & 1.050 & 1.150 & 1.250 & 1.350 \end{Bmatrix} \quad \text{(B.6-110)}$$

4) Update the remaining number of bits $u = u - b_{grd}(\Omega_b(j))$, the remaining number of sub-bands for gradient adjustment $v = v - 1$, and increment iteration number *j*. If $u > 0$ and $v > 0$, go to Step 2.

The indices of gradient gains, $I_{swbl1\_grd}(j)$, are sequentially multiplexed as I_{swbl1_grd}. After the above iteration steps, the gradient adjustment is carried on to the remaining bits in SWBL2 layer in the same manner, where the number of remaining bits are initialized as $u = B_{swbl2\_unused}$ and the remaining number of sub-bands for gradient adjustment as $v = n_{SWBL2}$. This gives the gradient gain indices I_{swbl2_grd}.

B.6.7.10 AVQ quantization with split multi-rate lattice VQ

Prior to the AVQ quantization, the spectrum *S'*(*k*) of 64 coefficients is split into eight consecutive sub-bands of eight coefficients each. In SWBL1 AVQ coding, the first three sub-bands are quantized with 36 bits, whereas in SWBL2 AVQ coding, four sub-bands are quantized with 40 bits. The sub-bands are quantized with an 8-dimensional multi-rate algebraic vector quantizer. The AVQ codebooks are subsets of the Gosset lattice, referred to as the RE_8 lattice.

B.6.7.10.1 Multi-rate AVQ with the Gosset lattice RE_8

B.6.7.10.1.1 Gosset lattice RE_8

The Gosset lattice RE_8 is defined as the following union:

$$RE_8 = 2D_8 \cup \left\{2D_8 + (1,1,1,1,1,1,1,1)\right\} \quad \text{(B.6-111)}$$

where D_8 is the 8-dimensional lattice composed of all points with integer components with the constraint that the sum of the eight components is even. The lattice $2D_8$ is simply the D_8 lattice scaled by two. This implies that the sum of the components of a lattice point in $2D_8$ is an integer multiple of four. Therefore, the eight components of a RE_8 lattice point have the same parity (either all even or all odd) and their sum is a multiple of four.

All points in lattice RE_8 lie on concentric spheres of radius $\sqrt{8n_j}$, n_j being the codebook number in sub-band *j*. 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.

B.6.7.10.1.2 Multi-rate codebooks in Gosset lattice RE_8

To form a vector codebook at a given rate, only lattice points inside a sphere in eight dimensions of a given radius are taken. Codebooks of different bit rates can be constructed by including only spheres up to a given radius. Multi-rate codebooks are formed by taking subsets of lattice points inside spheres of different radii.

B.6.7.10.1.2.1 Base codebooks

First, base codebooks are designed. A base codebook contains all lattice points from a given set of spheres up to a number n_j. Four base codebooks, noted Q_0, Q_2, Q_3, and Q_4, are used. There are 36 non-null absolute leaders plus the zero leader (the origin): Table B.6-11 gives the list of these leaders and indicates to which codebook a leader belongs. Q_0, Q_2, Q_3, and Q_4 are constructed with respectively 0, 8, 12, and 16 bits. Hence codebook Q_{n_j} requires $4n_j$ bits to index any point in that codebook.

B.6.7.10.1.2.2 Voronoi extensions

From a base codebook C_{AVQ} (i.e., a codebook containing all lattice points from a given set of spheres up to a number n_j), an extended codebook can be generated by multiplying the elements of C_{AVQ} by a factor M_j^v, and adding a second-stage codebook called the Voronoi extension. This construction is given by

$$\mathbf{c}_j = M_j^v \cdot \mathbf{z}_j + \mathbf{v}_j \qquad \text{(B.6-112)}$$

where M_j^v is the scaling factor, $\mathbf{z}_j$ is a point in a base codebook C_{AVQ} and $\mathbf{v}_j$ is a point in the Voronoi extension. The extension is computed in such a way that any point $\mathbf{c}_j$ from Equation (B.6-112) is also a lattice point in RE_8. The scaling factor M_j^v is a power of 2 ($M_j^v = 2^{r_j^v}$), where r_j^v is called the Voronoi extension order.

Such extended codebooks include lattice points that extend further out from the origin than the base codebook. When a given lattice point $\mathbf{c}_j$ is not included in a base codebook C_{AVQ} (Q_0, Q_2, Q_3 or Q_4), the so-called Voronoi extension is applied, using the Q_3 or Q_4 base codebook part.

Giving the available bit-budget in particular layers, the maximum Voronoi extension order is $r_j^v = 2$. Therefore, for Q_3 or Q_4, two extension orders are used: $r_j^v = 1$ or 2 ($M_j^v = 2$ or 4).

When $r_j^v = 0$, there is no Voronoi extension, and only a base codebook is used.

B.6.7.10.1.2.3 Codebook rates

There are eight codebooks: the first four are base codebooks without Voronoi extension and the last four with Voronoi extension. The codebook number n_j is encoded as a unary code with n_j "1" bits and a terminating "0". Table B.6-10 gives for each of the eight codebooks, its base codebook, its Voronoi extension order ($r_j^v = 0$ indicates that there is no Voronoi extension), and its unary code.

Table B.6-10 – Multi-rate codebooks in RE_8 lattice

Codebook number n_j	Base Codebook	Voronoi extension order r_j^v	Unary code for n_j
0	Q_0	0	0
2	Q_2	0	10
3	Q_3	0	110
4	Q_4	0	1110
5	Q_3	1	11110
6	Q_4	1	111110
7	Q_3	2	1111110
8	Q_4	2	11111110

For the base codebook Q_0, ($n_j = 0$), there is only one point in the codebook and one bit is used to transmit the unary code corresponding to n_j.

For the other three base codebooks Q_{n_j} ($n_j = 2$, 3, or 4) without Voronoi extension:

– n_j bits are used to transmit the unary code corresponding to n_j;

– $4n_j$ bits are required to index a point in Q_{n_j};

– thus $5n_j$ bits are used in total.

For codebooks with Voronoi extension ($n_j > 4$):

– n_j bits are used to transmit the unary code corresponding to the base codebook number Q_3 (respectively Q_4) if n_j is even (respectively odd) and the Voronoi extension order r_j^v is 1 if $n_j < 7$, or 2 otherwise;

– 12 bits (respectively 16 bits) are required to index the point $\mathbf{z}_j$ in the base codebook Q_3 (respectively Q_4);

– $8\,r_j^v$ bits are required to index the 8-dimensional point $\mathbf{v}_j$ in the Voronoi extension of order r_j^v;

– thus, $5n_j$ bits are used in total.

In the codebook number encoding, a simple bit overflow check is performed: in case when the last AVQ coded sub-band of the spectrum $S'(k)$ is quantized, $n_j > 0$ and only $5n_j - 1$ bits are available for the quantization, the terminating "0" in the codebook number coding is not encoded. At the decoder, the same bit overflow check enables the right decoding of the codebook number in this sub-band.

B.6.7.10.2 Quantization with RE_8 lattice

In lattice quantization, the operation of finding the nearest neighbour of the input spectrum $S'(k)$ among all codebook points is reduced to a few simple operations, involving rounding the components of spectrum $S'(k)$ 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 index that will be sent to the decoder. The larger the components of the input spectrum $S'(k)$, 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 spectrum is first scaled down by the AVQ gain, then each 8-dimensional block of spectrum coefficients is quantized in the lattice and finally scaled up again to produce the quantized spectrum.

B.6.7.10.2.1 AVQ gain estimation

Prior to the quantization (nearest neighbour search and indexation of the nearest neighbour), the input spectrum has to be scaled down to ensure that the total bit consumption will remain within the available bit-budget (36 bits in SWBL1, and 40 bits in SWBL2).

As there is a high correlation between the AVQ gain and the SWBL0 global gain, the quantized SWBL0 global gain is used to multiply each block of the encoded spectrum.

A first estimation of the total bit-budget *nbits* without scaling (i.e., with an AVQ gain equal to 1) is performed:

$$nbits = \sum_j R_j \qquad \text{(B.6-113)}$$

where R_j is a first estimate of the bit budget to encode the sub-band j given by:

$$R_j = 5\log_2\left(\frac{E_j}{2}\right) \tag{B.6-114}$$

with E_j being the energy (with a lower limit set to 2) of each sub-band $\mathbf{S}'(8j)$:

$$E_j = \max\left(2, \sum_{i=0}^{7}\left[S'(8j+i)\right]^2\right) \tag{B.6-115}$$

Constant β_{avq} used in Equations (B.6-85) and (B.6-95) causes sub-band energies E_j to be too large to ensure that the total bit consumption (*nbits*) remains within the available bit-budget. Hence, it is necessary to estimate an AVQ gain so that the quantization of scaled down spectrum $S'_{norm}(k)$ in the RE_8 lattice will produce a set of parameters that stay within the bit-budget.

This gain estimation is performed in the iterative procedure described below.

Let NB_BITS (36 or 40) be the number of bits available for the quantization process, and NB_SBANDS the number of 8-dimensional sub-bands to be quantized (i.e., 3 or 4):

Initialization:

fac = 128,

$offset$ = 0,

$nbits_{max}$ = 0.95 (NB_BITS – NB_SBANDS)

for i = 1:10

$offset = offset + fac$

$$nbits = \sum_{j=1}^{\text{NB\_SBANDS}} \max\left(0, R_j - offset\right)$$

if $nbits \le nbits_{max}$, then

$offset = offset - fac$

$fac = fac / 2$

After the 10-th iteration the AVQ gain is equal to $10\exp\left(0.1 offset \cdot \log_{10}(2)\right)$ and is used to obtain the scaled spectrum $S'_{norm}(k)$:

$$S'_{norm}(k) = S'(k) \Big/ \left[10\exp\left(0.1 \cdot offset \cdot \log_{10}(2)\right)\right] \tag{B.6-116}$$

B.6.7.10.2.2 Nearest neighbour search

The search of the nearest neighbour in the lattice RE_8 is equivalent to searching for the nearest neighbour in the lattice $2D_8$ and for the nearest neighbour in the lattice $2D_8$ + (1,1,1,1,1,1,1,1), and finally selecting among those two lattice points the closest to $\mathbf{S}'_{norm}(8j)$ in its quantized version $\hat{\mathbf{S}}'(8j)$.

Based on the definition of RE_8, the following fast algorithm is used to search the nearest neighbour of an 8-dimensional sub-band $\mathbf{S}'_{norm}(8j)$ among all lattice points in RE_8:

Search for the nearest neighbour $\mathbf{y}_{1j}$ in $2D_8$ of $\mathbf{S}'_{norm}(8j)$:

Compute $\mathbf{z}_j = 0.5\mathbf{S}'_{norm}(8j)$.

Round each component of $\mathbf{z}_j$ to the nearest integer to generate $\mathbf{z}'_j$.

Compute $\mathbf{y}_{1j} = 2\mathbf{z}'_j$.

Calculate the sum S of the eight components of $\mathbf{y}_{1j}$.

if S is not an integer multiple of four, then modify its I-th component as follows:

$$y_{1j}(I) = \begin{cases} y_{1j}(I) - 2, & \text{if } z_j(I) - y_{1j}(I) < 0, \\ y_{1j}(I) + 2, & \text{otherwise.} \end{cases}$$

where I = arg (max(| $z_j(i) - y_{1j}(i)$ |))

Search for the nearest neighbour $\mathbf{y}_{2j}$ in $2D_8 + (1,1,1,1,1,1,1,1)$ of $\mathbf{S}'_{norm}(8j)$:

Compute $\mathbf{z}_j = 0.5(\mathbf{S}'_{norm}(8j) - \mathbf{1.0})$ where **1.0** denotes an 8-dimensional vector with all ones.

Round each component of $\mathbf{z}_j$ to the nearest integer to generate $\mathbf{z}'_j$.

Compute $\mathbf{y}_{2j} = 2\mathbf{z}'_j$.

Calculate the sum S of the eight components of $\mathbf{y}_{2j}$.

if S is not an integer multiple of four then modify its I-th component as follows:

$$y_{2j}(I) = \begin{cases} y_{2j}(I) - 2, & \text{if } z_j(I) - y_{2j}(I) < 0, \\ y_{2j}(I) + 2, & \text{otherwise.} \end{cases}$$

where I = arg (max(| $z_j(i) - y_{2j}(i)$ |))

Compute $\boldsymbol{y}_{2j} = \boldsymbol{y}_{2j} + \mathbf{1.0}$.

Select between $\mathbf{y}_{1j}$ and $\mathbf{y}_{2j}$ as the closest point $\hat{\mathbf{S}}'(8j)$ in RE_8 to $\mathbf{S}'_{norm}(8j)$:

$$\hat{\mathbf{S}}'(8j) = \begin{cases} \mathbf{y}_{1j}, & \text{if } e_{1j} > e_{2j}, \\ \mathbf{y}_{2j}, & \text{otherwise.} \end{cases}$$

where $e_{1j} = (\mathbf{S}'_{norm}(8j) - \mathbf{y}_{1j})^2$ and $e_{2j} = (\mathbf{S}'_{norm}(8j) - \mathbf{y}_{2j})^2$.

B.6.7.10.2.3 Indexation

The quantized scaled sub-band $\hat{\mathbf{S}}'(8j)$ of $\mathbf{S}'_{norm}(8j)$ is a point $\mathbf{c}_j$ in a RE_8 lattice codebook, an index for each $\mathbf{c}_j$ has to be computed and later inserted into the bitstream.

This index is actually composed of three parts:

1) a codebook number n_j;
2) a vector index I_j, which uniquely identifies a lattice vector in a base codebook C_{AVQ};
3) and if $n_j > 4$, an 8-dimensional Voronoi extension index $\mathbf{I}_j^v$ that is used to extend the base codebook when the selected point in the lattice is not in a base codebook C_{AVQ}.

The calculation of an index for a given point $\mathbf{c}_j$ in the RE_8 lattice is performed as follows:

First, it is verified whether $\mathbf{c}_j$ is in a base codebook C_{AVQ} by identifying its sphere and its leader:

– if $\mathbf{c}_j$ is in a base codebook, the index used to encode $\mathbf{c}_j$ is thus the codebook number n_j plus the index I_j of the lattice point $\mathbf{c}_j$ in Q_{n_j}.

– Otherwise, the parameters of the Voronoi extension (see Equation (B.6-112)) have to be determined: the scaling factor M_v, the base codebook C_{AVQ} (Q_3 or Q_4), the point $\mathbf{z}_j$ in this base codebook, and the point $\mathbf{v}_j$ in the Voronoi extension. Then, the index used to encode is

composed of the codebook number n_j ($n_j > 4$) plus the index I_j of the lattice point $\mathbf{z}_j$ in the base codebook C_{AVQ} (Q_3 or Q_4), and the index I_j^v of $\mathbf{v}_j$ in the Voronoi extension.

B.6.7.10.2.3.1 Indexing a codebook number

As explained in clause B.6.7.10.1.2.3, the codebook index n_j is unary encoded with n_j bits except for $n_j = 0$ that is coded with one bit (see Table B.6-10).

B.6.7.10.2.3.2 Indexing of codevector in base codebook

The index I_j indicates the rank of codevector $\mathbf{z}_j$ in j-th sub-band, i.e., the permutation to be applied to a specific leader to obtain $\mathbf{z}_j$. The index computation is done in several steps, as follows:

1) The input codevector $\mathbf{z}_j$ is decomposed into a sign vector $\mathbf{s}_0$ and an absolute vector $\mathbf{y}_0$ following a two-path procedure.

2) The sign vector is encoded, the associated index $bits_{sign}(\mathbf{s}_0)$ and the number of non-zero components in $\mathbf{z}_j$ $sign_{nb}(s_0)$ are obtained. More details are given in subsequent clauses.

3) The absolute vector is encoded using a multi-level permutation-based index encoding method, and the associated index $rank(\mathbf{y}_0)$ is obtained.

4) The absolute vector index $rank(\mathbf{y}_0)$ and the sign index $bits_{sign}(\mathbf{s}_0)$ are added together in order to obtain the input vector rank: $rank(\mathbf{z}_j)$.

$$rank(\mathbf{z}_j) = rank(\mathbf{y}_0) \cdot 2^{sign_{nb}(\mathbf{s}_0)} + bits_{sign}(\mathbf{s}_0) \tag{B.6-117}$$

5) Finally, the offset $lead_{offset}(\mathbf{z}_j)$ is added to the rank. The index I_j is obtained by

$$I_j = lead_{offset}(\mathbf{z}_j) + rank(\mathbf{z}_j) \tag{B.6-118}$$

B.6.7.10.2.3.2.1 Sign vector encoding

The number of bits required for encoding the sign vector elements is equal to the number of non-zero elements in the codevector. "1" represents a negative sign and "0" a positive sign. As lattice RE_8 quantization is used, the sum of all the elements in a codevector is an integer multiple of four. If there is any change of sign in the non-zero element, the sum may not be a multiple of four anymore, in that case, the last element sign in the sign vector will be omitted. For example, the sign vector of the input vector (–1, –1, 1, 1, 1, 1, –1, –1) in leader 1 (see Table B.6-11) has seven bits and its value is 0x1100001.

B.6.7.10.2.3.2.2 Encoding of the absolute vector and its position vector

The encoding method for the absolute vector works as follows. The absolute vector is first decomposed into ML_{max} levels. The highest-level vector L_0 is the original absolute vector. The n value for L_n is initialized to zero. Then:

1) Increment the n value. At level L_n ($0 < L_n < ML_{max}$), the intermediate absolute value vector is obtained by removing the most frequent element as given in the decomposition order column of Table B.6-11 from the upper-level vector. The remaining elements are built into a new absolute vector for the current level; it has a position order related to the level L_{n-1} absolute vector. All position values of the remainder elements are used to build a position vector.

2) The position vector of the current lower-level vector related to its upper-level vector is indexed based on a permutation and combination function, the indexing result being called the middle index $I_{mid,n}$. For the absolute vector in the current lower level, the position vector indexing is computed as follows:

$$I_{mid,n} = C_{m_{n-1}}^{m_n} - C_{m_{n-1}-q_0}^{m_n} + \sum_{i=1}^{i<m_n} \left(C_{m_{n-1}-q_{i-1}}^{m_n-i} - C_{m_{n-1}-q_i}^{m_n-i} \right) \tag{B.6-119}$$

$$I_{final} = I_{final} \cdot C_{m_{n-1}}^{m_n} + I_{mid,n} \quad \text{(B.6-120)}$$

where I_{final} is initialized to zero before the first step at the beginning of the procedure. The elements q_0, q_1, $q_2 \cdots$ are the element values in the L_n level position vector ranged from left to right according to their level, m_{n-1} is the dimension of the upper-level absolute vector, m_n is the dimension of the current-level absolute vector, C_q^m represents the permutation and combination formula $C_q^m = \frac{q!}{m!(p-q)!}$, $q,m=\{1,2,3,4,8\}$, and $q > m$. All the values for C_q^m can be stored in a simple table in order to avoid calculation of factorials. The L_{n-1} level final-index, I_{final}, is multiplied by the possible index value number, $C_{m_{n-1}}^{m_n}$, in the current level and is added to the index, $I_{mid,n}$, in the current level, to obtain the final index, I_{final}, of the current level.

3) Repeat steps 1 and 2 until there is only one type of element left in the current absolute vector. The I_{final} for the lowest level is the rank of the absolute vector called *rank(***y**$_0$*)*. Table B.6-11 is a sample extracted from the 36 leader table case. The leaders are indexed by K_a. The decomposition order corresponds to the level order. The decomposition order column gives the order in which the element will be removed from the higher level. The last column gives the three class parameters, the first one is the number of sign bits, S_n, the second one is the number of decomposition levels and equals the number of element types in the leader, V_c, from the third one to the last one they represent the absolute vector dimension in each lower level except the highest level, m_1, m_2, m_3 (note that the dimension for the highest level is eight, but is not listed in Table B.6-11).

Table B.6-11 – List of leaders in base codebooks Q_{n_j} with their decomposition order and set parameter of multi-level permutation-based encoding

K_a	Leader	Decomposition order	S_n, V_c, m_1, m_2, m_3	Q_0	Q_2	Q_3	Q_4
	{0,0,0,0,0,0,0,0}			X			
0	{1,1,1,1,1,1,1,1}	{1}	{7,1}		X	X	
1	{2,2,0,0,0,0,0,0}	{0,2}	{2,2,2}		X	X	
2	{2,2,2,2,0,0,0,0}	{0,2}	{4,2,4}			X	
3	{3,1,1,1,1,1,1,1}	{1,3}	{7,2,1}			X	
4	{4,0,0,0,0,0,0,0}	{0,4}	{1,2,1}		X	X	
5	{2,2,2,2,2,2,0,0}	{2,0}	{6,2,2}				X
6	{3,3,1,1,1,1,1,1}	{1,3}	{7,2,2}				X
7	{4,2,2,0,0,0,0,0}	{0,2,4}	{3,3,3,1}			X	
8	{2,2,2,2,2,2,2,2}	{2}	{8,1}				X
9	{3,3,3,1,1,1,1,1}	{1,3}	{7,2,3}				X
10	{4,2,2,2,2,0,0,0}	{2,0,4}	{5,3,4,1}				X
11	{4,4,0,0,0,0,0,0}	{0,4}	{2,2,2}			X	
12	{5,1,1,1,1,1,1,1}	{1,5}	{7,2,1}				X
13	{3,3,3,3,1,1,1,1}	{1,3}	{7,2,4}				X
14	{4,2,2,2,2,2,2,0}	{2,0,4}	{7,3,2,1}				X
15	{4,4,2,2,0,0,0,0}	{0,2,4}	{4,3,4,2}				X

Table B.6-11 – List of leaders in base codebooks Q_{n_j} with their decomposition order and set parameter of multi-level permutation-based encoding

K_a	Leader	Decomposition order	S_n, V_c, m_1, m_2, m_3	Q_0	Q_2	Q_3	Q_4
16	{5,3,1,1,1,1,1,1}	{1,3,5}	{7,3,2,1}				X
17	{ 6,2,0,0,0,0,0,0}	{0,2,6}	{2,3,2,1}			X	
18	{ 4,4,4,0,0,0,0,0}	{0,4}	{3,2,3}				X
19	{ 6,2,2,2,0,0,0,0}	{0,2,6}	{4,3,4,1}				X
20	{ 6,4,2,0,0,0,0,0}	{0,2,4,6}	{3,4,3,2,1}				X
21	{ 7,1,1,1,1,1,1,1}	{1,7}	{7,2,1}				X
22	{ 8,0,0,0,0,0,0,0}	{0,8}	{1,2,1}				X
23	{6,6,0,0,0,0,0,0}	{0,6}	{2,2,2}				X
24	{8,2,2,0,0,0,0,0}	{0,2,8}	{3,3,3,1}				X
25	{8,4,0,0,0,0,0,0}	{0,4, 8}	{2,3,2,1}				X
26	{9,1,1,1,1,1,1,1}	{1,9}	{7,2,1}				X
27	{10,2,0,0,0,0,0,0}	{0,2,10}	{2,3,2,1}				X
28	{8,8,0,0,0,0,0,0}	{0,8}	{2,2,2}				X
29	{10,6,0,0,0,0,0,0}	{0,6,10}	{2,3,2,1}				X
30	{12,0,0,0,0,0,0,0}	{0,12}	{1,2,1}				X
31	{12,4,0,0,0,0,0,0}	{0,4,12}	{2,3,2,1}				X
32	{10,10,0,0,0,0,0,0}	{0,10}	{2,2,2}				X
33	{14,2,0,0,0,0,0,0}	{0,2,14}	{2,3,2,1}				X
34	{12,8,0,0,0,0,0,0}	{0,8,12}	{2,3,2,1}				X
35	{16,0,0,0,0,0,0,0}	{0,16}	{1,2,1}				X

The last value of the decomposition order for the leader K_a = 20 is stored separately because this leader is the only one with four different values, the second dimension of the decomposition order being thus reduced from 4 to 3.

Figure B.6-9 gives an encoding example for the leader K_a = 20.

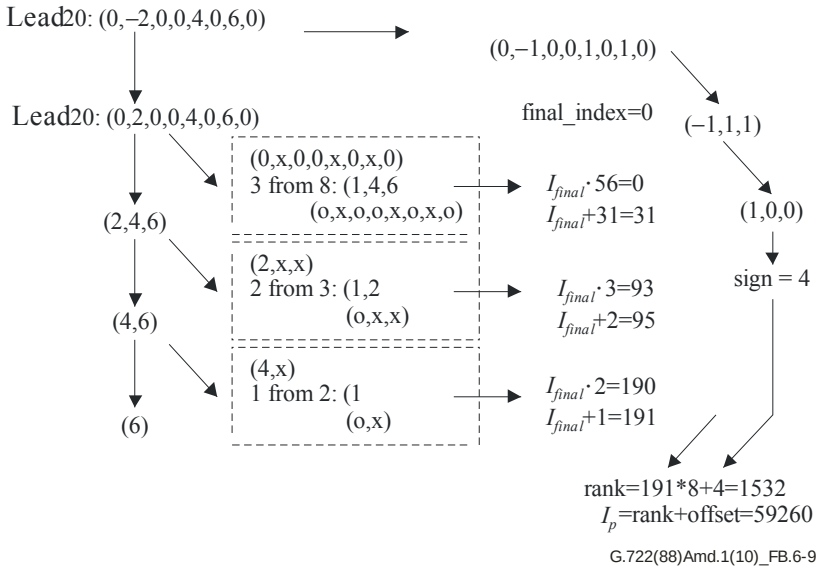


Figure B.6-9 – Example processing for K_a = 20

For example, in case the input vector is {0,–2,0,0,4,0,6,0}, the absolute input vector will be {0,2,0,0,4,0,6,0}, its associated leader can be found for K_a equal to 20. The set of decomposition order is {0,2,4,6}. For the highest level L_0, element "0" is removed first from the absolute vector. The first level L_1 absolute vector is {2,4,6}, its position vector is {1,4,6}. The second element which will be removed is "2", the second level L_2 absolute vector is {4,6}, its position vector is {1,2}. The third element which will be removed is "4", the third level L_3 absolute vector is {6}, its position vector is {1}.

B.6.7.10.2.4 Voronoi extension determination and indexing

If the nearest neighbour $\mathbf{c}_j$ is not in the base codebook, then the Voronoi extension has to be determined through the following steps.

a) Set the Voronoi extension order $r_j^v = 1$ and the scaling factor $M_j^v = 2^{r_j^v}$.

b) Compute the Voronoi index $\mathbf{I}_j^v$ of the lattice point $\mathbf{c}_j$ that depends on the extension order r_j^v and the scaling factor M_j^v. The Voronoi index is computed via component-wise modulo operations such that $\mathbf{I}_j^v$ depends only on the relative position of $\mathbf{c}_j$ in a scaled and translated Voronoi region:

$$\mathbf{I}_j^v = \mathrm{mod}_{M_j^v}(\mathbf{c}_j \mathbf{G}^{-1}) \qquad \text{(B.6-121)}$$

where $\mathbf{G}$ is the RE_8 generator matrix. Hence, the Voronoi index $\mathbf{I}_j^v$ is a vector of integers with each component in $[0,..., M_j^v - 1]$.

c) Compute the Voronoi codevector $\mathbf{v}_j$ from the Voronoi index $\mathbf{I}_j^v$. The Voronoi codevector is obtained as

$$\mathbf{v}_j = \mathbf{y}_{1j} - M_j^v \cdot \hat{\mathbf{y}}_{2j} \qquad \text{(B.6-122)}$$

where $\hat{\mathbf{y}}_{2j}$ is the nearest neighbour of $\mathbf{y}_{2j}$ in infinite RE_8 (see clause B.6.7.10.2.2 for search details) and $\mathbf{y}_{1j}$ and $\mathbf{y}_{2j}$ are defined as

$$\mathbf{y}_{1j} = \mathbf{I}_j^v \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{B.6-123}$$

and

$$\mathbf{y}_{2j} = \left(\mathbf{y}_{1j} - [2,0,0,0,0,0,0,0]\right) \Big/ M_j^v \tag{B.6-124}$$

d) Compute the difference vector $\mathbf{w}_j = \mathbf{c}_j - \mathbf{v}_j$. This difference vector $\mathbf{w}_j$ always belongs to the scaled lattice $M_j^v \cdot RE_8$. Compute $\mathbf{z}_j = \mathbf{w}_j / M_j^v$, i.e., apply the inverse scaling to the difference vector $\mathbf{w}_j$. The codevector $\mathbf{z}_j$ belongs to the lattice RE_8 since $\mathbf{w}_j$ belongs to $M_j^v \cdot RE_8$ lattice.

e) Verify whether $\mathbf{z}_j$ is in the base codebook C_{AVQ} (i.e., in Q_3 or Q_4).

If $\mathbf{z}_j$ is not in C, increment the extension order r_j^v by 1, multiply the scaling factor M_j^v by 2, and go back to sub-step (b).

Otherwise, if $\mathbf{z}_j$ is in C, then the Voronoi extension order r_j^v has been found and the scaling factor $M_j^v = 2^{r_j^v}$ is sufficiently large to encode the index of $\mathbf{c}_j$.

B.6.7.10.3 Insertion of AVQ parameters into the bitstream

The parameters of the AVQ in each sub-band j consist of the codebook number n_j, the vector index in base codebook I_j and the 8-dimensional Voronoi index $\mathbf{I}_j^v$. The codebook numbers n_j are in the set of integers {0, 2, 3, 4, 5, 6, 7, 8} and the size of its unary code representation is n_j bits with the exception of Q_0 that requires one bit and a possible overflow in the last AVQ coded sub-band. The size of each index I_j and $\mathbf{I}_j^v$ is given by $4n_j$ bits and $8\,r_j^v$ bits, respectively. The number of sub-bands is equal to three in SWBL1 AVQ coding, four in SWBL2 AVQ coding.

The AVQ parameters n_j, I_j, $\mathbf{I}_j^v$, are written sequentially in groups corresponding to the same sub-band into the corresponding bitstream as

$$[(n_0 I_0 \mathbf{I}_0^v)(n_1 I_1 \mathbf{I}_1^v)...] \tag{B.6-125}$$

Note that if the lattice point in the block j is in the base codebook C, the Voronoi extension is not searched and consequently the index $\mathbf{I}_j^v$ is not written into the bitstream in this group.

The actual bit-budget needed to encode AVQ parameters in current frame and layer varies from frame to frame. The difference of bits between the allocated bits and actually spent bits are the "Unused AVQ bits".

B.7 Functional description of the decoder

B.7.1 Decoder overview

Figure B.7-1 shows the high-level block diagram of the ITU-T G.722 SWB decoder. The whole bitstream is de-multiplexed into three parts: ITU-T G.722 compatible core bitstream, wideband enhancement bitstreams (G722EL0 and G722EL1) and super higher band bitstreams (SWBL0, SWBL1 and SWBL2).

R1sm bitrate mode is based on the ITU-T G.722 core at 56 kbit/s wideband layer, and gives a superwideband signal reproduced by SWBL0 on top of an enhanced version of the wideband signal using G722EL0. R2sm bitrate mode is based on the ITU-T G.722 core at 64 kbit/s, and reproduces superwideband signals enhanced further from R1sm using SWBL1. R3sm bitrate mode is also based on the ITU-T G.722 core at 64 kbit/s, and reproduces a superwideband signal enhanced further from R2sm using G722EL1 and SWBL2.

Meanwhile, in order to improve the performance of the wideband signal, a post-processing of higher band signals is performed. To improve the quality under channel errors such as packet losses, a frame erasure concealment (FERC) algorithm is applied to the lower- and higher band signals, and independently to the super higher band signals. In case of erased frames, the FERC modules generate a replacement signal in function of the past memorized decoded signal. For the first valid frames after an erasure, the lower band FERC ensures the continuity of the signal by crossfading between the replacement signal and the decoded signal. For the other valid frames, the FERC modules only memorize the decoded signal to be able to generate a replacement signal in case of an erasure. The reproduced wideband and super higher band signals, $\hat{s}_{WB}(n)$ and $\hat{s}_{SHB}(n)$, are combined using a synthesis QMF filterbank to generate a 32 kHz sampled signal $\hat{S}_{SWB}(n)$.

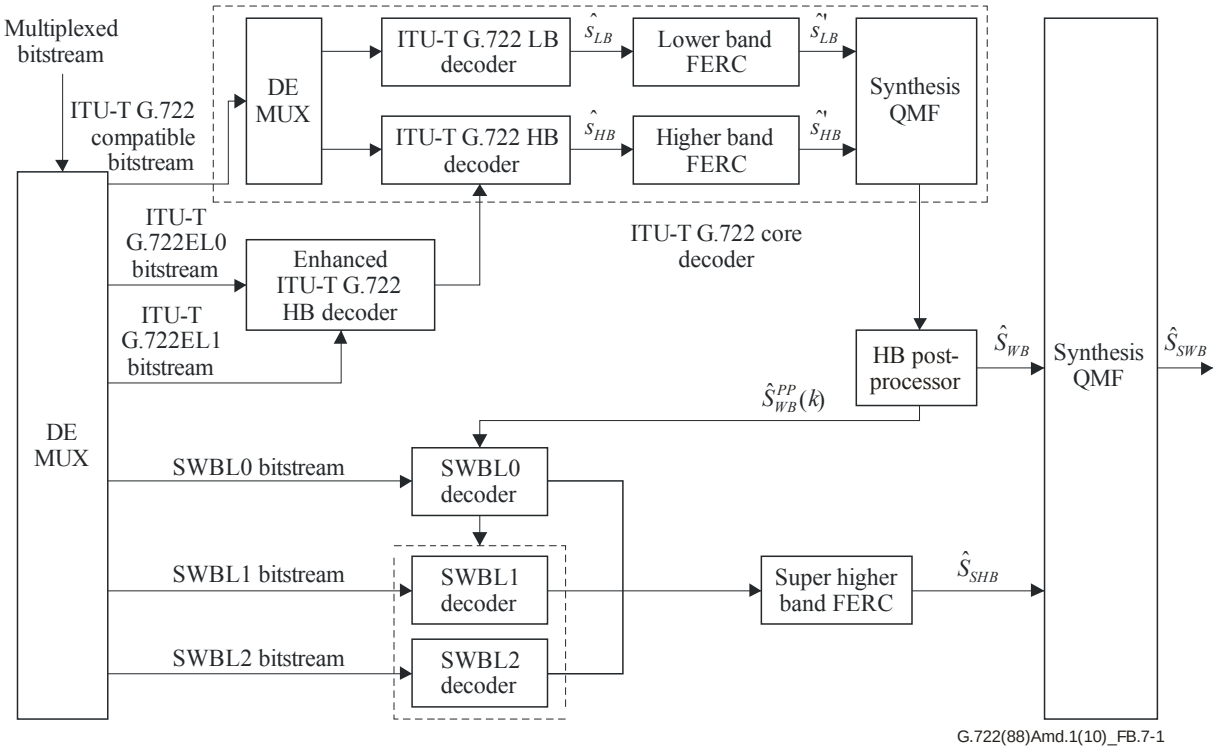


Figure B.7-1 – High-level decoder block diagram

B.7.2 ITU-T G.722 core decoder and G722EL0 and G722EL1 decoders

The ITU-T G.722 core layer decoder is an improved version of ITU-T G.722. As shown in the block diagram of Figure B.7-1, this decoder comprises the analysis QMF and the ITU-T G.722 LB, ITU-T G.722 HB, and enhanced ITU-T G.722 HB decoders. These blocks are described in the

following subclauses. In the following, the output of the ITU-T G.722 LB and HB decoders will be denoted $\hat{s}_{LB}(n)$ and $\hat{s}_{HB}(n)$, respectively.

B.7.2.1 Decoding ITU-T G.722 core

The same as clauses 4.1-4.3 of ITU-T G.722 for lower band and higher band ADPCM decoding.

The ITU-T G.722 lower band decoder operates at 5 or 6 bits per sample (depending on the operating ITU-T G.722 mode). The ITU-T G.722 LB 5- or 6-bit index $I_L(n)$ decoding results in the signal $r_L(n)$ defined as (see clause 4.3.2 of ITU-T G.722):

$$r_L(n) = d_L(n) + s_L(n) \qquad n = 0,...,39 \qquad \text{(B.7-1)}$$

where $d_L(n)$ is the quantized difference signal and $s_L(n)$ is the predicted signal. In both ITU-T G.722 modes, $\hat{s}_{LB}(n) = r_L(n)$.

Similarly, the ITU-T G.722 higher band decoder operates at 2 bits per sample. The ITU-T G.722 HB 2-bit index $I_H(n)$ decoding results in the signal $r_H(n)$ defined as:

$$r_H(n) = d_H(n) + s_H(n) \qquad n = 0,...,39 \qquad \text{(B.7-2)}$$

where $d_H(n)$ is the quantized difference signal and $s_H(n)$ is the predicted signal. If ITU-T G.722 EL0 and G722EL1 are present, the signal $r_H(n)$ is further enhanced to obtain $\hat{s}_{HB}(n)$. This enhancement in ITU-T G.722 HB decoding is illustrated in Figure B.7-2.

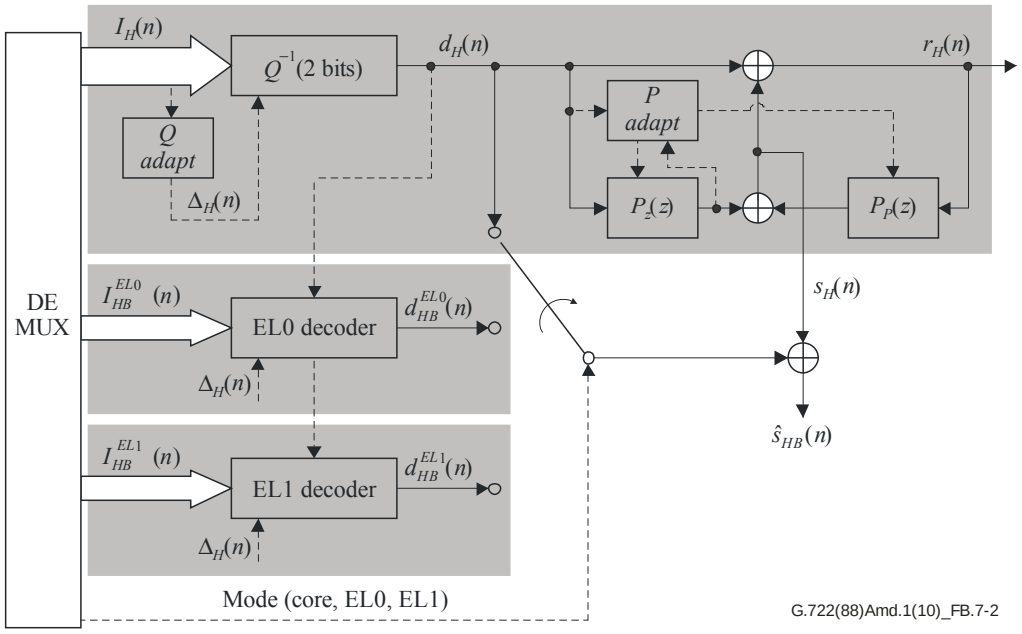


Figure B.7-2 – ITU-T G.722 HB decoding

B.7.2.2 Decoding G722EL0 layer

Before G722EL0 layer decoding, the SHB signal class F_{class} is decoded first (see clause B.7.3.1). G722EL0 layer is decoded for non-TRANSIENT frames.

In this case, 19 samples enhanced by G722EL0 are selected using the same procedure as described in clause B.6.4.3.3. One bit $I_{HB}^{EL0}(n)$ (0 or 1) is decoded for each selected sample, i.e.:

$$I_{HB}^{EL0}(n)=\begin{cases}1, & \text{if the decoded bit is 1 and } n \text{ is selected}\\ 0, & \text{if the decoded bit is 0 and } n \text{ is selected}\end{cases} \quad \text{(B.7-3)}$$

If G722EL1 is not available, the signal decoded using EL0 is given by $\hat{s}_{HB}(n)=d_{HB}^{EL0}(n)+s_H(n)$ with:

$$d_{HB}^{EL0}(n)=\begin{cases}\Delta_H(n)Q3[2I_H(n)+I_{HB}^{EL0}(n)], & \text{if } n \text{ is selected}\\ \Delta_H(n)Q2[I_H(n)], & \text{otherwise}\end{cases} \quad \text{(B.7-4)}$$

where $I_H(n)$ is the ITU-T G.722 HB 2-bit index, $I_{HB}^{EL0}(n)$ is the 1-bit index in the G722EL0 layer, $\Delta_H(n)$ is the step size of the higher band ADPCM decoder, $s_H(n)$ is the predicted signal from the main body of this Recommendation, and $Q3[.]$ and $Q2[.]$ are respectively defined in Table B.6-3 and Table B.6-2.

B.7.2.3 Decoding G722EL1 layer

For each sample n, one bit is decoded to obtain the binary index $I_{HB}^{EL1}(n)$ and the decoded signal becomes $\hat{s}_{HB}(n)=d_{HB}^{EL1}(n)+s_H(n)$ with:

$$d_{HB}^{EL1}(n)=\begin{cases}\Delta_H(n)Q4[4I_H(n)+2I_{HB}^{EL0}(n)+I_{HB}^{EL1}(n)], & \text{if } n \text{ is selected in EL0}\\ \Delta_H(n)Q3[2I_H(n)+I_{HB}^{EL1}(n)], & \text{otherwise}\end{cases} \quad \text{(B.7-5)}$$

where $I_H(n)$ is the ITU-T G.722 HB 2-bit index, $I_{HB}^{EL0}(n)$ is the 1-bit index of the G722EL0 layer, $\Delta_H(n)$ is the step size of the higher band ADPCM decoder, $s_H(n)$ is the predicted signal from the main body of this Recommendation, and $Q4[.]$ and $Q3[.]$ are defined respectively in Table B.6-5 and Table B.6-3.

B.7.2.4 High-pass post-processing

A DC offset of very small magnitude may appear in the decoded higher band. After the QMF synthesis, this introduces an 8-kHz component. To avoid this annoying high-frequency noise, a first-order pole/zero filter with a cut-off frequency of 50 Hz is used prior to the QMF. As shown in Figure B.7-3 this high-pass post processing is also applied in case of erased frames. This filter is given by:

$$H_{post}(z)=\frac{\frac{983}{1024}\left(1-z^{-1}\right)}{1-\frac{28835}{32768}z^{-1}} \quad \text{(B.7-6)}$$

B.7.2.5 Synthesis QMF

The same as clause 4.4 of ITU-T G.722.

Note that the synthesis QMF is adapted to operate with frames of 5 ms.

B.7.2.6 Higher band post-processor

For non-TRANSIENT frames, an MDCT domain post-processor may be applied to improve the quality of the signal reproduced by ITU-T G.722 in the 4.4-8 kHz range. The application of this

post-processing depends on the frequency amplitude parameters of the wideband, M_{wb}, and of the SHB, M_{shb}. These amplitude parameters are computed as follows:

First, an MDCT (see Equation (B.6-56)) is performed on the decoded wideband signal, i.e., the output signal of the QMF synthesis. Then the wideband amplitude parameter, M_{wb}, is computed from the obtained wideband MDCT coefficients $\hat{S}_{WB}(k), k = 0,...,79$. The SHB amplitude parameter M_{shb} is obtained as the average of the eight decoded spectral envelopes $\hat{f}_{env}(j)$.

$$M_{wb} = \sqrt{\frac{1}{80}\sum_{k=0}^{79} \hat{S}_{WB}{}^2(k)}, \qquad M_{shb} = \frac{1}{8}\sum_{j=0}^{7} \hat{f}_{env}(j) \tag{B.7-7}$$

The MDCT domain post-processor is applied to the last 36 wideband MDCT coefficients, $\hat{S}_{WB}(k),\ k = 44,\ldots,79$ when $M_{wb} > 0.8M_{shb}$.

Some spectrum having sufficient quality is amplified by multiplying by gain factors slightly larger than one. On the other hand, some spectrum having poor quality is multiplied by factors smaller than one or reduce it to a level below the estimated masking threshold. Three magnitude parameters are defined in order to estimate the quality, which are respectively called local masking magnitudes, noted $M_0(k), k = 44,...,79$, local masked magnitudes, noted $M_1(k), k = 44,...,79$, and overall average magnitude, noted M_{av}; the three parameters are estimated using the frequency coefficients; especially, the estimation of $M_0(k)$ and $M_1(k)$ are based on the perceptual masking effect.

In principle, if a frequency tone acts as a masking tone, this masking tone influences more the area above the tone than the area below it. The influence area of the masking tone is larger when it is located in a high frequency region than in a low frequency region. Usually, ordinary audio signals do not consist of just a tone. If the spectrum energy exists on the related band, the "perceptual loudness" at a specific frequency location depends not only on the energy at the location but also on the energy distribution around its location. The local masking magnitudes $M_0(k), k = 44,...,79$ are viewed as the "perceptual loudness" at location *k* and estimated by taking a weighted sum of the spectral magnitudes around it:

$$M_0(k) = \frac{\sum_{j=0}^{b_0^f(k)-1} w_0^f(k)\left|\hat{S}_{WB}(k-j)\right| + \sum_{j=1}^{b_0^l(k)-1} w_0^l(k)\left|\hat{S}_{WB}(k+j)\right|}{\sum_{j=0}^{b_0^f(k)-1} w_0^f(k) + \sum_{j=1}^{b_0^l(k)-1} w_0^l(k)} \qquad k = 44,...,79 \tag{B.7-8}$$

where $w_0^f(k)$ and $w_0^l(k)$ are the weighting windows, $b_0^f(k)$ is the first boundary of the masking MDCT coefficients and $b_0^l(k)$ is the last boundary of the masking MDCT coefficients.

Note that the weighting windows are asymmetric: the tail of the window is longer on the left side than on the right side of k. Furthermore, the window size is larger at the higher frequency area than at the lower frequency area. $w_0^f(k)$, $w_0^l(k)$, $b_0^f(k)$, and $b_0^l(k)$ are defined as follows:

$$w_0^{tmpf}(k) = \frac{4000(36-(k-44))+8000(k-44)}{36}\cdot 0.0005625+1$$

$$w_0^{tmpl}(k) = \frac{4000(36-(k-44))+8000(k-44)}{36}\cdot 0.00039375+1$$

,

$$b_0^f(k) = \min(\lfloor w_0^{tmpf}(k)\rfloor, k-44),$$

$$b_0^l(k) = \min(\lfloor w_0^{tmpl}(k)\rfloor, 36-(k-44)), \qquad \text{(B.7-9)}$$

$$w_0^f(k,j) = 1-\frac{0.75j}{w_0^{tmpf}(k)},\ j=0,...,b_0^f(k)-1,$$

$$w_0^l(k,j) = 1-\frac{0.75j}{w_0^{tmpl}(k)},\ j=1,...,b_0^l(k)-1,$$

where $k = 44,...,79$

Local masked magnitudes $M_1(k), k = 44,...,79$ are viewed as the estimated local "perceptual error floor". Since the encoder encodes signals in the perceptual domain, high energy frequency coefficients at the decoder side usually have low relative error but high absolute error. Low energy frequency coefficients at the decoder side have high relative error but low absolute error. The errors at different frequencies also perceptually influence each other in a way similar to the masking effect of a normal signal. Therefore, the local masked magnitudes $M_1(k)$ are estimated similarly to $M_0(k)$, as:

$$M_1(k) = \frac{\sum_{j=1}^{b_1^f(k)-1} w_1^f(k,j)\left|\hat{S}_{WB}(k-j)\right| + \sum_{j=1}^{b_1^l(k)-1} w_1^l(k,j)\left|\hat{S}_{WB}(k+j)\right|}{\sum_{j=0}^{b_1^f(k)-1} w_1^f(k,j) + \sum_{j=1}^{b_1^l(k)-1} w_1^l(k,j)} \qquad k = 44,...,79 \qquad \text{(B.7-10)}$$

This time, the shape of the weighting windows $w_1^f(k)$ and $w_1^l(k)$ is flatter and longer than $w_0^f(k)$ and $w_0^l(k)$. The weighting windows, $w_1^f(k)$ and $w_1^l(k)$, and the frequency boundaries, $b_1^f(k)$ and $b_1^l(k)$, are calculated as follows:

$$w_1^{tmpf}(k) = \frac{4000(36-(k-44))+8000(k-44)}{36}\cdot 0.001125+4,$$

$$w_1^{tmpl}(k) = \frac{4000(36-(k-44))+8000(k-44)}{36}\cdot 0.0007875+4,$$

$$b_1^f(k) = \min(\lfloor w_1^{tmpf}(k)\rfloor, k-44),$$

$$b_1^l(k) = \min(\lfloor w_1^{tmpl}(k)\rfloor, 36-(k-44)), \qquad \text{(B.7-11)}$$

$$w_1^f(k,j) = 1-\frac{0.5j}{w_1^{tmpf}(k)},\ j=0,...,b_1^f(k)-1,$$

$$w_1^l(k,j) = 1-\frac{0.5j}{w_1^{tmpl}(k)},\ j=1,...,b_1^l(k)-1,$$

$$\text{where } k=44,...,79$$

The overall average magnitude M_{av} is estimated as:

$$M_{av} = \frac{1}{36}\sum_{k=44}^{79}\left|\hat{S}_{WB}(k)\right| \qquad \text{(B.7-12)}$$

The ratio $M_0(k)/M_1(k)$ can somehow reflect the local relative perceptual quality at location *k*. Considering the possible influence of the overall average magnitude, one way to initialize the estimation of the gain factor along the frequency is to compare the local masked magnitude with the overall average magnitude to avoid a large gain factor. It can be described as:

$$g_{pp}(k) = \frac{M_0(k)}{0.75M_1(k)+0.25M_{av}} \qquad k=44,...,79 \qquad \text{(B.7-13)}$$

In order to avoid too much change in the overall energy after the post-processing, a gain normalization factor $g_{pp}^{'}$ is applied.

$$g_{pp}^{temp} = \frac{36M_{av}}{\sum_{k=44}^{79}\left(g_{pp}(k)\cdot\left|\hat{S}_{WB}(k)\right|\right)},$$

$$g_{pp}^{'} = \begin{cases} \dfrac{1.5}{g_{pp}^{\max}}\cdot(0.5+\dfrac{M_{av}}{16}), & \text{if } g_{pp}^{\max}\cdot g_{pp}^{temp}>1.5 \text{ and } M_{av}<32 \\ \dfrac{1.5}{g_{pp}^{\max}}, & \text{else if } g_{pp}^{\max}\cdot g_{pp}^{temp}>1.5 \\ g_{pp}^{temp}\cdot(0.5+\dfrac{M_{av}}{16}), & \text{else if } M_{av}<32 \\ g_{pp}^{temp}, & \text{otherwise} \end{cases} \qquad \text{(B.7-14)}$$

where the maximum gain factor is $g_{pp}^{\max} = \max_{j=44,...,79}(g_{pp}(k))$. Then the gain factor g_{pp} is multiplied by the gain normalization factor $g_{pp}^{'}$ to obtain the current normalized gain factor g_{pp}^{norm}:

$$g_{pp}^{norm}(k) = g_{pp}(k)\cdot g_{pp}^{'} \qquad k=44,...,79 \qquad \text{(B.7-15)}$$

The current normalized gain factors are smoothed with the ones from the previous frame and then applied to the MDCT coefficients,

$$g_{pp}^{norm\_sm}(k) = \frac{1}{2}\left(g_{pp}^{norm}(k) + g_{pp}^{norm\_sm}\right) \qquad k = 44,...,79 \qquad \text{(B.7-16)}$$

$$\hat{S}_{WB}^{pp}(k) = \begin{cases} g_{pp}^{norm\_sm}(k)\hat{S}_{WB}(k), & \text{if } g_{pp}^{norm\_sm}(k) < 1.15 \\ \hat{S}_{WB}(k), & \text{otherwise} \end{cases} \qquad k = 44,...,79 \qquad \text{(B.7-17)}$$

If the previous frame is TRANSIENT, $g_{pp}^{norm\_sm}(k)$ is set to $g_{pp}^{norm}(k)$ before performing Equations (B.7-16) and (B.7-17). For the frames not applying this post-processor, $g_{pp}^{norm\_sm}(k)$ is set to 1.

The newly obtained MDCT coefficients $\hat{S}_{WB}^{pp}(k)$ are transformed back to the decoded wideband time domain signal $\hat{s}_{WB}$ by the inverse MDCT as described in Equations (B.7-56) and (B.7-57). Note that those MDCT coefficients are also used for the frequency excitation generation in clause B.7.3.5.

B.7.3 SWBL0 decoder

First, the super higher band signal class is decoded. Then spectral envelopes or spectral/time envelopes are adaptively decoded depending on the above decoded SHB signal class. Four spectral envelopes and four time envelopes are decoded for TRANSIENT frame. For other cases (non-TRANSIENT frame), eight spectral envelopes are decoded and no time envelope is decoded. Frequency excitations are also generated according to the SHB signal class. Finally, the super higher band signal is decoded with the signal class, decoded envelopes and generated frequency excitations.

B.7.3.1 Decoding SHB signal class

Two bits are decoded from SWBL0 bitstream to get the SHB signal class according to Table B.6-8.

B.7.3.2 Decoding spectral envelope

Five bits are decoded to obtain the global gain, which is converted into the linear domain as follows:

$$\hat{g}_{glob} = 2^{g_{glob}} \qquad \text{(B.7-18)}$$

– If the current frame is TRANSIENT, four bits are decoded to obtain each normalized spectral envelope index $f_{rms\_idx\_t}(j)$, $j = 0,\cdots,3$. The normalized spectral envelope $\hat{f}_{rms}(j)$ is decoded using Equation (B.6-74).

  The decoded spectral envelope is as follows:

$$\hat{f}_{env}(j) = \hat{g}_{glob} \cdot \hat{f}_{rms}(j) \qquad j = 0,\cdots,3 \qquad \text{(B.7-19)}$$

– If the current frame is non-TRANSIENT, two vectors are decoded for the normalized spectral envelopes of the lower four sub-bands and the higher four sub-bands. For each vector, one bit is first decoded to obtain the codebook flag, $F_{cb}(i), i = 0,1$. Then the vector quantization index $f_{rms\_idx\_nt}(j)$, i=0, 1 is read from the bitstream. The normalized spectral envelope is decoded using Equation (B.6-75).

  The decoded spectral envelope is as follows:

$$\hat{f}_{env}(j) = \hat{g}_{glob} \cdot \hat{f}_{rms}(j) \qquad j = 0,\cdots,7 \qquad \text{(B.7-20)}$$

Additional spectral envelope adjustment is performed if the current frame is NORMAL or NOISE. The maximum, minimum, and average of the spectral envelope ($f_{env\_\max}$, $f_{env\_\min}$ and f_{env_avg}) are calculated as follows:

$$\begin{aligned} f_{env\_\max} &= \max_{j=0,\ldots,7}\left(\hat{f}_{rms}(j)\right), \\ f_{env\_\min} &= \min_{j=0,\ldots,7}\left(\hat{f}_{rms}(j)\right), \\ f_{env\_avg} &= \frac{1}{8}\sum_{j=0}^{7}\hat{f}_{rms}(j) \end{aligned} \qquad \text{(B.7-21)}$$

The spectral envelope of each sub-band is multiplied by 0.5, if conditions $(f_{env\_\max} - f_{env\_\min}) > 2.5$, $f_{env\_\min} < 12$ and $\hat{f}_{env}(j) < 0.4 f_{env\_avg}$ are satisfied,

$$\hat{f}_{env}(j) = 0.5\hat{f}_{env}(j) \qquad j = 0,\cdots,7 \qquad \text{(B.7-22)}$$

B.7.3.3 Decoding time envelope

If the current frame is TRANSIENT, four bits are decoded to obtain the index of each time envelope, $t'_{rms}(j)$. This envelope is converted into the linear domain as follows:

$$\hat{t}'_{rms}(j) = 2^{t'_{rms}(j)} \qquad j = 0,\cdots,3 \qquad \text{(B.7-23)}$$

The linear domain time envelope of the previous sub-frame is saved as $\hat{t}'^{(-1)}_{rms}$. $\hat{t}'^{(-1)}_{rms}$ is set to zero for the first frame. A time envelope adjustment flag bit is decoded and set as F_{tenv}. Time envelope denormalization is performed after the frequency domain processing in clause B.7.6.

B.7.3.4 Signal class counting for frequency excitation selection

Counter $\hat{c}_{class}$ is a counter for the SHB signal class to help frequency excitation selection in the decoder side. It is initialized to zero. Counter $\hat{c}_{class}$ is updated according to the SHB signal class, as:

$$\hat{c}_{class} = \begin{cases} 0, & \text{if} \quad F_{class} = TRANSIENT \\ \hat{c}_{class} + 1, & \text{else if} \quad F_{class} = HARMONIC \\ \max(\hat{c}_{class} - 1, 0), & \text{otherwise} \end{cases} \qquad \text{(B.7-24)}$$

B.7.3.5 Frequency excitation generation

The wideband MDCT coefficients after the higher band post-processing $\hat{S}^{PP}_{WB}(k)$ are used for SHB frequency excitation generation in SWBL0 depending on the decoded SHB signal class, as follows.

The base frequency excitation signal $\hat{S}_{exc\_base}(k)$ is generated from the wideband MDCT coefficients or from a random noise.

If the current frame is NOISE and the previous frame is not HARMONIC,

$$\hat{S}_{exc\_base}(k) = S_{noise}(k) \qquad k = 0,\cdots,63 \qquad \text{(B.7-25)}$$

where

$$S_{noise}(k) = \frac{\lambda_{seed}}{32768} \text{ and } \lambda_{seed} = 12345\lambda_{seed} + 20101 \qquad k = 0,\cdots,63 \qquad \text{(B.7-26)}$$

Parameter λ_{seed} is initialized as 21211 and updated for each MDCT coefficient. It is noted that $S_{noise}(k)$ is calculated for every frame.

For other frames,

$$\hat{S}_{exc\_base}(k) = \hat{S}_{WB}^{PP}(k) \qquad k = 0,\cdots,79 \qquad \text{(B.7-27)}$$

If both signal classes of the current frame and the previous frame are NORMAL/NOISE and $\hat{c}_{class}$ equals to zero, the base frequency excitation is updated as follows:

$$\hat{S}_{exc\_base}(k) = \hat{S}_{exc\_base}(k+32) \qquad k = 0,\cdots,31 \qquad \text{(B.7-28)}$$

For this case, MDCT coefficients in the frequency range 3.2 kHz-6.4 kHz are used for the base frequency excitation.

B.7.3.6 Frequency excitation normalization and spectral envelope denormalization

Firstly, the spectral envelope is adjusted according to the SHB signal class. Then the base frequency excitation signal is normalized to remove the envelope information. Finally, the spectral envelope is applied to the normalized excitation signal.

For TRANSIENT frames, $\hat{f}_{env}(j)$ is set to $0.02\hat{g}_{glob}$ when $\hat{f}_{env}(j)$ equals to zero. The decoded super higher band frequency coefficients are calculated as follows:

$$\hat{S}_{SHB}^{BWE}(16j+i) = \hat{S}_{exc\_base}(16j+i)\cdot \hat{f}_{env}(j)\cdot \sqrt{\frac{N_{swbcf}(j)}{\left(\sum_{k=b_{swb}(j)}^{b_{swb}(j+1)-1} \hat{S}_{exc\_base}{}^{2}(k)\right)+\varepsilon_{rms}}} \qquad \text{(B.7-29)}$$

where $i = 0,\cdots,15$ and $j = 0,\cdots,3$.

For non-TRANSIENT frames, the spectral envelope of the current frame is smoothed with the spectral envelope of the previous frame $\hat{f}_{env}^{(-1)}(j)$, $j = 0,\cdots,7$, when the following conditions are satisfied:

$$\left|\hat{f}_{env}(j) - \hat{f}_{env}^{(-1)}(j)\right| \le 3\hat{g}_{glob} \text{ and } F_{class}^{(-1)} \ne TRANSIENT \qquad \text{(B.7-30)}$$

In this case, the smoothed spectral envelope, $\hat{f}_{env\_sm}(j)$, is calculated as follows:

$$\hat{f}_{env\_sm}(j) = w_{fenv}\cdot \hat{f}_{env}(j) + (1-w_{fenv})\cdot \hat{f}_{env}^{(-1)}(j) \qquad j = 0,\cdots,7 \qquad \text{(B.7-31)}$$

where

$$w_{fenv} = \begin{cases} 0.7, & \text{if} \quad F_{class} = NORMAL \text{ or } NOISE \\ 0.5, & \text{otherwise} \end{cases} \qquad \text{(B.7-32)}$$

The decoded super higher band frequency coefficients are calculated as follows:

$$\hat{S}_{SHB}^{BWE}(8j+i) = \hat{S}_{exc\_base}(8j+i)\cdot \hat{f}_{env\_sm}(j)\cdot \sqrt{\frac{N_{swbcf}(j)}{\left(\sum_{k=b_{swb}(j)}^{b_{swb}(j+1)-1} \hat{S}_{exc\_base}{}^{2}(k)\right)+\varepsilon_{rms}}} \qquad \text{(B.7-33)}$$

where $i = 0,\cdots,7$ and $j = 0,\cdots,7$.

Finally, $\hat{f}_{env}(j)$ is saved for next frame. In case of frame erasure, the spectral envelope of the previous frame $\hat{f}_{env}^{(-1)}(j)$, $j = 0,\cdots,7$, is attenuated by a factor of 0.85.

Note that in case that only SWBL0 is received (MODE_R1sm), the spectrum $\hat{S}_{SHB}^{BWE}(k)$ is equal to the output decoded spectrum $\hat{S}_{SHB}(k)$.

B.7.4 SWBL1 and SWBL2 decoder

The decoded normalized spectral envelope from SWBL0 is used to set the perceptual importance order of sub-bands $\Omega_b(j)$.

The decoding of the SWBL1 and SWBL2 spectrum coefficients depends on the SHB mode corresponding to the sparseness of the spectrum where the SHB mode information is obtained from the SWBL1 bitstream and the signal class extracted from the SWBL0 bitstream.

When both signal classes of the current frame and the previous one are NORMAL/NOISE, the SHB mode 0 for the sparse spectrum is selected on the condition that the SHB mode flag is 0, and the SHB mode 1 for the non-sparse spectrum is chosen if the SHB mode is 1. For other cases of the signal classes, the SHB mode 2 is applied.

B.7.4.1 Decoding the SHB in mode 0

The decoding in SHB mode 0 starts with reading and decoding the AVQ parameters to obtain the decoded spectrum $\hat{S}'(k)$; for details, see clause B.7.4.9.

B.7.4.1.1 Problematic frame information decoding

The detection flag f_0 is read from the SWBL1 bitstream if there is at least one unused AVQ bit in this layer and both SWBL1 and SWBL2 are received. The detection flag f_0 is used to form the MDCT spectrum (see clause B.7.4.5). The number of unused AVQ bits in SWBL1 is consequently reduced by one. If only SWBL1 is received, the detection flag is set to $f_0 = 0$.

B.7.4.1.2 Filling zero sub-bands

If both SWBL1 and SWBL2 are received, the flag $f_0 = 0$, and there are at least four remaining unused AVQ bits in SWBL1 or in SWBL2, the base spectrum $\hat{S}'_{base}(k)$ is computed as described in clause B.6.7.4.3. Then,

– if there are at least four remaining unused AVQ bits in SWBL1 and in SWBL2, both lags δ_1 and δ_2 are read from the SWBL1 and SWBL2 bitstreams respectively. Finally, vectors $\hat{S}'_{0sb1}(i)$ (if $\delta_1 < 15$) and $\hat{S}'_{0sb2}(i)$ (if $\delta_2 < 15$) are computed using Equations (B.6-89) and (B.6-92);
– if there is only one SWB layer with at least four remaining unused AVQ bits, only lag δ_1 is read from this SWBL1 or SWBL2 bitstream. Finally, if $\delta_1 < 15$, vector $\hat{S}'_{0sb1}(i)$ is computed using Equation (B.6-89).

B.7.4.1.3 Backward reordering and denormalization

The same as described in clause B.6.7.4.4.

B.7.4.2 Decoding the SHB in mode 1

B.7.4.2.1 Recalculation of the decoded spectral envelope

Before the AVQ decoding (for details see clause B.7.4.9) of the SHB mode 1, the envelopes of the error spectrum, $\hat{f}_{rms}^{err}(i)$, which are required for the denormalization of the AVQ in place of the decoded normalized envelopes, $\hat{f}_{rms}(i)$, are computed using Equation (B.6-94).

B.7.4.2.2 Backward reordering and denormalization

The same as clause B.6.7.5.3.

B.7.4.2.3 Calculation of the decoded normalized spectra

The recalculated decoded envelopes $\hat{f}_{rms}^{'}(j)$ are computed using Equation (B.6-96) and then the decoded normalized spectra, $\hat{S}_{SHB}^{norm}(k)$, of which the sub-band is not a zero sub-band, are extracted as:

$$\hat{S}_{SHB}^{norm}(8j+i)=\begin{cases}\mathrm{sgn}\left(\hat{S}_{exc\_base}(8j+i)\right)\cdot\hat{f}_{rms}^{'}, & \text{if } \hat{S}_{SHB}^{err}(8j+i)=0,\\ \mathrm{sgn}\left(\hat{S}_{SHB}^{err}(8j+i)\right)\cdot\left(\left|\hat{S}_{SHB}^{err}(8j+i)\right|+0.5\hat{f}_{rms}(i)\right), & \text{otherwise}\end{cases} \quad \text{(B.7-34)}$$

where $i = 0,\cdots,7$ and $j = 0,\cdots,7$, for j not corresponding to a zero sub-band.

B.7.4.2.4 Problematic frame information decoding

The detection flag f_1 is read from the SWBL1 bitstream if there is at least one unused AVQ bit in this layer and both SWBL1 and SWBL2 are received. The number of bits (1 or 2) used to quantize flag f_1 is deducted from the number of remaining unused AVQ bits in SWBL1 (see clause B.7.4.9). Then, the value of flag f_1 (f_1 =0, 1, 2, 3) is used to update the spectral envelope in all zero sub-bands of the SHB spectrum as follows:

$$\hat{f}_{rms}(j)=2^{-f_1}\cdot\hat{f}_{rms}(j) \quad \text{(B.7-35)}$$

where $\hat{f}_{rms}(j)$ is the decoded normalized spectral envelope for any j corresponding to a zero sub-band. The number of unused AVQ bits in SWBL1 is finally reduced by one or two. Note that if only SWBL1 is received, the detection flag is set to $f_1 = 0$.

B.7.4.2.5 Filling zero sub-bands

In case both SWBL1 and SWBL2 are received and there are at least four unused bits in SWBL1 or SWBL2, the lags δ_1 and δ_2 are read from the SWBL1 and SWBL2 bitstreams, respectively. Furthermore, the base spectrum $\hat{S}'_{base}(k)$ is obtained as described in clause B.6.7.5.6. Then vectors $\hat{S}'_{0sb1}(i)$ (if $\delta_1 < 15$) and $\hat{S}'_{0sb2}(i)$ (if $\delta_2 < 15$) are computed using (B.6-101). Finally, vectors $\hat{S}'_{0sb1}(i)$ and $\hat{S}'_{0sb2}(i)$ are used to fill zero sub-bands in the spectrum $\hat{S}_{SHB}^{abs}(k)$ to form the normalized optimized decoded SHB spectrum $\hat{S}_{SHB}^{norm}(k)$.

B.7.4.2.6 Sign information decoding

When the SHB mode is 1 and the number of unused remaining bits of the AVQ in the two SWB layers is more than one, the sign information is extracted from the SWBL1 and SWBL2 bitstreams, $I_{swbl1\_sign}(u),\ u=0,...,B_{swbl1\_unused}-1$ and $I_{swbl2\_sign}(u),\ u=0,...,B_{swbl2\_unused}-1$, respectively. B_{swbl1_unused} and B_{swbl2_unused} are the number of remaining bits from the SWBL1 and SWBL2 layers. For incrementing *k* from 0 to 63, on the condition that $\hat{S}^{err}_{SHB}(8j+i)=0$ and $j\left(=\lfloor k/8 \rfloor\right)$ corresponds to the sub-band coded in SWBL1, the decoded normalized spectrum, $\hat{S}^{norm}_{SHB}(k)$, is modified using $I_{swbl1\_sign}(u)$:

$$\hat{S}^{norm}_{SHB}(k)=\begin{cases}-\left|\hat{S}^{norm}_{SHB}(k)\right|, & \text{if } I_{swbl1\_sign}(u)=0,\\ \left|\hat{S}^{norm}_{SHB}(k)\right|, & \text{otherwise,}\end{cases} \tag{B.7-36}$$

and then *u* is incremented by one. The above calculation continues until there are no remaining bits in the SWBL1 layer. After that, on the condition that $\hat{S}^{err}_{SHB}(8j+i)=0$ and $j\left(=\lfloor k/8 \rfloor\right)$ corresponds to the sub-band coded in SWBL1, the decoded spectrum is also modified using $I_{swbl2\_sign}(v)$:

$$\hat{S}^{norm}_{SHB}(k)=\begin{cases}-\left|\hat{S}^{norm}_{SHB}(k)\right|, & \text{if } I_{swbl2\_sign}(v)=0,\\ \left|\hat{S}^{norm}_{SHB}(k)\right|, & \text{otherwise,}\end{cases} \tag{B.7-37}$$

Then *v* is incremented by one, where *u* and *v* are defined in clause B.6.7.5.7. The above calculation continues until there are no bits remaining in the SWBL2 layer.

B.7.4.3 Decoding the SHB in mode 2

Decoding the SHB in mode 2 is the same as the decoding in SHB mode 0 described in clause B.7.4.1, except that, for SHB mode 2 bitstreams, the one-bit flag for the decision of sparseness is not included.

B.7.4.4 Gain adjustment decoding

The decoded value of the adjusted gain for SWBL1 layer is computed from its index, I_{gc}, by

$$\hat{g}_{adj}=\hat{g}_{correct}(I_{gc})\cdot\hat{g}_{glob} \tag{B.7-38}$$

where

$$\hat{g}_{correct}(i)=\begin{cases}0.2i & \text{if } g_{glob}=0 \text{ and } 0\le i\le 4\\ 2^{\frac{k_i}{8}},\ k_i=\{-7,-5,-3,-2,-1,0,1,3\} & \text{otherwise,}\end{cases} \tag{B.7-39}$$

B.7.4.5 Decoding of MDCT coefficients

The AVQ decoded SHB spectrum $\hat{S}^{AVQ}_{SHB}$ is obtained on a sub-band basis depending on the SHB mode and the number of decoded layers. If all SWB layers are received and decoded, then

$$\hat{S}^{AVQ}_{SHB}(8j+i)=\hat{g}_{adj}\cdot\hat{S}^{norm}_{SHB}(8j+i) \tag{B.7-40}$$

where $i = 0,\cdots,7$ and $j = 0,\cdots,7$, for *j* corresponding to sub-bands coded in SWBL1, and

$$\hat{S}^{AVQ}_{SHB}(8j+i)=\hat{g}_{glob}\cdot\hat{S}^{norm}_{SHB}(8j+i) \tag{B.7-41}$$

for j corresponding to sub-bands coded in SWBL2. Note that sub-bands coded in SWBL2 also comprise the sub-bands filled using the technique described in clauses B.7.4.1.2 and B.7.4.2.5, respectively.

Finally, the zero sub-bands are dealt with depending on the SHB mode, as follows.

For all j corresponding to zero sub-bands:

– If SHB mode equals 1, then

$$\hat{S}_{SHB}^{AVQ}(8j+i) = \mathrm{sgn}\left(\hat{S}_{exc\_base}(8j+i)\right)\cdot \hat{g}_{glob}\cdot \hat{f}_{rms}(j) \qquad \text{(B.7-42)}$$

– Otherwise (SHB mode is other than 1),

• if the detection flag $f_0 = 1$, then

$$\hat{S}_{SHB}^{AVQ}(8j+i) = 0.1\cdot \mathrm{sgn}\left(\hat{S}_{exc\_base}(8j+i)\right)\sqrt{\frac{\hat{g}_{glob}}{8}\sum_{i=0}^{7}\hat{S}_{exc\_base}(8j+i)} \qquad \text{(B.7-43)}$$

• otherwise (the detection flag $f_0 = 0$), the SWBL0 output spectrum $\hat{S}_{SHB}^{BWE}(k)$ is used to fill zero sub-bands in spectrum $\hat{S}_{SHB}^{AVQ}(k)$.

B.7.4.6 Gradient adjustment of the spectrum decoding

In case that there are still unused AVQ bits in at least one of the SWB layers and the SHB mode is not one, these bits represent the indices of the gradient adjustment factor. For each of two SWB layers, the indices of the gradient adjustment factors are de-multiplexed from each bitstream by one or two bits per sub-band as long as there are remaining unused AVQ bits.

For the SWBL1 layer, the gradient adjustment is done in the following steps:

– The adjustment is performed in the perceptual importance order of sub-bands, $\Omega_b(j)$, that have been coded in the SWBL1 layer. The number of bits allocated for each sub-band is calculated using Equation (B.6-108).

– Based on the results of the bit allocation, the index of the $\Omega_b(j)$-th sub-band, $I_{swbl1\_grd}(\Omega_b(j))$, is sequentially de-multiplexed.

– The adjusted decoded SHB spectrum of b (=$\Omega_b(j)$)-th sub-band, $\hat{S}_{SHB}^{adj}(8b+i)$, is computed as follows:

$$\hat{S}_{SHB}^{adj}(8b+i) = \gamma_{grd}(I_{swbl1\_grd}(b), i)\cdot \hat{S}_{SHB}^{AVQ}(8b+i) \qquad \text{(B.7-44)}$$

where $\hat{S}_{SHB}^{AVQ}(8b+i)$ is the decoded SHB spectrum of b-th sub-band and $\gamma_{grd}(q,i)$ is the gradient adjustment factor, given in Equation (B.6-110). This gradient adjustment is continued till there are no more bits. After the above processing is finished for the SWBL1 layer, the same procedure is performed for bitstream I_{swbl2_grd}, for the adjustment of the sub-bands corresponding to the SWBL2 layer.

B.7.4.7 BWE/AVQ adaptation

The MDCT coefficients decoded by AVQ are used to replace the BWE coefficients in higher layers SWBL1 and SWBL2. For empty sub-bands, i.e., $\hat{S}_{SHB}^{norm}(8j+i) = 0$ for all $i = 0,...,7$ in j-th sub-band, or the zero MDCT coefficients in non-empty sub-bands, BWE/AVQ adaptation is performed to reduce the perceptual noise and improve the subjective quality especially for layer SWBL1. Empty sub-bands often occur when only decoding SWBL1 layer.

According to the correlation between the current frame and the previous frames, first select at least two MDCT coefficients which have higher correlation with the MDCT coefficients to be adjusted. The selected MDCT coefficients and the MDCT coefficients to be adjusted are weighted to obtain the predicted values of current MDCT coefficients. Then, the predicted values combined with the same sign before the adjustment are decoded as the adjusted MDCT coefficients. The obtained MDCT coefficients as a result of BWE/AVQ adaptation are $\hat{S}_{SHB}^{adp}(k)$.

1) Current and previous frames are HARMONIC

If both the current frame and the previous frame are HARMONIC, there should be high correlations between those two frames. In this case, the BWE/AVQ adapted MDCT coefficients $\hat{S}_{SHB}^{adp(m-1)}(k)$ in the previous (m – 1)-th frame and the AVQ decoded MDCT coefficients in the previous (m – 1)-th and (m – 2)-th frames are used for prediction.

If j-th sub-band is empty in the current frame and was not empty in the previous frame, the following BWE/AVQ adaptation operation is performed:

$$\hat{S}_{SHB}^{adp}(k) = \mathrm{sgn}\left(\hat{S}_{SHB}^{adj}(k)\right) \cdot e_{adp}(k) \tag{B.7-45}$$

where $e_{adp}^{(m)}(k)$ is the predicted MDCT magnitude obtained by:

$$e_{adp}(k) = \begin{cases} 0.375\left|\hat{S}_{SHB}^{AVQ(m-1)}(k)\right| + 0.375\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right| + 0.25\left|\hat{S}_{SHB}^{adj(m)}(k)\right| \\ \qquad \text{if } \left|\hat{S}_{SHB}^{AVQ(m-2)}(k)\right| > \left|\hat{S}_{SHB}^{AVQ(m-1)}(k)\right| + \left|\hat{S}_{SHB}^{adp(m-1)}(k)\right|, \\ 0.125\left|\hat{S}_{SHB}^{AVQ(m-2)}(k)\right| + 0.375\left|\hat{S}_{SHB}^{AVQ(m-1)}(k)\right| + 0.375\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right| + 0.125\left|\hat{S}_{SHB}^{adj(m)}(k)\right| \\ \qquad \text{otherwise,} \end{cases} \tag{B.7-46}$$

2) Current or previous frame is TRANSIENT

If the current or the previous frame is TRANSIENT, there should be low correlation between those two frames. For non-empty sub-bands, MDCT coefficients are weighted by their neighbouring MDCT coefficients. The BWE/AVQ adapted MDCT coefficients $\hat{S}_{SHB}^{adp}(k)$ are obtained:

$$\hat{S}_{SHB}^{adp}(k) = \begin{cases} \mathrm{sgn}\left(\hat{S}_{SHB}^{adj}(k)\right) \cdot \min\left(\theta_{adp}(k),\ e_{adp}(k)\right) & \text{if } \hat{S}_{SHB}^{norm}(k) = 0 \\ \hat{S}_{SHB}^{adj}(k) & \text{otherwise} \end{cases} \qquad k = 0,...,63 \tag{B.7-47}$$

where the rectification threshold is obtained by $\theta_{adp}(k) = 0.6 \cdot \hat{g}_{glob}\hat{f}_{rms}(\lfloor k/16 \rfloor)$ and adapted MDCT magnitude $e_{adp}(k)$ is obtained by:

$$e_{adp}(k) = \begin{cases} 0.5\left(\left|\hat{S}_{SHB}^{AVQ}(k+1)\right| + \left|\hat{S}_{SHB}^{adj}(k)\right|\right), & k = 0 \\ 0.25\left|\hat{S}_{SHB}^{AVQ}(k-1)\right| + 0.25\left|\hat{S}_{SHB}^{AVQ}(k+1)\right| + 0.5\left|\hat{S}_{SHB}^{adj}(k)\right|, & 0 < k < 63 \end{cases} \tag{B.7-48}$$

3) Other cases

For other cases, there should be medium correlations between the current and the previous frames. In this case, the BWE/AVQ adapted MDCT coefficients $\hat{S}_{SHB}^{adp(m-1)}(k-1)$, $\hat{S}_{SHB}^{adp(m-1)}(k)$ and $\hat{S}_{SHB}^{adp(m-1)}(k+1)$ in the previous (m – 1)-th frames are used for prediction.

In case SHB mode is 1, the MDCT coefficients in the current frames are adapted:

$$\hat{S}_{SHB}^{adp}(k)=\begin{cases}\mathrm{sgn}\left(\hat{S}_{SHB}^{adj}(k)\right)\cdot\min\left(\theta_{adp}(k),\ e_{adp}(k)\right) & \text{if } \hat{S}_{SHB}^{AVQ}(k)=0\\ \hat{S}_{SHB}^{adj}(k) & \text{otherwise}\end{cases} \qquad k=0,...,63 \qquad \text{(B.7-49)}$$

where the rectification threshold is similarly obtained by $\theta_{adp}(k)=0.6\cdot\hat{g}_{glob}\hat{f}_{rms}(\lfloor k/8\rfloor)$.

If j-th sub-band is empty, the adapted MDCT magnitudes are calculated as:

$$e_{adp}(k)=\begin{cases}0.35\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right|+0.45\left|\hat{S}_{SHB}^{adj(m)}(k)\right| & \\ \quad+0.1\left|\hat{S}_{SHB}^{adp(m-1)}(k+1)\right|+0.1\left|\hat{S}_{SHB}^{adj(m)}(k+1)\right|, & k=0\\ 0.1\left|\hat{S}_{SHB}^{adp(m-1)}(k-1)\right|+0.1\left|\hat{S}_{SHB}^{adj(m)}(k-1)\right|+0.25\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right| & \\ \quad+0.35\left|\hat{S}_{SHB}^{adj(m)}(k)\right|+0.1\left|\hat{S}_{SHB}^{adp(m-1)}(k+1)\right|+0.1\left|\hat{S}_{SHB}^{adj(m)}(k+1)\right|, & k=1,...,62\\ 0.25\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right|+0.35\left|\hat{S}_{SHB}^{adj(m)}(k)\right| & \\ \quad+0.15\left|\hat{S}_{SHB}^{adp(m-1)}(k-1)\right|+0.25\left|\hat{S}_{SHB}^{adj(m)}(k-1)\right|, & k=63\end{cases} \qquad \text{(B.7-50)}$$

If j-th sub-band is not empty, the adapted MDCT magnitudes are calculated as:

$$e_{adp}(k)=\begin{cases}0.15\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right|+0.65\left|\hat{S}_{SHB}^{adj(m)}(k)\right| & \\ \quad+0.1\left|\hat{S}_{SHB}^{adp(m-1)}(k+1)\right|+0.1\left|\hat{S}_{SHB}^{adj(m)}(k+1)\right|, & k=0\\ 0.05\left|\hat{S}_{SHB}^{adp(m-1)}(k-1)\right|+0.05\left|\hat{S}_{SHB}^{adj(m)}(k-1)\right|+0.15\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right| & \\ \quad+0.65\left|\hat{S}_{SHB}^{adj(m)}(k)\right|+0.05\left|\hat{S}_{SHB}^{adp(m-1)}(k+1)\right|+0.05\left|\hat{S}_{SHB}^{adj(m)}(k+1)\right|, & k=1,...,62\\ 0.15\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right|+0.65\left|\hat{S}_{SHB}^{adj(m)}(k)\right| & \\ \quad+0.05\left|\hat{S}_{SHB}^{adp(m-1)}(k-1)\right|+0.15\left|\hat{S}_{SHB}^{adj(m)}(k-1)\right|, & k=63\end{cases} \qquad \text{(B.7-51)}$$

In case SHB mode is not 1, the adaptation is not performed:

$$\hat{S}_{SHB}^{adp}(k)=\hat{S}_{SHB}^{adj}(k) \qquad k=0,...,63 \qquad \text{(B.7-52)}$$

B.7.4.8 Spectrum post-processor

When the SHB mode equals one, a spectrum post-processor is applied to smooth the decoded SHB spectrum. Using the current frame and the previous frame, the decoded SHB spectrum $\hat{S}_{SHB}(k)$ is updated for incrementing *k* from 0 to 63 for $\hat{S}_{SHB}^{adp(m-2)}(k)\neq 0$:

$$\hat{S}_{SHB}^{(m)}(k)=\begin{cases}\mathrm{sgn}\left(\hat{S}_{SHB}^{adp(m)}(k)\right)\cdot\left(0.85\left|\hat{S}_{SHB}^{adp(m)}(k)\right|+0.15\left|\hat{S}_{SHB}^{adp(m-1)}(k)\right|\right) & \text{if } \hat{S}_{SHB}^{adp(m-1)}(k)\neq 0,\\ \hat{S}_{SHB}^{adp(m)}(k) & \text{otherwise}\end{cases} \qquad k=0,...,63 \qquad \text{(B.7-53)}$$

If the SHB mode is not equal to 1,

$$\hat{S}_{SHB}^{(m)}(k)=\hat{S}_{SHB}^{adp(m)}(k) \qquad k=0,...,63 \qquad \text{(B.7-54)}$$

and all previously adapted MDCT coefficients $\hat{S}_{SHB}^{adp(m-1)}(k)$ $k = 0,...,63$ are initialized to zero.

B.7.4.9 AVQ decoding

The reading of the AVQ parameters from the bitstream is complementary to the insertion described in clause B.6.7.10.3. The codebook numbers n_j are used to estimate the actual bit-budget needed to encode AVQ parameters at the decoder and the number of unused AVQ bits is computed as a difference between the allocated and actual bit budgets.

Parameter decoding involves decoding the AVQ parameters describing each 8-dimensional quantized sub-bands $\hat{\mathbf{S}}'(8j)$ of the quantized spectrum $S'(k)$. The $\hat{\mathbf{S}}'(8j)$ in SWBL1 and SWBL2 comprise respectively three and four sub-bands, each of eight samples. The decoded AVQ parameters for each sub-band $\hat{\mathbf{S}}'(8j)$ comprise:

– the codebook number n_j,

– the vector index I_j,

– and, if the codevector (i.e., lattice point) is not in a base codebook, the Voronoi index $\mathbf{I}_j^v$.

The unary code for the codebook number n_j, is first read from the bitstream and n_j is determined (see Table B.6-10). From the codebook number n_j, the base codebook and the Voronoi extension order r_j^v are then obtained. If $n_j < 5$, there is no Voronoi extension ($r_j^v = 0$) and the base codebook is Q_{n_j}. If $n_j \geq 5$ the base codebook is either Q_3 (n_j even) or Q_4 (n_j odd) and the Voronoi order (1 or 2) is also determined ($r_j^v = 1$ if $n_j < 7$; $r_j^v = 2$, otherwise).

Then, if $n_j > 0$, the vector index I_j, coded on $4n_j$ bits is read from the bitstream and the base codevector $\mathbf{z}_j$ is decoded.

After the decoding of the base codevector, if the Voronoi order r_j^v is greater than 0, the Voronoi extension index $\mathbf{I}_j^v$ is decoded to obtain the Voronoi extension vector $\mathbf{v}_j$. The number of bits in each component of index vector $\mathbf{I}_j^v$ is given by the Voronoi extension order r_j^v, and the scaling factor M_j^v of the Voronoi extension is given by $M_j^v = 2^{r_j^v}$.

Finally, from the scaling factor M_j^v, the Voronoi extension vector $\mathbf{v}_j$ and the base codebook vector $\mathbf{z}_j$, each 8-dimensional AVQ sub-band $\hat{\mathbf{S}}'(8j)$ is computed as:

$$\hat{\mathbf{S}}'(8j) = M_j^v \cdot \mathbf{z}_j + \mathbf{v}_j \qquad \text{(B.7-55)}$$

B.7.4.9.1 De-indexing of the codevector in the base codebook

The index decoding of the codevector $\mathbf{z}_j$ is done in several steps. First, the absolute leader and its offset are identified by comparing the index with the offset in the look-up table. The offset is subtracted from the index to produce a new index. From this index, the sign index and the absolute vector index are extracted. The sign index is decoded and the sign vector is obtained. The absolute vector index is decoded by using a multi-level permutation-based index decoding method and the absolute vector is obtained. Finally, the decoded vector is reconstructed by combining the sign vector with the absolute vector.

B.7.4.9.1.1 Sign decoding

The sign vector is obtained by extracting from left to right all the sign bits for non-zero elements in the absolute vector. The bit number of the sign code is read from the (S_n) in Table B.6-11. If the bit number of the sign index is not equal to the number of the non-zero elements in the decoded absolute vector, the sign of the last non-zero element is recovered.

B.7.4.9.1.2 Decoding of the absolute vector and of its position vector

The decoding method of the absolute vector index is described as follows:

1) The absolute vector index is decomposed into several mid-indices for each level from lowest level to highest level. The absolute vector index is the starting value for the lowest level. The mid-index of each lower level is obtained by dividing the absolute vector index by the possible index value count, $C_{m_{n-1}}^{m_n}$, the quotient is the absolute vector index for the next lower level. The remainder is the middle index, $I_{mid,n}$, for the current level.

2) The $I_{mid,n}$ of each lower level is decoded based on a permutation and combination function and the position vector of each lower level vector related to its upper level vector is obtained.

3) Finally, one-by-one from the lowest level to the highest level, each lower level absolute vector is used to partly replace the upper level absolute vector elements according to the position parameter. The highest level vector is the decoded output absolute vector.

B.7.4.9.1.3 Position vector decoding

To obtain the position vector from the middle index in each lower level, the algorithm uses a permutation and combination procedure to estimate the position sequence. The procedure is as follows:

1) Increment the *pos* value beginning from zero, until $I_{mid,n}$ is not more than $C_{m_{n-1}}^{m_n} - C_{m_{n-1}-pos}^{m_n}$.

2) Let $q_0 = pos - 1$ be the first position, and subtract $C_{m_{n-1}}^{m_n} - C_{m_{n-1}-q_0}^{m_n}$ from the I_{mid}.

3) Increase *pos*, beginning from $q_{i-1}+1$, until I_{mid} is not more than $C_{m_{n-1}-q_{i-1}-1}^{m_n-i} - C_{m_{n-1}-pos}^{m_n-i}$, where q_{i-1} is the position decoded at the previous step.

4) Let $q_i = pos - 1$ be the position number i, and subtract $C_{m_{n-1}-q_{i-1}-1}^{m_n-i} - C_{m_{n-1}-q_i}^{m_n-i}$ from the I_{mid}.

5) Repeat steps 3 and 4 until all positions are decoded for the current level position sequence.

B.7.4.9.1.4 Absolute vector decoding

For the lowest level, the absolute vector only includes one type of element whose value can be obtained from the decomposition order column in Table B.6-11. The lowest level absolute vector is passed to the next level and at the next step another type of element is added. This new element is obtained from the decomposition order column in Table B.6-11. This procedure is repeated until the highest level is reached.

B.7.4.9.1.5 Construction of the output codevector in base codebook

Constructing the 8-dimensional output codevector in the base codebook is the final step of the decoding procedure. The codevector $\mathbf{z}_j$ is obtained by combining the sign vector with the absolute vector. If the bit number of the sign index is not equal to the number of the non-zero elements in the decoded absolute vector, the sign of the last non-zero element is recovered. The recovery rule, based on the RE_8 lattice property, is as follows: if the sum of all output vector elements is not an integer multiple of four, the sign of the last element is set to negative.

B.7.5 Inverse MDCT and overlap-add

Before applying inverse MDCT, the last 20 super higher band frequency coefficients are set to zero to obtain 14 kHz bandwidth output. Then, the super higher band frequency coefficients are transformed to the time domain by an inverse MDCT transform:

$$\hat{s}_{SHB}^{cur}(n) = \sqrt{2}\sum_{k=0}^{79}\cos\left(\frac{\pi}{80}(k+0.5)(n+40.5)\right)\hat{S}_{SHB}(k) \qquad \text{(B.7-56)}$$

The super higher band signal is obtained by the following overlap-add operation:

$$\hat{s}_{SHB}^{'}(n) = w_{TDAC}(n+80)\hat{s}_{SHB}^{pre}(n) + w_{TDAC}(n)\hat{s}_{SHB}^{cur}(n) \qquad n = 0,...,79 \qquad \text{(B.7-57)}$$

where $w_{TDAC}(n)$ is the synthesis weighting window:

$$w_{TDAC}(n) = \sin\left(\frac{\pi}{160}(n+0.5)\right) \qquad n = 0,...,159 \qquad \text{(B.7-58)}$$

and $\hat{s}_{SHB}^{pre}(n)$ was obtained from the previous inverse MDCT transform, which is updated as:

$$\hat{s}_{SHB}^{pre}(n) = \hat{s}_{SHB}^{cur}(80+n), \quad n = 0,...,79$$

B.7.6 Time envelope denormalization

Time envelope is applied if the current or previous frame is classified as TRANSIENT. The time envelope $\hat{t}_{rms}^{'}(j)$ and the time envelope adjustment flag F_{tenv} are decoded in clause B.7.3.3.

When the current frame is not TRANSIENT, the time envelope is obtained as follows:

$$\hat{t}_{rms}^{'}(j) = w_{tenv}(j)\hat{t}_{rms}^{'(-1)} \qquad j = 0,\cdots,3 \qquad \text{(B.7-59)}$$

where $w_{tenv} = [0.95, 0.94, 0.93, 0.92]$ and $\hat{t}_{rms}^{'(-1)}$ is the decoded time envelope of the previous sub-frame as described in clause B.7.3.3.

The decoded super higher band signal after applying time envelope is calculated as follows:

$$\hat{s}_{SHB}^{''}(20j+i) = \hat{s}_{SHB}^{'}(20j+i)\cdot\hat{t}_{rms}^{'}(j)\cdot\sqrt{\frac{20}{\left(\sum_{n=20j}^{20(j+1)-1}\hat{s}_{SHB}^{'\,2}(n)\right)+\varepsilon_{rms}}} \qquad \text{(B.7-60)}$$

where $i = 0,\cdots,19$ and $j = 0,\cdots,3$.

When the current frame is TRANSIENT and the time envelope adjustment flag F_{tenv} equals to one, additional time envelope adjustment is applied to $\hat{s}_{SHB}^{''}$. Firstly, the index of the sub-frame with the peak time envelope idx_{trans} is calculated with $idx_{trans} = \arg\max_{j=0,...,3}\left(\hat{t}_{rms}^{'}(j)\right)$.

– If idx_{trans} is not equal to zero (i.e., the peak time envelope is not in the first sub-frame), the first half samples of sub-frame idx_{trans} are attenuated by the ratio between the time envelopes of the previous sub-frame and current sub-frame. Then the decoded super higher band signal is obtained as follows:

$$\hat{s}_{SHB}^{fold}(20idx_{trans}+j) = \hat{s}_{SHB}^{''}(20idx_{trans}+j)\cdot\frac{\hat{t}_{rms}^{'}(idx_{trans}-1)}{\hat{t}_{rms}^{'}(idx_{trans})} \qquad j = 0,\cdots,9 \qquad \text{(B.7-61)}$$

– Otherwise (if idx_{trans} is equal to zero), the first half samples of the first sub-frame are attenuated by the ratio between the average energy of the last ten samples of the previous frame and the time envelope of the current sub-frame, $t'_{rmsq}(0)$. Then the decoded super higher band signal is obtained as follows:

$$\hat{s}^{fold}_{SHB}(j) = \frac{1}{t'_{rmsq}(0)} \sqrt{\frac{1}{10}\sum_{k=1}^{10} \hat{s}'_{SHB}{}^{2}(-k)} \cdot \hat{s}''_{SHB}(j) \qquad j = 0,\cdots,9 \qquad \text{(B.7-62)}$$

The last ten samples of $\hat{s}^{fold}_{SHB}$ are saved as $\hat{s}'_{SHB}(-j), \quad j = 1,...,10$ for the next frame. For the first frame, $\hat{s}'_{SHB}(-j), \quad j = 1,...,10$ is initialized to zero.

B.7.7 Time envelope post-processor

If the current frame and the previous frame are not decoded as TRANSIENT, a time envelope post-processor is applied to smooth the decoded super higher band signal.

$$\hat{s}^{fold}_{SHB}(i) = 0.85\hat{s}'_{SHB}(i) + 0.08\hat{s}'_{SHB}(i-1) + 0.05\hat{s}'_{SHB}(i-2) + 0.02\hat{s}'_{SHB}(i-3) \qquad \text{(B.7-63)}$$

where $i = 0,\cdots,79$.

The last ten samples of $\hat{s}^{fold}_{SHB}$ are saved as $\hat{s}'_{SHB}(-j), \quad j = 1,...,10$ for the next frame.

B.7.8 Frame erasure concealment

B.7.8.1 Frame erasure concealment for the wideband portion of the signal

In case of frame erasures, a FERC algorithm derived from the Appendix IV low complexity PLC algorithm is used to extrapolate missing samples for the wideband part of the signal. The algorithm was adapted to a 5 ms frame length and optimized for performance.

B.7.8.1.1 Enhanced ITU-T G.722 decoder

The enhanced ITU-T G.722 decoder integrating the FERC modules is illustrated in Figure B.7-3. Decoding and frame error concealment is performed in two sub-bands, which are combined using the QMF synthesis filterbank of this Recommendation.

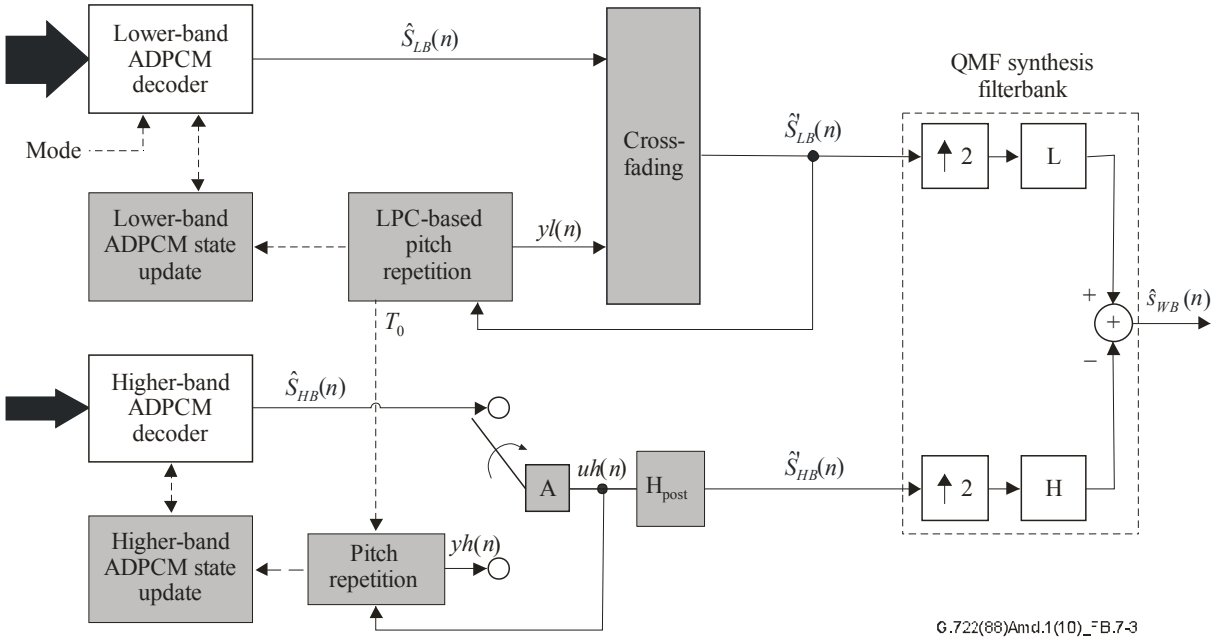


Figure B.7-3 – Block diagram of the ITU-T G.722 decoder with FERC

The ITU-T G.722 decoder with FERC generates an output signal sampled at 16. Its behaviour depends on the type of the current and previous frame (either good or bad frame):

– Without frame erasures (i.e., in the presence of good frames only, see clause B.7.2):

  The bitstream of the lower band (LB) is decoded according to the specified ITU-T G.722 mode (1 or 2, indicating 64 or 56 kbit/s, respectively). The cross-fading block does not change the reconstructed signal, i.e., $\hat{s}'_{LB}(n) = \hat{s}_{LB}(n)$. Similarly, the bitstream of the higher band (HB) is decoded and switch A selects $uh(n) = \hat{s}_{HB}(n)$. Signal *uh(n)* is high-pass filtered by a remove-DC filter H_{post} to obtain $\hat{s}'_{HB}(n)$. The decoded signals $\hat{s}'_{HB}(n)$ and *uh(n)* are stored to be used in case of erasure in future frames.

– In case of frame erasure:

  • In the lower band, for the first erased frame, short- and long-term predictors are updated using the past valid signal $\hat{s}'_{LB}(n)$, $n < 0$. Class information is also extracted. Signal *yl(n)* is generated using these predictors and the class information. The signal for the erased frame is reconstructed as $\hat{s}'_{LB}(n) = yl(n)$, $n = 0,\cdots,39$. In addition, ADPCM states are updated. The process of erased frame reconstruction and ADPCM states update is repeated until a good frame is received. Note that not only the missing frame is generated, but also an additional 10 ms signal, *yl(n)*, $n = 40,\cdots,119$, to be used for cross-fading with the first decoded samples after the erasure.

  • In the higher band, the missing frame is extrapolated using the past signal *uh(n)*, $n < 0$, and ADPCM states are updated. The extrapolated signal *yh(n)* is obtained by repeating pitch-synchronously the previous frame of *uh(n)*. The switch A selects *uh(n) = yh(n)*, $n = 0,\cdots,39$. The signal *uh(n)* is high-pass filtered by a remove-DC filter H_{post} to obtain $\hat{s}'_{HB}(n)$. This process is repeated until a good frame is received.

– In case of good frames following erased frames:

  To insure the continuity of the signal, in the lower band, for the first two good frames (first 10 ms, 80 samples), following erased frames the signal reconstructed by the ADPCM decoder, $\hat{s}_{LB}(n)$ is crossfaded with the signal stored in the cross-fade buffer *yl(n)*, $n = 40,\cdots,119$ to form the lower band output signal $\hat{s}'_{LB}(n)$.

B.7.8.1.2 Functional description of the WB FERC algorithm

B.7.8.1.2.1 Lower band decoding

B.7.8.1.2.1.1 Extrapolation of missing frame: Case of bad frame following a good frame

The extrapolation of a missing frame in the lower band $\hat{s}_{LB}(n)$, $n = 0,\cdots,39$ is illustrated in Figure B.7-4. It comprises, for the first erased frame after a valid frame (N_{erase}=1), the analysis of the past valid signal $\hat{s}'_{LB}(n)$, $n<0$, followed by synthesis of the signal *yl(n)*, $n = 0,\cdots,39$.

The past signal $\hat{s}'_{LB}(n)$, $n = -297,\cdots,-1$ is buffered using a buffer length of 297 samples, which can be divided as follows:

– 288 samples corresponding to twice the maximal pitch delay (2 × 144) used in the PLC algorithm;

– one sample for pitch jitter; and

– eight samples used for LPC memory.

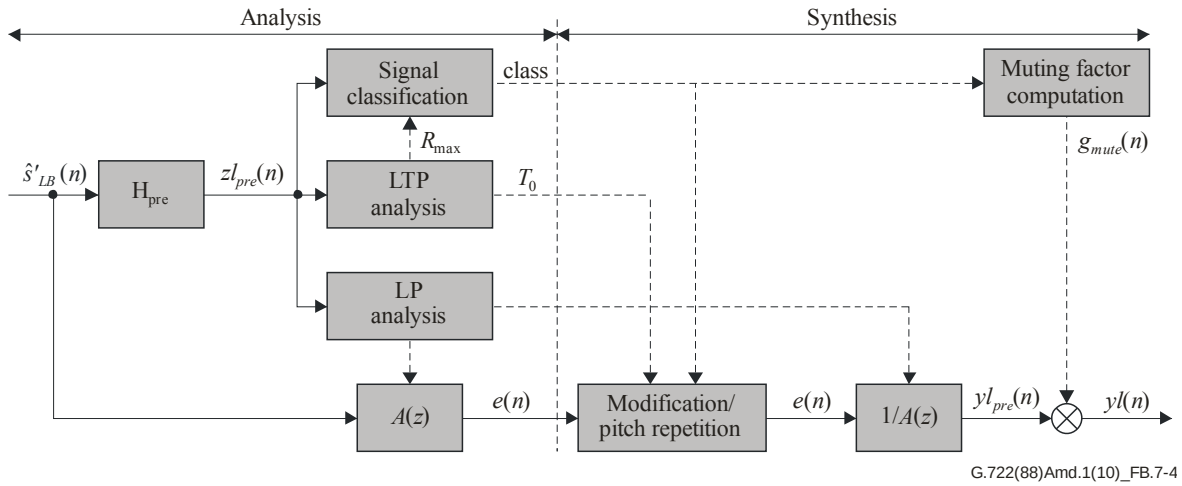


Figure B.7-4 – Block diagram of lower band extrapolation of missing frame

B.7.8.1.2.1.2 Pre-processing

A high-pass filter protects against undesired low-frequency components. A first-order pole/zero filter with a cut-off frequency of 50 Hz is used. This filter is given by:

$$H_{pre}(z) = \frac{1 - z^{-1}}{1 - \frac{123}{128} z^{-1}} \tag{B.7-64}$$

The past signal $\hat{s}'_{LB}(n)$, $n = -297,\cdots,-1$, is filtered through $H_{pre}(z)$ to obtain the pre-processed signal $zl_{pre}(n)$,

$$zl_{pre}(n) = \hat{s}'_{LB}(n) - \hat{s}'_{LB}(n-1) + \frac{123}{128} zl_{pre}(n-1); \quad n = -297,...,-1 \tag{B.7-65}$$

where $\hat{s}'_{LB}(-298)$ and $zl_{pre}(-298)$ are set to 0.

B.7.8.1.2.1.3 LP analysis

The short-term analysis and synthesis filters, $A(z)$ and $1/A(z)$, are based on eighth-order linear prediction (LP) filters. The LP analysis filter is defined as:

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_8 z^{-8} \tag{B.7-66}$$

The LP analysis is made on the past valid pre-processed signal $zl_{pre}(n)$, $n = -80,\cdots,-1$. It consists of windowing, autocorrelation computation and the Levinson-Durbin algorithm. The LP window here is an asymmetrical Hamming window defined as:

$$w_{lp}(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{(n+80)\pi}{69}\right), & n = -80,\cdots,-11 \\ 0.54 + 0.46\cos\left(\frac{(n+11)\pi}{10}\right), & n = -10,\cdots,-1 \end{cases} \tag{B.7-67}$$

This window $w_{lp}(n)$, which is limited to 80 samples (10 ms at 8-kHz sampling frequency) to reduce complexity, is applied to the last 10 ms of $zl_{pre}(n)$, $n = -80,\cdots,-1$:

$$zl'_{pre}(n) = w_{lp}(n)\, zl_{pre}(n) \quad n = -80,...,-1 \tag{B.7-68}$$

The windowed speech $zl'_{pre}(n)$ is used to compute the autocorrelation coefficients:

$$r(k) = \sum_{n=k}^{80} zl'_{pre}(n)\, zl'_{pre}(n-k) \quad k = 0,...,8 \tag{B.7-69}$$

To avoid arithmetic problems for low-level input signals the value of $r(0)$ has a lower boundary of $r(0) = 1.0$. A 60 Hz bandwidth expansion is applied by multiplying the autocorrelation coefficients with:

$$w_{lag}(k) = \exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 k}{f_s}\right)^2\right] \quad k = 1,...,8 \tag{B.7-70}$$

where $f_0 = 60$ Hz is the bandwidth expansion and $f_s = 8\,000$ Hz is the sampling frequency. Furthermore, $r(0)$ is multiplied by a white-noise correction factor 1.0001, which is equivalent to adding a noise floor at –40 dB. The modified autocorrelation coefficients are given by:

$$\begin{aligned} r'(0) &= 1.0001\, r(0) \\ r'(k) &= w_{lag}(k)\, r(k) \quad k = 1,...,8 \end{aligned} \tag{B.7-71}$$

The Levinson-Durbin algorithm is identical to that described in clause B.6.4.2.1.

After the LP analysis, the past signal $\hat{s}'_{LB}(n)$, $n = -289,\cdots,-1$, is filtered through $A(z)$ to obtain the residual signal $e(n)$

$$e(n) = s'_{LB}(n) + \sum_{i=1}^{8} a_i\, \hat{s}'_{LB}(n-i);\ n = -289,...,-1 \tag{B.7-72}$$

B.7.8.1.2.1.4 LTP analysis

The PLC algorithm uses pitch period repetition. The pitch period or pitch delay, T_0, is determined on the past valid pre-processed signal just before erasure, $zl_{pre}(n)$, $n = -288,\cdots,-1$. T_0 is estimated in open loop by a long-term predictive (LTP) analysis.

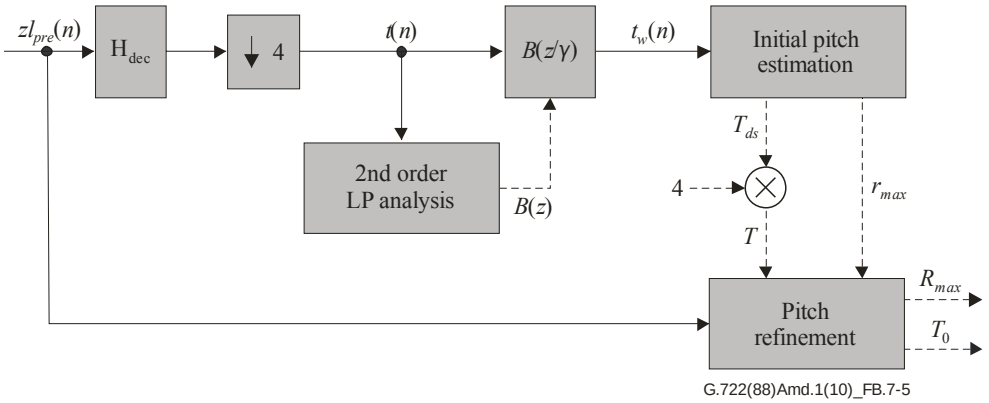


Figure B.7-5 – Block diagram of LTP analysis

As illustrated in Figure B.7-5, pitch estimation is conducted in the following steps:

– The signal $zl_{pre}(n)$, $n = -288,\cdots,-1$, is low-pass filtered by $H_{dec}(z)$, where:

$$H_{dec}(z) = \frac{3692(1+z^{-8}) + 6190(z^{-1}+z^{-7}) + 8525(z^{-2}+z^{-6}) + 10186(z^{-3}+z^{-5}) + 10787z^{-4}}{65536} \tag{B.7-73}$$

is an eighth-order FIR filter, and decimated by a factor of four to obtain the signal $t(n)$, $n = -72, ..., -1$, sampled at 2 kHz. The filter memory of length 8 is initialized to 0 at each first erased frame.

– The signal $t(n)$, $n = -72,\cdots,-1$, is weighted by a filter $B(z/\gamma_{LTP\_FEC})$, where $B(z) = 1 - b_1 z^{-1} - b_2 z^{-2}$ and $\gamma_{LTP\_FEC} = 0.94$, to obtain the signal $t_w(n)$, $n = -70,\ldots,-1$. The coefficients of $B(z)$ are obtained by 2nd-order LP analysis of $t(n)$ using the windowing, autocorrelation computation and Levinson-Durbin algorithm described in the previous clause. Note that only the last 72 samples of the window $w_{lp}(n)$, $n = -72,\cdots,-1$, are used, which gives a 36 ms time support at 2 kHz sampling frequency.

– A first estimation T_{ds} of the pitch delay is computed in the weighted decimated signal domain by normalized cross-correlation as follows:

a) Initialization: $T_{ds} = 18$.

b) Computation of the normalized cross-correlation, *r(i)*:

$$r(i) = \frac{\sum_{j=-35}^{-1} t_w(j) t_w(j-i)}{\max\left(\sum_{j=-35}^{-1} t_w^2(j), \sum_{j=-35}^{-1} t_w^2(j-i) \right)}, \; i = 1,...,35 \tag{B.7-74}$$

c) Computation of zero crossings, *zcr(i)*, in the last i samples, for $i = 2,\cdots,35$

$$zcr(i) = \sum_{n=-i}^{-1} \left[\left((t_w(n) \geq 0) \otimes (t_w(n+1) < 0) \right) \oplus \left((t_w(n+1) \geq 0) \otimes (t_w(n) < 0) \right) \right], \quad i=2,...35 \tag{B.7-75}$$

where the comparisons ≥ and < give a binary result (1 for true, 0 for false).

d) Determination of the first delay i_0 in [1,...,35] for which $r(i) < 0$ and *zcr(i)* > 0. Note that if i_0 is not found in [1,...,35], steps e) and f) are omitted and the initial value $T_{ds} = 18$ is kept.

e) Determination of the lower bound for the maximum correlation search:

$$i_1 = \max(i_0, 4) \tag{B.7-76}$$

f) Search for the maximum correlation and its index T_{ds} in $[i_1, 35]$ by a procedure favouring smaller pitch values, to avoid choosing pitch multiples. A second best candidate T_{ds2} is also memorized with the following constraint: $T_{ds} - T_{ds2} > 1$.

– The pitch delay T_{ds} estimated in the 2 kHz sub-sampled signal domain is then refined in the 8 kHz sampled pre-processed signal domain. The pitch delay T_0 is searched for by maximizing a normalized cross-correlation function *R(i)* in the interval $[T-2,..., T+2]$ where $T = 4T_{ds}$:

$$T_0 = \arg \max_{i=T-2,\cdots,T+2} R(i) \tag{B.7-77}$$

with:

$$R(i) = \frac{\sum_{j=-T}^{-1} zl_{pre}(j)\, zl_{pre}(j-i)}{\max\left(\sum_{j=-T}^{-1} zl_{pre}^2(j), \sum_{j=-T}^{-1} zl_{pre}^2(j-i) \right)} \tag{B.7-78}$$

– The value R_{max} is computed as $R_{max} = R(T_0)$.

– If $R_{max} > 0.4$, another pitch delay $T_0^{'}$ is searched for by maximizing a normalized cross-correlation function *R′(i)* in the interval $[T_2 - 2,..., T_2 + 2]$ where $T_2 = 4T_{ds2}$:

$$T_0^{'} = \arg \max_{i=T_2-2,\cdots,T_2+2} R'(i) \tag{B.7-79}$$

with

$$R'(i) = \frac{\sum_{j=-T_2}^{-1} zl_{pre}(j) zl_{pre}(j-i)}{\max\left(\sum_{j=-T_2}^{-1} zl_{pre}^2(j), \sum_{j=-T_2}^{-1} zl_{pre}^2(j-i) \right)}, \quad i = T_2 - 2,..., T_2 + 2 \tag{B.7-80}$$

If $R'\left(T_0^{'}\right) > R\left(T_0\right)$ then $T_0^{'}$ is considered as the pitch value, the value of T_0 is updated as $T_0 = T_0^{'}$ and the value of R_{max} is updated as $R_{max} = R'(T_0^{'})$.

– Otherwise, if the value $R_{max} < 0.25$ and $T_0 < 32$ then the pitch value is doubled to avoid high frequency resonance due to too short period repetitions: $T_0 = 2*T_0$.

B.7.8.1.2.1.5 Signal classification

The PLC strategy uses signal classification based on signal characteristics to optimize quality. For instance, if the frame preceding an erasure is a non-stationary segment (e.g., plosives), the signal should be rapidly muted; if this frame is a stationary segment (e.g., strongly voiced speech), it can be pitch-synchronously repeated and slowly damped. Classification is used in the PLC algorithm for LP residual extrapolation and muting control.

The signal $zl_{pre}(n)$, $n = –288,\cdots,–1$ preceding an erasure is classified into one out of five possible classes, which are defined as follows:

– TRANSIENT (TR) for transients with large energy variation (e.g., plosives);

– UNVOICED (UV) for unvoiced signals;

– VUV_TRANSITION (VUV) for a transition from voiced to unvoiced signals;

– WEAKLY_VOICED (WV) for weakly voiced signals (e.g., onset or offset of vowels);

– VOICED (V) for voiced signals (e.g., steady vowels).

The features used for classification are the following:

– the maximum normalized correlation R_{max}, which is a side product of the LTP analysis;

– the higher and lower bands energy ratio, which is obtained here in the log domain by taking the difference between the lower- and higher band ADPCM delayed logarithmic quantizer scale factors, NBH – NBL using the ITU-T G.722 notations. NBL and NBH are computed as in clause 3.5 of ITU-T G.722;

– the zero-crossing rate *zcr* of $zl_{pre}(n)$, $n = –80,\cdots,–1$, defined as:

$$zcr = \sum_{n=-80}^{-1} \left[(zl_{pre}(n) \le 0) \otimes (zl_{pre}(n-1) > 0) \right] \tag{B.7-81}$$

where the comparisons ≤ and > give a binary result (1 for true, 0 for false);

– the number *cnt_peak* of detected high amplitude peaks in the LP residual in the last pitch period with respect to the last but one period: This value is computed only when *class* is WEAKLY_VOICED or VUV_TRANSITION.

$$cnt\_peak = \sum_{n=-T_0}^{-1} \left[\frac{|e(n)|}{8} > \max_{i=-2,\ldots,2} \left(|e(n-T_0+i)| \right) \right] \quad \text{(B.7-82)}$$

where the comparison > gives a binary result (1 for true, 0 for false) and T_0 is the pitch delay estimated by the LTP analysis. The counter, *cnt_peak*, represents the number of detected large peaks in the last pitch period that were not present in the previous pitch period.

Based on these features, the signal category, *class*, is obtained by heuristics according to the flowchart shown in Figure B.7-6. Note also that if *class* is not VOICED, and T_0 is even, T_0 is increased by 1. The so-called pitch delay T_0 determines the repetition period used in the residual signal generation procedure.

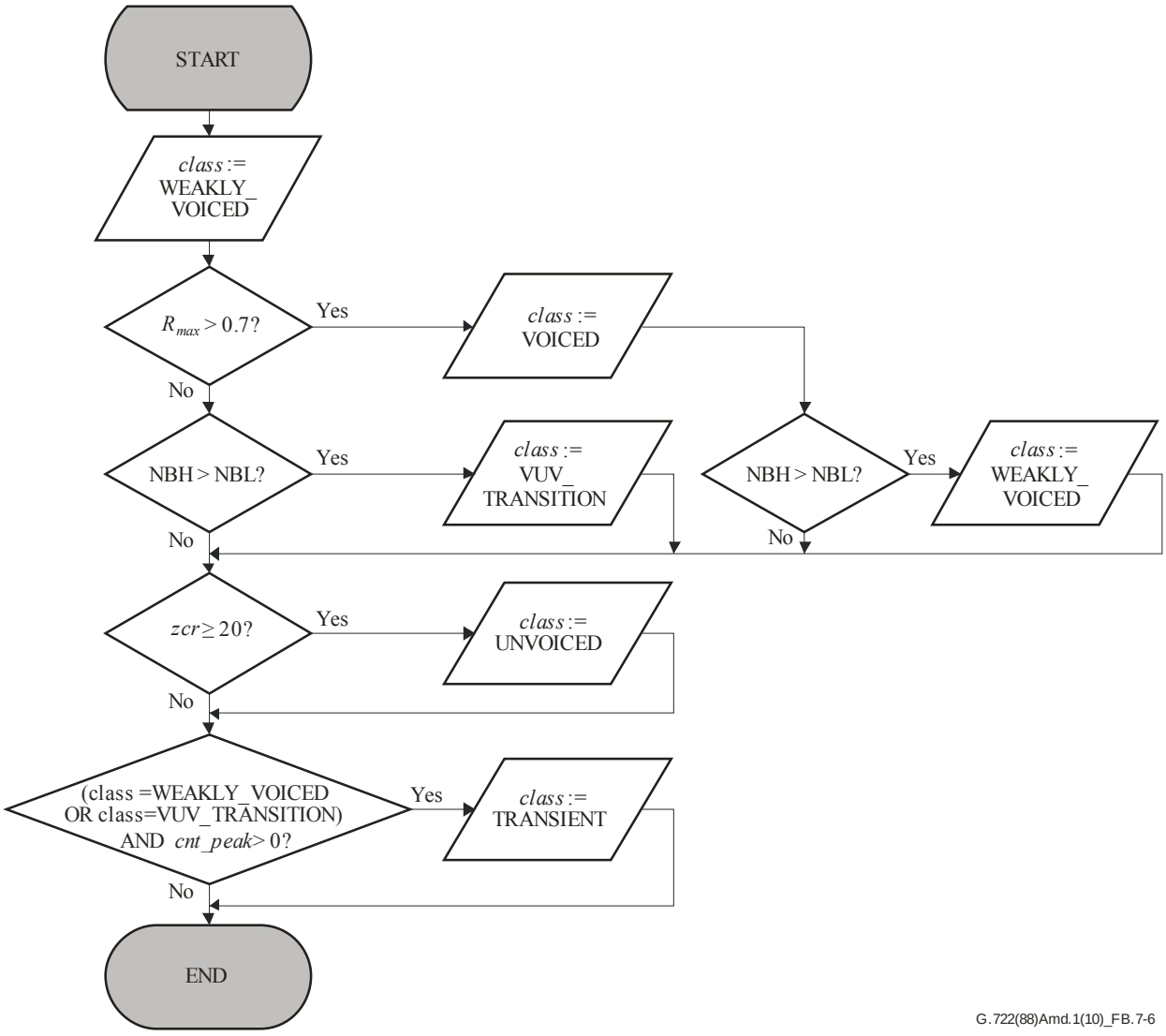


Figure B.7-6 – Classification flowchart

B.7.8.1.2.1.6 Modifications of the repetition period

Based on the classification results, the repetition period T_0 value can be modified in the following cases:

– If *class* is set to UNVOICED and $T_0 < 32$ the pitch delay T_0 value is doubled: $T_0 = 2*T_0$ to avoid artefacts due to too short period repetition.

– If *class* is set to TRANSIENT, T_0 is upper bounded by 40 (5 ms): $T_0 = \min(40, T_0)$.

– If *class* is not VOICED, and T_0 is even, T_0 is increased by 1.

– If *class* is set to VOICED it is verified that there are not two glottal pulses in the last period due to decreasing pitch. The procedure uses the following values:

$$m_T = \frac{\sum_{i=1}^{T_0} |e(-i)|}{T_0} \tag{B.7-83}$$

$$mx_T = \max(|e(-i)|), i = 1,...,T_0 \tag{B.7-84}$$

$$I_{mx} = \arg \max_{i=1,\cdots,T_0} |e(-i)| \tag{B.7-85}$$

When $mx_T > 4m_T$, a second maximum is searched in the following intervals:

$$mx_{T2} = \max(|e(-i)|), i = [I_{mx} + T_0 - 5, T_0], [1, I_{mx} - T_0 + 5] \tag{B.7-86}$$

and I_{mx2} is the index of this second maximum. Note that when the first maximum is not close to an extremity of the repetition period, these intervals are empty and mx_{T2} is undefined. If mx_{T2} is defined and $mx_{T2} > mx_T/2$ and the signs of these two pulses are identical, the pitch value is set to

$$T_0 = |I_{mx2} - I_{mx}| \tag{B.7-87}$$

to improve the correspondence of the repetition period with the pitch period for voiced signals.

B.7.8.1.2.1.7 Modification/pitch repetition of LP residual

Before performing the pitch repetition procedure, the LP residual that forms the repetition period is modified as follows:

– if *class* is WEAKLY_VOICED or VUV_TRANSITION, the repetition period is corrected to limit the amplitude of an eventual transition signal. The modification consists in limiting the magnitude of each sample in the repetition period in function of the previous period as follows:

$$e(n) = \min\left(\max_{i=-2,\cdots,+2} \left(|e(n - T_0 + i)| \right), |e(n)| \right) \times sgn(e(n)), \quad n = -T_0, \cdots, -1 \tag{B.7-88}$$

– if *class* is UNVOICED, the last T_0 samples are smoothed as follows:

$$e(n) = e(n)/4, \quad \text{if } |e(n)| > m_{th}, \quad n = -T_0, ..., -1$$
$$\text{where } m_{th} = \frac{\sum_{i=1}^{80} |e(-i)|}{32}. \tag{B.7-89}$$

B.7.8.1.2.1.8 Pitch repetition of LP residual

The LP excitation signal $e(n)$, $n = 0,\cdots,39$, in the missing frame is extrapolated based on the classification. In addition, 80 extra samples (10 ms), $e(n)$, $n = 40,\cdots,119$, are generated for the purpose of cross-fading.

– If *class* is VOICED, the missing excitation signal, $e(n)$, $n = 0,\cdots,39$, and the excitation signal for cross-fading $e(n)$, $n = 40,\cdots,119$, are generated by repeating pitch-synchronously the repetition period:

$$e(n) = e(n - T_0) \tag{B.7-90}$$

– If *class* is not VOICED, the pitch-synchronous repetition procedure is modified to avoid over-voicing by introducing, sample by sample, a small jitter using the following procedure. The samples of the repetition period can be viewed as grouped two by two; then, every two samples forming a group are swapped and the swapped groups are concatenated to form the extrapolated residual signal. With this procedure, the missing excitation signal, $e(n)$, n= 0,⋯,39 and the excitation signal for cross-fading $e(n)$, n= 40,⋯,119, are obtained as:

$$e(n) = e(n - T_0 + (-1)^n) \tag{B.7-91}$$

B.7.8.1.2.1.9 LP synthesis

The extrapolated excitation signal $e(n)$, n = 0,⋯,39, is filtered with the LP synthesis filter $1/A(z)$ to obtain the reconstructed missing frame, $yl_{pre}(n)$:

$$yl_{pre}(n) = e(n) - \sum_{i=1}^{8} a_i \, yl_{pre}(n-i) \tag{B.7-92}$$

Then, 80 samples (10 ms), $yl_{pre}(n)$, n = 40,⋯,119 are generated by LP synthesis filtering of the excitation signal for cross-fading $e(n)$, n = 40,⋯,119 with an attenuated LP synthesis filter, $A(z/\gamma_{LPC\_FEC})$, with γ_{LPC_FEC} = 0.99:

$$yl_{pre}(n) = e(n) - \sum_{i=1}^{8} a_i \gamma_{LPC\_FEC}{}^{i} \, yl_{pre}(n-i), \quad \text{n=0,...,119} \tag{B.7-93}$$

B.7.8.1.2.1.10 Adaptive muting

The energy of the reconstructed signal is controlled by applying to each sample a gain factor that is computed and adapted sample by sample. Thus, the synthesized signal $yl_{pre}(n)$, n = 0,⋯,119 (40 samples for the current lost frame and 80 samples needed for the cross-fading), is muted sample by sample with a muting factor function *g_mute(n)* for n = 0,⋯,119 to obtain the reconstructed lower band signal $yl(n)$ before crossfading:

$$yl(n) = g\_mute(n) \cdot yl_{pre}(n) \tag{B.7-94}$$

The value of *g_mute(n)* depends on the value of *class*. The muting factor adaptation is illustrated in Figure B.7-7 for the three different cases. In case of consecutive erased frames, the index of the muting factor n increases continuously and so the muting is done according to the following equation:

$$yl(n) = g\_mute(40(N_{erase} - 1) + n) \cdot yl_{pre}(n) \tag{B.7-95}$$

where N_{erase} is the number of erased frames. N_{erase} is set to one at the first erased frame and incremented by one at each consecutive erased frame.

As can be observed in Figure B.7-7, the complete muting is achieved in 10, 30 or 60 ms respectively for the three different classes (TRANSIENT, VUV_TRANSITION and other classes).

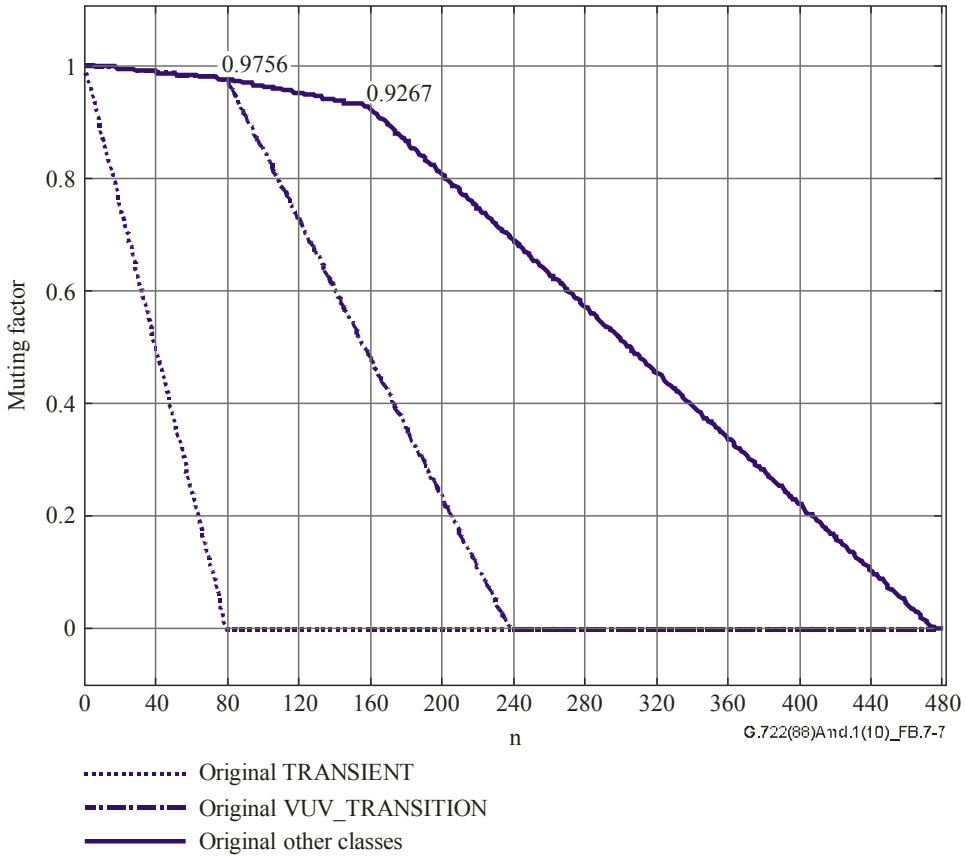


Figure B.7-7 – Muting factor as a function of the sample index and the class value

Initialized to one (32767 in Q15), the muting factor is decreased sample by sample in function of the class and index range of *n* according to Table B.7-1.

Table B.7-1 – Sample by sample muting factor decrease values (in Q15)

	Index range of *n*			
Class	**0-79**	**80-159**	**160-239**	**240-479**
TRANSIENT	409	0	0	0
VUV_TRANSITION	10	200	200	0
Other classes	10	20	95	95

B.7.8.1.2.1.11 Extrapolation of missing frame: Case of a bad frame following a bad frame

In the case of a bad frame following a bad frame, the analysis parameters computed for the first erased frame (a_i, i = 1,⋯,8, T_0, *class*) are kept. The first 40 samples of the signal generated in the previous frame for cross-fading $yl(n)$, n = 40,⋯,79 are copied to $yl(n)$, n = 0,⋯,39, and the other 40 cross-fading samples $yl(n)$, n = 80,⋯,119 are also shifted to $yl(n)$, n = 40,⋯,79. The last 40 samples for the cross-fading with the next frame $yl(n)$, n = 80,⋯,119 are synthesized as described above:

– The LP excitation signal $e(n)$ n = 80,⋯,119 is generated by the pitch-synchronous repetition procedure (see clause B.7.8.1.2.1.8).

– The signal $yl_{pre}(n)$, n = 80,⋯,119 is obtained by LP synthesis filtering of the excitation signal $e(n)$, n = 80,⋯,119 using the attenuated LP synthesis filter $A(z/\gamma_{LPC\_FEC})$ (see clause B.7.8.1.2.1.9).

– The signal $yl_{pre}(n)$, n = 80,⋯,119 is attenuated according to Equation (B.7-95) in clause B.7.8.1.2.1.10 to obtain the signal $yl(n)$, n = 80,⋯,119.

B.7.8.1.2.1.12 Update of ADPCM decoder states

The states of the lower band ADPCM decoder are updated after extrapolating missing frames to help in recovery from frame erasures. This update is more elaborate than a simple ADPCM decoder reset. However, to minimize complexity, the ADPCM states are updated based on available or *a priori* information, without additional processing. The decoder states are modified as follows, using ITU-T G.722 notation:

The quantized delayed difference signal for the adaptive predictor is updated as:

$$\mathrm{DLT}^i = 0,\ i = 1,\cdots,6 \tag{B.7-96}$$

The partially reconstructed signal with delays 1 and 2 predictor is updated as:

$$\mathrm{PLT}^i = \frac{yl(40-i)}{2},\ i = 1,2 \tag{B.7-97}$$

The reconstructed signal for the adaptive predictor with delay 1 predictor is updated as:

$$\mathrm{RLT}^1 = yl(39) \tag{B.7-98}$$

The predictor output value predictor is updated as:

$$\mathrm{SL} = yl(40) \tag{B.7-99}$$

The zero section output signal predictor is updated as:

$$\mathrm{SZL} = \frac{yl(40)}{2} \tag{B.7-100}$$

If more than four frames were erased, the delayed quantizer scale factor predictor DETL and the delayed logarithmic quantizer scale factor predictor NBL are updated as:

$$\mathrm{DETL} = 32 \tag{B.7-101}$$

$$\mathrm{NBL} = 0 \tag{B.7-102}$$

B.7.8.1.2.1.13 Cross-fading

The cross-fading is detailed in Table B.7-2. The cross-fading window with a time length of 10 ms is a concatenation of a flat part of 20 samples and a Bartlett window (triangular) part of 60 samples. When the current frame is erased ("Bad"), the output is always equal to the concealment module output $\hat{s}'_{LB}(n) = yl(n)$, $n = 0,\cdots,39$.

When the current frame is received ("Good"), the cross-fading depends on the status of the two previous frames. Table B.7-2 summarizes the possible cases. In this table *yl(n), n*=40,⋯,119, refers to the cross-fade buffer that contains 80 samples and it is not shifted any more in case of received frames.

Table B.7-2 – Cross-fading operation *e*

Frame N-2	Frame N-1	Current frame N received
Bad or Good	Bad	$\hat{s}'_{LB}(n) = yl(n+40) \quad n = 0,...,19$ $\hat{s}'_{LB}(n) = \frac{546(n-19)}{32767}\hat{s}_{LB}(n) + (1-\frac{546(n-19)}{32767})yl(n+40) \quad n = 20,...,39$
Bad	Good	$\hat{s}'_{LB}(n) = \frac{546(n+21)}{32767}\hat{s}_{LB}(n) + (1-\frac{546(n+21)}{32767})yl(n+80) \quad n = 0,...,39$
Good	Good	$\hat{s}'_{LB}(n) = \hat{s}_{LB}(n) \quad n = 0,...,39$

B.7.8.1.2.2 Higher band decoding

B.7.8.1.2.2.1 Extrapolation of missing frame

The extrapolation of a missing frame in the higher band uses the past signal $uh(n)$, $n = -160,\cdots,-1$, to form the repetition period.

B.7.8.1.2.2.2 Modification of the past high-band signal

Before performing the pitch repetition procedure, the repetition period is modified if *class* is UNVOICED. The last 80 samples are smoothed in the following way:

$$uh(n) = uh(n)/4, \quad \text{if } |uh(n)| > m_{th}, \quad n\text{=-80,...,-1}$$
$$\text{where } m_{th} = \frac{\sum_{i=1}^{80}|uh(-i)|}{32}. \tag{B.7-103}$$

B.7.8.1.2.2.3 Pitch repetition of the past high-band signal

The extrapolation of a missing frame in the higher band consists of pitch synchronous repeating of the previous signal $uh(n)$ if *class* = VOICED; otherwise, the repetition period is set to 80 samples (10 ms):

$$yh_{pre}(n) = zh(n - T_h), n = 0,..., L-1 \tag{B.7-104}$$

where $T_h = T_0$ if *class* = VOICED, $T_h = 80$ otherwise.

B.7.8.1.2.2.4 Adaptive muting

As for the lower band reconstructed signal, the energy of the higher band reconstructed signal is also controlled by applying a gain factor computed and adapted sample by sample. To obtain the reconstructed higher band signal $yh(n)$, the synthesized signal $yh_{pre}(n)$ for $n = 0,\cdots,39$ is muted sample by sample with the same adaptive muting factor function used for muting the corresponding lower band sample, as described in clause B.7.8.1.2.1.10:

$$yh(n) = g\_mute\left(40\left(N_{erase} - 1\right) + n\right) \cdot yh_{pre}(n) \tag{B.7-105}$$

B.7.8.1.2.2.5 Update of ADPCM decoder states

Similar to lower band decoding, the states of the higher band ADPCM decoder are updated after extrapolating a missing frame. The update is described below using ITU-T G.722 notation:

The delayed logarithmic quantizer scale factor NBH and the delayed quantizer scale factor DETH are updated as:

$$\text{NBH} = \text{NBH}/2 \tag{B.7-106}$$

$$\text{DETH} = \text{SCALEH(NBH)} \tag{B.7-107}$$

If more than four frames were erased, the delayed logarithmic quantizer scale factor NBH and the delayed quantizer scale factor NETH are updated as:

$$\text{NBH} = 0 \tag{B.7-108}$$

$$\text{NETH} = 8 \tag{B.7-109}$$

The update is restricted to the higher band scale factor.

B.7.8.2 Frame erasure concealment in the super higher band

In the case of an erased frame, the super higher band FERC algorithm is used to recover the super higher band signal. Except for the operational coder mode R1sm, for the frame following the erased one, on the condition that the signal class is detected as NORMAL or NOISE, that frame has to be treated as an erased frame. Note that the class information of two sequential non-erased frames are required in order to extract the proper SHB mode for this condition. The FERC flag of the current frame, f_{FERC}^{cur}, is calculated as,

$$f_{FERC}^{cur} = \begin{cases} 1 & \text{if } f_{FERC}^{det} = 1 \text{ or} \\ & \left(f_{FERC}^{(-1)} = 1 \text{ and } F_{class} = \left(\text{NORMAL or NOISE}\right) \text{ and not for R1sm}\right) \\ 0 & \text{otherwise} \end{cases} \tag{B.7-110}$$

where f_{FERC}^{det} represents the detected frame erasure, $f_{FERC}^{(-1)}$ is the copied flag of f_{FERC}^{det} in the previous frame. When f_{FERC}^{cur} equals to one, buffers $\hat{S}_{SHB}^{adp(m-1)}(k)$, $\hat{S}_{SHB}^{AVQ(m)}(k)$, $\hat{S}_{SHB}^{AVQ(m-1)}(k)$ and $\hat{S}_{SHB}^{AVQ(m-2)}(k)$, for BWE/AVQ adaptation described in clause B.7.4.7, are reset to zero, and F_{class} is set to NORMAL. If the f_{FERC}^{det} is zero, that means, the signal class of the current frame is correctly obtained. The signal class of the previous frame, $F_{class}^{(-1)}$, should be updated with the correct current one for the preparation of the SHB mode extraction in the next frame. Since no SHB mode is required for R1sm, f_{FERC}^{cur} is always identical to f_{FERC}^{det}.

Then, for the erased frame ($f_{FERC}^{cur} = 1$), an attenuated inverse MDCT signal from the last Good frame is copied and used as the inverse MDCT data for erased frames. Overlap and add (OLA) is performed to generate the recovered super higher band signal.

The intermediate inverse-transformed super higher band signal, $\hat{s}_{SHB}^{cur}(n)$ is recovered using an attenuated version of the previous signal $\hat{s}_{SHB}^{pre}(n)$ by:

$$\hat{s}_{SHB}^{cur}(n) = \alpha_{HBPLC2}\hat{s}_{SHB}^{pre}(n) \qquad n = 0,\ldots,79 \tag{B.7-111}$$

where $\alpha_{HBPLC2} = 0.875$ is an attenuation factor. Then, to synthesize output signal $\hat{s}_{SHB}^{'}(n)$, an OLA is performed using Equation (B.7-57). Finally, a time envelope is applied as described in clauses B.7.6 and B.7.7.

B.7.9 Bandwidth switching

Since the scalable bitstream structure allows bitrate switching, this may result in bandwidth switching, where there is a change in frequency bandwidth between adjacent frames (e.g., a wideband frame followed by a superwideband one). The process described in this clause mitigates the audible artefact by smoothing out the spectral coefficients across frames with different bandwidth. Note that this smoothing takes place for some succeeding frames.

Let the decoded coder mode of this frame for the current *m*-th frame be $F_{bw}^{(m)}$. $F_{bw0}^{(m-1)}$ and $F_{bw1}^{(m-1)}$ are the decoded coder mode of the previous frame and the bitstream coder mode of the previous frame, respectively. The initial value of $F_{bw0}^{(-1)}$ and $F_{bw1}^{(-1)}$ is –1. In case the first frame is superwideband, $F_{bw1}^{(m-1)}$ is set to the decoded coder mode of current frame $F_{bw}^{(m)}$ to avoid switching from wideband to superwideband.

A flag $F_{bws}^{(m)}$ is defined to indicate the bandwidth switching mode and calculated as follows:

– If $F_{bw}^{(m)} < MODE\_R1sm$ and $F_{bw0}^{(m-1)} \geq MODE\_R1sm$, $F_{bws}^{(m)}$ is set to one and $F_{bw1}^{(m-1)}$ is set to $F_{bw}^{(m)}$. It indicates that there has been a switch from superwideband to wideband;

– If $F_{bw}^{(m)} \geq MODE\_R1sm$ and $F_{bw1}^{(m-1)} < MODE\_R1sm$, $F_{bws}^{(m)}$ is set to two. It indicates that there has been a switch from wideband to superwideband;

– For other cases, $F_{bws}^{(m)}$ is set to zero. There is no bandwidth switching.

$F_{bws}^{(m-1)}$ is the bandwidth switching mode of previous frame. It is initialized to zero.

B.7.9.1 Superwideband to wideband switching

In case $F_{bw}^{(m)}$ = 1, the decoder is forced to operate in the R1sm mode to reduce the effect of a sudden loss of 7-14 kHz frequency components. The current coder mode $F_{bw}^{(m)}$ is set to MODE_R1sm. The MDCT coefficients in the missing frequency range are obtained by BWE (see clause B.7.3), as described below.

Firstly, the spectral envelope of the excitation signal $f_{env}^{exc}(j), j = 0,...,7$ is calculated

$$f_{env}^{exc}(j) = \begin{cases} \sqrt{\frac{1}{8}\sum_{k=b_{wb}(0)}^{b_{wb}(1)-1} \hat{S}_{exc\_base}^{(m)\,2}(k)} & j = 0 \\ \sqrt{\frac{1}{N_{wbcf}(j)}\sum_{k=b_{wb}(j)}^{b_{wb}(j+1)-1} \hat{S}_{exc\_base}^{(m)\,2}(k)} & j = 1,...,7 \end{cases} \quad \text{(B.7-112)}$$

where $b_{wb}(j)$ and $N_{wbcf}(j)$ are the sub-band boundaries and the number of coefficients per sub-band as defined in Table B.7-3. The signal class of the super higher band is set to NORMAL. $\hat{S}_{exc\_base}^{(m)}(k)$ is obtained as described in clause B.7.3.5.

Table B.7-3 – Sub-band boundaries and number of coefficients per sub-band in wideband for bit-rate switching

j	$b_{wb}(j)$	$N_{wbcf}(j)$
0	20	4
1	24	8
2	32	8
3	40	8
4	48	8
5	56	8
6	64	8
7	72	8
8	80	–

The maximum envelope value is obtained with $f_{env\_\max}^{exc} = \max_{j=0,\ldots,7}(f_{env}^{exc}(j))$. The RMS value of the last half spectrum of the excitation signal E_{exc} is calculated as follows:

$$E_{exc} = \sqrt{\frac{1}{40}\sum_{k=40}^{79}(\hat{S}_{exc\_base}^{(m)}(k))^2} \qquad \text{(B.7-113)}$$

The predicted spectral envelope of the super higher band signal is obtained as follows:

$$\hat{f}_{env}^{'}(j) = \frac{E_{exc}}{8}\cdot\frac{f_{env}^{exc}(j)}{f_{env\_\max}^{exc}} \qquad j = 0,\cdots,7 \qquad \text{(B.7-114)}$$

The predicted spectral envelope is smoothed according to the spectral envelope of the previous frame when bandwidth switching from superwideband to wideband.

An energy ratio between the same parts of the wideband spectrum of the current frame and the previous frame is introduced to control the smoothing of the predicted spectral envelope. This ratio r_{bws} reflects the correlation of the spectral envelope of the current frame and the previous frame:

$$r_{bws} = \frac{E_{exc\_lb}^{(m-1)}}{E_{exc\_lb}^{(m)}} \qquad \text{(B.7-115)}$$

where $E_{exc\_lb}^{(m)}$ is the RMS value of the first 45 spectral coefficients of the current excitation signal, calculated as:

$$E_{exc\_lb}^{(m)} = \sqrt{\frac{1}{45}\sum_{k=0}^{44}\hat{S}_{exc\_base}^{(m)\,2}(k)} \qquad \text{(B.7-116)}$$

and $E_{exc\_lb}^{(m-1)}$ is the one for the previous frame. $E_{exc}^{lb(m-1)}$ is initialized to zero.

Factor w_{bws2} is the weighting factor for the spectral envelope of the current frame, and $(1-w_{bws2})$ is the one for the previous frame. w_{bws2} is initialized to 0.1. Note that w_{bws2} is reset to 0.1 if $F_{bws}^{(m-1)}$ is equal to zero.

– If w_{bws2} is less than 0.5, the predicted spectral envelope is differently weighted according to the energy ratio r_{bws}. Then, the decoded spectral envelope is obtained as follows:

$$\hat{f}_{env}(j) = \begin{cases} w_{bws} \cdot \hat{f}'_{env}(j) + (1 - w_{bws}) \cdot \hat{f}^{(m-1)}_{env}(j), & \text{if} \quad 0.5 < r_{bws} < 2.0 \\ \frac{1}{2}\left(\hat{f}'_{env}(j) + \hat{f}^{(m-1)}_{env}(j)\right), & \text{otherwise} \end{cases} \qquad j = 0, \cdots, 7 \qquad \text{(B.7-117)}$$

where $\hat{f}^{(m-1)}_{rms}(j)$ is the spectral envelope of the previous frame. Factor w_{bws2} is incremented by 0.01 and saved for the next frame.

– Otherwise, the predicted spectral envelope is weighted as follows:

$$\hat{f}_{env}(j) = 0.5 \cdot (\hat{f}'_{env}(j) + \hat{f}^{(m-1)}_{env}(j)) \qquad j = 0, \cdots, 7 \qquad \text{(B.7-118)}$$

Then, the decoded super higher band MDCT coefficients $\hat{S}_{SHB}(k)$, $k = 0, \cdots, 59$, are obtained using Equation (B.7-33) found in clause B.7.3.6.

Counter c_{bws} denotes the number of consecutive wideband frames after superwideband frames. The initial value of c_{bws} is zero. If $F^{(m)}_{bws} = 1$, c_{bws} is incremented by one. Otherwise, c_{bws} is reset to zero.

The obtained super higher band frequency coefficients are attenuated by a factor w_{bws3}.

$$\hat{S}_{SHB}(k) = w_{bws3} \cdot \hat{S}_{SHB}(k) \qquad k = 0, \cdots, 59 \qquad \text{(B.7-119)}$$

Factor w_{bws3} is initialized to one and is updated as follows:

– In case $F^{(m)}_{bws} = 1$,

a) If c_{bw} is larger than 200, w_{bws3} is decremented by 0.01;

b) If w_{bws3} is less than 0, w_{bws3} is set to 0;

c) Otherwise, w_{bws3} is not changed.

– In case $F^{(m)}_{bws}$ is not equal to 1, w_{bws3} is set to 1.

Finally, the decoded coder mode $F^{(m)}_{bw}$ is saved to $F^{(m-1)}_{bw0}$.

B.7.9.2 Wideband to superwideband switching

In case $F^{(m)}_{bw}$ is equal to two, it means that there has been a bandwidth switch from wideband to superwideband. In the first superwideband frame immediately after the bandwidth switching and $F^{(m)}_{bw} \geq MODE\_R2sm$, the MDCT coefficients are generated by SHB frame erasure concealment as described in clause B.7.8.2. In later superwideband frames, the MDCT coefficients are obtained through ordinary decoding procedure. The decoded super higher band frequency coefficients are attenuated by a factor w_{bws} for better switching quality:

$$\hat{S}_{SHB}(k) = w_{bws} \cdot \hat{S}_{SHB}(k) \qquad k = 0, \cdots, 59 \qquad \text{(B.7-120)}$$

Factor w_{bws} is initialized to 0.1. Note that this attenuation is carried on to the next superwideband frames. Factor w_{bws} is updated as follows:

– Factor w_{bws} is incremented by 0.02 and $F^{(m-1)}_{bw1}$ is set to MODE_R0wm. It means that the bitstream coder mode $F^{(m-1)}_{bw1}$ is set to wideband.

– If w_{bws} is larger than 1.0, w_{bws} is set to 0.1; accordingly $F_{bws}^{(m)}$ is set to 0 and $F_{bw1}^{(m-1)}$ is set to $F_{bw}^{(m)}$. It means that no more attenuation is needed and the bitstream coder mode $F_{bw1}^{(m-1)}$ is set to superwideband.

Finally, the decoded coder mode $F_{bw}^{(m)}$ is saved to $F_{bw0}^{(m-1)}$.

B.7.9.3 Additional attenuation for the switching from wideband to superwideband and superwideband to wideband

Additional attenuation using a factor w_{bws1} is performed when there has been a switch from wideband to superwideband and superwideband to wideband, i.e., when $F_{bws}^{(m)}$ is either one or two:

$$\hat{S}_{SHB}(k) = w_{bws1} \cdot \hat{S}_{SHB}(k), \quad \text{if } w_{bws1} < 1.0 \qquad k = 0,\cdots,59 \qquad \text{(B.7-121)}$$

Factor w_{bws1} is initialized to 0.1 and is incremented by 0.02 after the attenuation performed in Equation (B.7-121). If $F_{bws}^{(m)} = 0$ and $F_{bws}^{(m-1)} = 1$, w_{bws1} is reset to 0.1.

B.7.10 Spectral folding of super higher band, signal upscaling and QMF synthesis filterbank

The super higher band synthesis $\hat{s}_{SHB}^{fold}(n)$ is spectrally folded as follows:

$$\hat{s}_{SHB}(n) = (-1)^n \hat{s}_{SHB}^{fold}(n) \qquad n = 0,\cdots,159 \qquad \text{(B.7-122)}$$

A synthesis QMF is used to synthesize the 32-kHz sampled output signal from the wideband decoded signal and super higher band decoded signal. Both 16 kHz sampled decoded signals in the wideband and super higher band are upsampled by a factor of two. Then, the upsampled signals are filtered through the synthesis filter for each band. The coefficients of those two synthesis filters are given by:

$$\begin{cases} h_L^{qmfS}(i) = h_L^{qmfA}(i) \\ h_H^{qmfS}(i) = -h_H^{qmfA}(i) \end{cases} \qquad i = 0,\cdots,31 \qquad \text{(B.7-123)}$$

where h_L^{qmfS} and h_H^{qmfS} are the coefficients of the wideband and super higher band synthesis filter, and h_L^{qmfA} and h_H^{qmfA} are those of the analysis QMF described in clause B.6.3, respectively. The 32 kHz sampled output $\hat{s}_{SWB}(n)$ is obtained by adding the two filtered signals as follows:

$$\hat{s}_{SWB}(n) = \sum_{i=0}^{31} h_L^{qmfS}(i)\,\hat{s}_{WS}(n-i) + \sum_{i=0}^{31} h_L^{qmfS}(n)\,\hat{s}_{SHS}(n-i), \qquad n = 0,\cdots,159 \qquad \text{(B.7-124)}$$

where $\hat{s}_{WS}(n)$ and $\hat{s}_{SHS}(n)$ are the upsampled signals in wideband and super higher band, respectively.

In order to reduce the complexity, the above calculations are optimized as follows. Firstly, two intermediate signals $\bar{s}_{sum}(n)$ and $\bar{s}_{diff}(n)$ are obtained by the following equations:

$$\begin{aligned} \hat{s}_{sum}(n) &= \sum_{i=0}^{15} h_0^{qmf}(i)\left(\hat{s}_{WB}(n-i) + \hat{s}_{SHB}(n-i)\right) \\ \hat{s}_{diff}(n) &= \sum_{i=0}^{15} h_1^{qmf}(n)\left(\hat{s}_{WB}(n-i) - \hat{s}_{SHB}(n-i)\right) \end{aligned} \qquad n = 0,\cdots,79 \qquad \text{(B.7-125)}$$

where $\hat{s}_{WB}(n)$ is the wideband decoded signal, $\hat{s}_{SHB}(n)$ is the super higher band decoded signal and h_0^{qmf} and h_1^{qmf} are the filter coefficients described in Table B.6-1. Then the intermediate signals $\hat{s}_{sum}(n)$ and $\hat{s}_{diff}(n)$ are interleaved to obtain the 32 kHz sampled signal $\hat{s}_{SWB}(n)$ as follows:

$$\begin{aligned} \hat{s}_{SWB}(2n) &= 2\hat{s}_{diff}(n), \\ \hat{s}_{SWB}(2n+1) &= 2\hat{s}_{sum}(n) \end{aligned} \qquad n = 0, \cdots, 79 \qquad \text{(B.7-126)}$$

B.8 Bit-exact description of the ITU-T G.722 superwideband extension coder

The description of the coding algorithm of this annex is made in terms of bit-exact fixed-point mathematical operations. The ANSI C code indicated in this clause, which is an integral part of this annex, reflects this bit-exact, fixed-point descriptive approach. The mathematical description of the encoder and decoder can be implemented in other fashions, possibly leading to a codec implementation not complying with this annex. Therefore, the algorithm description of the ANSI code of this clause shall take precedence over the mathematical descriptions whenever discrepancies are found. A non-exhaustive set of test signals, which can be used with the ANSI C code, is available as an electronic attachment.

B.8.1 Use of the simulation software

The C code consists of two main programs, encoder.c and decoder.c, which simulate the main encoder and main decoder, respectively.

The command line for the encoder is as follows:

encoder [-options] <infile> <codefile> [rate]

where

rate	is the desired encoding bitrate in kbit/s: either 64 or 96 (64 for the ITU-T G.722 core at 56 kbit/s or 96 for the ITU-T G.722 core at 64 kbit/s)
infile	is the name of the input file to be encoded
codefile	is the name of the output bitstream file

Options:

-quiet	quiet processing

The command line for the decoder is as follows:

decoder [-options] <codefile> <outfile> [rate]

where

rate	is the desired decoding bitrate in kbit/s: 64 for ITU-T G.722 R1sm, 80 for ITU-T G.722 R2sm and 96 for ITU-T G.722 R3sm
codefile	is the name of the input bitstream file
outfile	is the name of the decoded output file

Options:

-quiet	quiet processing
-bitrateswitch [bsflag]	bsflag is 1 for ITU-T G.722 core at 56 kbit/s, and 0 for the ITU-T G.722 core at 64 kbit/s

The encoder input and the decoder output files are sampled data files containing 16-bit PCM signals. The default file format for the encoder output and decoder input files follow the [b-ITU-T G.192] bitstream format.

B.8.2 Organization of the simulation software

Table B.8-1 – Summary of encoder specific routines

Filename	Description
encoder.c	ITU-T G.722-SWB encoder interface routine
pcmswbenc.c	ITU-T G.722-SWB main encoder
prehpf.c	High-pass pre-filter routine
bwe_enc.c	SWBL0 encoder
swb_avq_encode.c	SWBL1/SWBL2 encoder
avq_cod.c	AVQ main encoder

Table B.8-2 – Summary of decoder specific routines

Filename	Description
decoder.c	ITU-T G.722-SWB decoder interface routine
pcmswbdec.c	ITU-T G.722-SWB main decoder
bwe_dec.c	SWBL0 decoder
bwe_mdct.c	MDCT routine for ITU-T G.722 post-processor
bwe_mdct_table.c	Tables for MDCT routine for ITU-T G.722 post-processor
swb_avq_decode.c	SWBL1/SWBL2 decoder
avq_dec.c	AVQ main decoder

Table B.8-3 – Summary of common routines

Filename	Description
qmfilt.c	QMF filterbank routine
softbit.c	Routine for conversion between hardbit and softbit
table_qmfilt.c	Tables for QMF filterbank
ns_common.c	ITU-T G.722 noise-shaping routine
dsputil.c	Fixed-point utility routines
errexit.c	Exit routine
mathtool.c	Square-root routines
oper_32b.c	Routine of basic operators in double precision (32 bits)
table_mathtool.c	Tables for square-root routines
table.c	Tables for SWBL0
rom.c	Tables for AVQ

Annex C

Floating-point implementation of ITU-T G.722 Annex B

(This annex forms an integral part of this Recommendation.)

This annex describes the reference floating-point implementation of the ITU-T G.722 Annex B coding algorithm.

C.1 Algorithmic description

This floating-point version of Recommendation ITU-T G.722 Annex B has the same algorithm steps as the fixed-point version. Similarly, the bit stream is identical to that of ITU-T G.722 Annex B. For algorithmic details, see main body and Annex B.

C.2 ANSI C code

ANSI C source code, simulating the floating-point version of Recommendation ITU-T G.722 Annex B defined in this annex, is available as an electronic attachment to Annex C. The current version of this ANSI C source code is Version 1.0 of May 2012. The structure of the floating-point source code is related to the corresponding fixed point source code. Tables C.1 to C.3 give the list of the software files names with a brief description. Note that files related to basic operators or mathematical operations are not used for floating-point arithmetic. A set of floating point related routines have been added as floatutil.c.

Table C.1 – Summary of encoder specific routines

Filename	Description
encoder.c	ITU-T G.722 Annex B encoder interface routine
pcmswbenc.c	ITU-T G.722 Annex B main encoder
prehpf.c	High-pass pre-filter routine
bwe_enc.c	SWBL0 encoder
swb_avq_encode.c	SWBL1/SWBL2 encoder
avq_cod.c	AVQ main encoder

Table C.2 – Summary of decoder specific routines

Filename	Description
decoder.c	ITU-T G.722 Annex B decoder interface routine
pcmswbdec.c	ITU-T G.722 Annex B main decoder
bwe_dec.c	SWBL0 decoder
bwe_mdct.c	MDCT routine for ITU-T G.722 post-processor
bwe_mdct_table.c	Tables for MDCT routine for ITU-T G.722 post-processor
swb_avq_decode.c	SWBL1/SWBL2 decoder
avq_dec.c	AVQ main decoder

Table C.3 – Summary of common routines

Filename	Description
qmfilt.c	QMF filterbank routine
softbit.c	Routine for conversion between hardbit and softbit
table_qmfilt.c	Tables for QMF filterbank
ns_common.c	ITU-T G.722 noise shaping routine
autocorr_ns.c	Autocorrelation of signal for ITU-T G.722 noise shaping
lpctools.c	Linear prediction tools for ITU-T G.722 noise shaping
table_lowband.c	Tables for ITU-T G.722 noise shaping
table.c	Tables for SWBL0
rom.c	Tables for AVQ and G711EL0
floatutil.c	Floating-point utility routines
errexit.c	Exit routine
bit_op.c	Bit operation routines

Annex D

Stereo embedded extension for ITU-T G.722

(This annex forms an integral part of this Recommendation.)

D.1 Scope

This Recommendation contains the description of an algorithm extending ITU-T G.722 for the scalable coding of stereo speech and audio signals from 64 to 80 kbit/s with ITU-T G.722 core at 56 kbit/s or from 80 to 128 kbit/s with ITU-T G.722 core at 64 kbit/s. ITU-T G.722 stereo extension codec is interoperable with ITU-T G.722 for wideband operating modes and with ITU-T G.722 Annex B for superwideband operating modes.

This annex is intended as a stereo extension to the ITU-T G.722 wideband coding algorithm and its superwideband Annex B. Compared to discrete two-channel (dual-mono) audio transmission, this stereo extension ITU-T G.722 Annex D saves valuable bandwidth for stereo transmission. It is specified to offer the stereo capability while providing backward compatibility with the monaural core in an embedded scalable way. This annex provides very good quality for stereo speech contents (clean speech and noisy speech with various stereo sound pickup systems: binaural, MS, etc.), and for most of the conditions it provides significantly higher quality than low bitrate dual-mono. For some music contents, e.g., highly reverberated and/or with diffuse sound, the algorithm may have some performance limitations and may not perform as good as dual-mono codecs, however it achieves the quality of state-of-the-art parametric stereo codecs.

The document is organized as follows. The normative references, abbreviations, acronyms and conventions used throughout this Recommendation are defined in clauses 2, 3 and 4, respectively. Clause 5 gives a general description of the coder. The ITU-T G.722 stereo extension encoder and decoder functional descriptions are discussed in clauses 6 and 7, respectively. Clause 8 describes the software that defines this coder in 16-32 bits fixed-point arithmetic.

D.2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T G.191] Recommendation ITU-T G.191 (2010), *Software tools for speech and audio coding standardization.*

D.3 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

Table D.3-1 – Glossary of abbreviations and acronyms

Abbreviation or acronym	Description
IC	Inter-channel coherence
IPD	Inter-channel phase difference
ITD	Inter-channel time difference
ILD	Inter-channel level difference
FERC	Frame erasure concealment
SL	Stereo Layer
FFT	Fast Fourier transform
iFFT	Inverse FFT
DFT	Discrete Fourier transform
HPF	High pass filter
iMDCT	Inverse MDCT
MDCT	Modified discrete cosine transform
OLA	Overlap and add
QMF	Quadrature mirror filterbank
SHB	Super higher band (8-16 kHz)
SWB	Superwideband (0-16 kHz)
WB	Wideband (0-8 kHz)
WMOPS	Weighted million operations per second

D.4 Conventions

The notational conventions are detailed below:

– Time-domain signals are denoted by their symbol and a sample index between parentheses, e.g., $s(n)$. The variable n is used as sample index.

– Frequency-domain transforms are denoted by converting the related time-domain signal to capital letters, e.g., $S(k)$ is the transform of $s(n)$. The variable k is used as coefficient index.

– Superscript indices between parentheses (e.g., $g^{(i)}$) are used to indicate time-dependency of variables. The variable i refers, depending on the context, to either a frame or sub-frame index.

– Recursion indices are identified by a superscript between square brackets (e.g., $E^{[k]}$).

– The symbol ^ identifies a quantized version of a parameter (e.g., $\hat{g}_c$).

– Parameter ranges are given between square brackets, and include the boundaries (e.g., [0.6, 0.9]).

– The sign function gives the polarity of the value and is denoted as $\mathrm{sgn}(x)$, where $\mathrm{sgn}(x) = \begin{cases} 1 & if\ x \geq 0 \\ -1 & if\ x < 0 \end{cases}$.

– Integer operator $\lfloor x \rfloor$ denotes rounding off x towards minus infinity ($\lfloor x \rfloor = \max\{n \in Z \mid x \geq n\}$).

– Absolute value calculations of x, performed with saturation operation, are denoted with $|x|$.

– The function $round(x)$ denotes the rounding to the nearest integer, i.e., $round(x) = \text{sgn}(x)\lfloor |x| + 0.5 \rfloor$.

– In some parts, bit special operators are used, where $\otimes$ and $\oplus$ represent the AND bit-operator and the XOR bit-operator, respectively.

– The constants with "0x" prefix mean that the values are noted in hexadecimal.

– N-bit right-shift operations of a variable x are denoted as multiplications with floor of 2 to the power of $-N$, i.e., $\lfloor 2^{-N} x \rfloor$.

– The floating-point numbers used are rounded versions of the values used in the 16 bit fixed-point ANSI C implementation.

– * denotes complex conjugate.

– $\text{Re}\{c\}$ is defined as the real part of a complex value *c*.

– $\text{Im}\{c\}$ is defined as the image part of a complex value *c*.

– arctan() is the arctangent function.

– mod is the modulo operation.

– Operation $\angle$ is the argument operator to compute the angle of a complex value *c*, and it is defined as $\arctan\left(\frac{\text{Re}\{c\}}{\text{Im}\{c\}}\right)$.

Table D.4-1 lists the most relevant symbols used throughout this annex.

Table D.4-1 – Glossary of most relevant symbols

Type	Name	Description
Signals	$l_{SWB}(n)$	Left channel SWB input signal
	$r_{SWB}(n)$	Right channel SWB input signal
	$l_{WB}(n)$	Left channel WB input signal or WB signal after QMF processing (decimated)
	$r_{WB}(n)$	Right channel WB input signal or WB signal after QMF processing (decimated)
	$\tilde{l}_{WB}(n)$	Pre-processed left channel WB input signal
	$\tilde{r}_{WB}(n)$	Pre-processed right channel WB input signal
	$\tilde{l}_{SWB}(n)$	Pre-processed left channel SWB input signal
	$\tilde{r}_{SWB}(n)$	Pre-processed right channel SWB input signal
	$l_{SHB}(n)$	Left channel SHB signal after QMF processing (decimated)
	$r_{SHB}(n)$	Right channel SHB signal after QMF processing (decimated)
	$m_{SHB}(n)$	Mono SHB signal after QMF processing (decimated)
	$d_{SHB}(n)$	Difference SHB signal after QMF processing (decimated)

Table D.4-1 – Glossary of most relevant symbols

Type	Name	Description
	$m_{WB}(n)$	Mono WB signal after inverse FFT and OLA
	$L_{SHB}(k)$	Left channel SHB MDCT coefficients
	$R_{SHB}(k)$	Right channel SHB MDCT coefficients
	$M_{SHB}(k)$	Mono SHB MDCT coefficients
	$D_{SHB}(k)$	Difference SHB MDCT coefficients
	$\tilde{M}_{SHB}(k)$	Mono SHB MDCT coefficients after gain correction
	$L_{WB}(k)$	Left channel WB FFT coefficients
	$R_{WB}(k)$	Right channel WB FFT coefficients
	$M_{WB}(k)$	Mono channel WB FFT coefficients
	$\hat{L}_{SHB}(k)$	Decoded left channel SHB MDCT coefficients
	$\hat{L}'_{SHB}(k)$	FERC decoded left channel SHB MDCT coefficients
	$\hat{R}_{SHB}(k)$	Decoded right channel SHB MDCT coefficients
	$\hat{R}'_{SHB}(k)$	FERC decoded right channel SHB MDCT coefficients
	$\hat{L}_{WB}(k)$	Decoded left channel WB FFT coefficients
	$\hat{L}'_{WB}(k)$	FERC decoded left channel WB FFT coefficients
	$\hat{R}_{WB}(k)$	Decoded right channel WB FFT coefficients
	$\hat{R}'_{WB}(k)$	FERC decoded right channel WB FFT coefficients
	$\hat{m}_{WB}(n)$	ITU-T G.722 decoded mono signal
	$\hat{M}_{WB}(k)$	ITU-T G.722 decoded mono channel WB FFT coefficients
	$\hat{M}^{pp}_{WB}(k)$	ITU-T G.722 decoded mono channel WB FFT coefficients after post processing
	$\hat{l}_{WB}(n)$	Decoded WB left channel signal
	$\hat{r}_{WB}(n)$	Decoded WB left channel signal
	$\hat{l}_{SHB}(n)$	Decoded left channel SHB signal
	$\hat{r}_{SHB}(n)$	Decoded right channel SHB signal
	$\hat{l}^{pp}_{SHB}(n)$	Stereo post processed decoded left channel SHB signal
	$\hat{r}^{pp}_{SHB}(n)$	Stereo post processed decoded right channel SHB signal
	$\hat{l}_{SWB}(n)$	Decoded SWB left channel signal
	$\hat{r}_{SWB}(n)$	Decoded SWB right channel signal

Table D.4-1 – Glossary of most relevant symbols

Type	Name	Description
Parameters	$h_0^{qmf}(i)$	QMF coefficient set 1
	$h_1^{qmf}(i)$	QMF coefficient set 2
	$w_{FFT}(i)$	WB FFT window

D.5 General description of the coder

The ITU-T G.722 stereo extension coder has a wideband (WB) and superwideband (SWB) stereo encoding/decoding capability and six scalable operational bitrate modes: ITU-T G.722 core R1ws/R2ws/R2ss/R3ss/R4ss/R5ss. Here, "Rx" specifies the rate and R1, R2, R3, R4 and R5 correspond to 64, 80, 96, 112 and 128 kbit/s modes, respectively. The notation "ws" or "ss" after the rate specifier indicates that the modes are in "WB stereo" or "SWB stereo" respectively. For R1ws and R2ss, the ITU-T G.722 core operates at 56 kbit/s, while it operates at 64kbit/s for other bitrates.

D.5.1 Coder modes and bit allocation

ITU-T G.722 R1ws includes the whole wideband ITD/IPD/IC and sub-band ILD information. ITU-T G.722 R2ws includes the same information as R1ws with additional IPD information. ITU-T G.722 R3ss/R4ss are based on R2ws, and also include the SHB ILD information. ITU-T G.722 R5ss is built on top of R4ss and has additional IPD information. Table D.5-1 illustrates the mode definition of the WB and SWB coder modes.

Table D.5-1 – Stereo mode definition

Coder Mode	Description
MODE_R1ws	ITU-T G.722 R1ws, 64 kbit/s WB stereo
MODE_R2ws	ITU-T G.722 R2ws, 80 kbit/s WB stereo (R1wm core)
MODE_R2ss	ITU-T G.722 R2ss, 80 kbit/s SWB stereo
MODE_R3ss	ITU-T G.722 R3ss, 96 kbit/s SWB stereo (R1wm core)
MODE_R4ss	ITU-T G.722 R4ss, 112 kbit/s SWB stereo (R1wm core)
MODE_R5ss	ITU-T G.722 R5ss, 128 kbit/s SWB stereo (R1wm core)

Three stereo layers, noted SL0, SL1 and SL2 are used. Table D.5-2 gives their bitrates.

Table D.5-2 – Stereo layer bitrates

Layer name	Bits per frame	Bitrate [kbit/s]
SL0	40	8.0
SL1	40	8.0
SL2	80	16.0

The bitstream structure is fully scalable, lower bitrates can be obtained by simply omitting certain parts of the bitstream at a higher bitrate. Layers SL0w is added on top of ITU-T G.722 core at 56 kbit/s to form R1ws bitstream. Layers SL0w and SL1w are added on top of ITU-T G.722 64 kbit/s core to construct R2ws bitstream. Layer SL0s and SL1s are added to ITU-T G.722 Annex B R1sm at 64kbit/s core to form R2ss bitstream. Layer SL0s and SL1s are added on top of ITU-T G.722 Annex B R2sm /R3sm to construct R3ss /R4ss bitstream respectively. Layer SL2 is added on top of

R4ss to construct R5ss bitstream. Here, SL0 is the layer that contains the whole wideband ITD/IPD/IC and sub-band ILD information. SL1 is the layer which contains the IPD information from frequency bins 2 to 9 for WB operating modes (SL1w) and IPD information from frequency bins 2 to 8 together with SWB ILD (SL1s), and SL2 is the layer only used for SWB modes which contains the IPD information from frequency bins 9 to bin 24. Table D.5-3 summarizes the scalable bitstream structure in the six stereo coder modes.

Table D.5-3 – Stereo layers in stereo coder modes

Coder Mode	Core (WB) Layer (kbit/s)	SWB Mono layer	SL0	SL1	SL2	Overall bitrate (kbit/s)
R1ws	56	–	SL0w	–	–	64
R2ws	64	–	SL0w	SL1w	–	80
R2ss	56	R1sm	SL0s	SL1s	–	80
R3ss	64	R2sm	SL0s	SL1s	–	96
R4ss	64	R3sm	SL0s	SL1s	–	112
R5ss	64	R3sm	SL0s	SL1s	X	128

The detailed bit allocation of stereo layers used in WB stereo modes is given in Table D.5-4. The bit allocation is the stereo layer SL0w WB is now described. One bit is needed to indicate the stereo bandwidth (i.e., whether it is WB or SWB bitstream). One bit is used to indicate whether the frame is classified as WB transient stereo. The ILD quantization depends on this classification. For WB transient stereo frame, two-frame mode quantization is used and ten ILDs are coded with 38 bits per frame; otherwise (WB normal stereo frame), four-frame mode quantization is used, five ILDs are coded with 24 bits. There is an ILD refinement in four-frame mode (WB normal stereo frame). The number of bits used for ILD refinement depends on whether whole WB IC is selected to be transmitted or not. In this case, two bits are used to encode this IC parameter and only seven bits are used for ILD refinement. Otherwise (IC is not transmitted), nine bits are used for ILD refinement. The whole wideband ITD/IPD is quantized with five bits. Note that one special value among the 32 possible values is reserved to indicate that IC is transmitted.

On top layer of SL0w, the stereo WB enhancement layer SL1w is used to encode IPDs of eight bins (from bins 2 to 9) with 40 bits.

Table D.5-4 – Detailed bit allocation in stereo layers used for WB stereo modes

Layer Name	Parameters	Two-frame mode (WB transient stereo frame)	Four-frame mode (WB normal stereo frame)	
			whole wideband ITD/IPD	whole wideband IC
SL0w	Bandwidth (SWB/WB) flag	1	1	
	WB frame type (transient/normal)	1	1	
	ILD	38	24	
	Whole wideband ITD/IPD	0	5	
	IC	0	0	2
	ILD refinement	0	9	7
SL1w	IPD (bins 2 to 9)	40	40	

The detailed bit allocation of stereo layers used in SWB stereo modes is listed in Table D.5-5. The bit allocation of the first two stereo layers also used in WB stereo modes is modified as described below. In SL0s, one bit is used to indicate whether the frame is classified as SHB transient stereo frame. This bit is stolen in the bits used for ILD quantization. If the frame is classified as WB transient stereo frame, 37 bits instead of 38 bits are used to quantize the ILDs. Otherwise (if the frame is classified as WB normal stereo frame), this bit is stolen in the bits used for ILD refinement quantization.

In SL1s, one SHB ILD is quantized with five bits and seven IPDs (from bins 2 to 8) are coded per frame (instead of eight IPDs in WB stereo modes). The number of bit allocated to these IPDs depends on the SHB classification. For SHB transient stereo frame, one-frame SHB mode quantization is used and IPDs are coded with 35 bits per frame. Otherwise (SHB normal stereo frame), two-frame SHB mode quantization is used and one bit is used to indicate the SHB frame index and IPDs are coded with 34 bits per frame.

In SL2, 16 IPDs (from bins 9 to 24) are coded with 80 bits.

Table D.5-5 – Detailed bit allocation in stereo layers used for SWB stereo modes

Layer	Parameters	SHB one-frame mode (SHB transient stereo frame)			SHB two-frame mode (SHB normal stereo frame)		
		WB two-frame mode (WB transient stereo frame)	WB four-frame mode (WB normal stereo frame)		WB two-frame mode (WB transient stereo frame)	WB four-frame mode (WB normal stereo frame)	
			whole wideband ITD/IPD	whole wideband IC		whole wideband ITD/IPD	whole wideband IC
SL0s	Bandwidth (SWB/WB) flag	1	1		1	1	
	WB frame mode (transient/normal)	1	1		1	1	
	SHB frame mode (transient/normal)	1	1		1	1	
	WB ILD	37	24		37	24	
	Whole wideband ITD/IPD	0	5		0	5	
	Whole wideband IC	0	0	2	0	0	2
	ILD refinement	0	8	6	0	8	6
SL1s	IPD (bins 2 to 8)	35	35		34	34	
	SHB frame index	0	0		1	1	
	SHB ILD	5	5		5	5	
SL2	IPD (bins 9 to 24)	80	80		80	80	

D.5.2 Input/output sampling rate

The encoder operates with a 16-bit linear PCM digital signal sampled at 16 kHz for WB and 32 kHz for SWB as input. Similarly, decoder output is 16-bit linear PCM with sampling frequency of 16 kHz for WB and 32 kHz for SWB. Other input/output formats should be converted to 16-bit linear PCM with16 kHz or 32 kHz sampling rate before encoding or from 16-bit linear PCM to the appropriate format after decoding. The bitstream is defined within this Annex.

D.5.3 Algorithmic delay

The ITU-T G.722 WB stereo extension coder operates with 5-ms frames and has an algorithmic delay of 13.625 ms. The delay contributions are listed below:

– 5 ms for input frame;

– 3.625 ms for the down-mix;

– 1.375 ms for the QMF analysis-synthesis filterbank in ITU-T G.722 core (WB);

– 3.625 ms for the stereo analysis.

The ITU-T G.722 SWB stereo extension coder has an algorithmic delay of 15.9375 ms. The following delay contributions are added to the WB delay contributions listed above:

– 0.9375 ms for QMF analysis-synthesis filterbank for SWB layers;

– 1.375 ms for compensating the delay of SWB stereo.

D.5.4 Computational complexity and storage requirements

The observed worst-case complexity and storage requirements of the ITU-T G.722 stereo extension coder (encoder plus decoder) are obtained based on the basic operators of ITU-T Software Tool Library STL2009 in [ITU-T G.191] and are detailed in Table D.5-6. The figures are based on in 16-bit words and given according to the mode.

Table D.5-6 – Complexity of ITU-T G.722-stereo coder

Mode		R1ws	R2ws	R2ss	R3ss	R4ss	R5ss
Worst case complexity (in WMOPS)	Encoder	13.53	13.81	20.55	21.97	23.173	23.28
	Decoder	15.05	14.72	17.33	18.65	19.147	18.85
	Overall	28.58	28.53	37.88	40.62	42.32	42.13
Dynamic RAM and Static RAM		7 906 kwords					
Data ROM		6 336 kwords					
Program ROM		9 174 operators					

D.6 Functional description of the encoder

D.6.1 Encoder overview

The encoder block diagram of ITU-T G.722 WB stereo extension is shown in Figure D.6-1. A pre-processing high-pass filter is applied to the 16-kHz sampled left and right channel input signal $l_{WB}(n)$ and $r_{WB}(n)$, to remove 0-50 Hz components. The pre-processed WB signals $\tilde{l}_{WB}(n)$ and $\tilde{r}_{WB}(n)$ are transformed into frequency domain as $L_{WB}(k)$ and $R_{WB}(k)$ by FFT. WB frequency domain down-mix is performed on $L_{WB}(k)$ and $R_{WB}(k)$ to generate frequency domain mono signal $M_{WB}(k)$. $M_{WB}(k)$ is converted back to time domain signal $m_{WB}(n)$ and encoded with an ITU-T G.722 enhanced core encoder which produces an ITU-T G.722 bitstream. Stereo parameters are estimated from $L_{WB}(k)$ and $R_{WB}(k)$ and written into the bitstream.

The encoder block diagram of ITU-T G.722 SWB stereo extension is shown in Figure D.6-13. A pre-processing high-pass filter is applied to the 32-kHz sampled left and right channel input signal $l_{SWB}(n)$ and $r_{SWB}(n)$ to remove 0-50 Hz components. The pre-processed signals $\tilde{l}_{SWB}(n)$ and $\tilde{r}_{SWB}(n)$ are divided into two 16-kHz sampled WB (0-8 kHz) signals, $l_{WB}(n)$ and $r_{WB}(n)$ and two SHB (8-16 kHz) signals $l_{SHB}(n)$, $r_{SHB}(n)$, using a 32-tap quadrature mirror filterbank (QMF) applied to the 5-

ms input frame size. The WB signals $l_{WB}(n)$ and $r_{WB}(n)$ are transformed into frequency domain as $L_{WB}(k)$ and $R_{WB}(k)$ by FFT. WB frequency domain down-mix is performed using $L_{WB}(k)$ and $R_{WB}(k)$ to generate a frequency domain mono signal $M_{WB}(k)$. $M_{WB}(k)$ is converted back to time domain signal $m_{WB}(n)$ and encoded with a ITU-T G.722 enhanced core encoder which produces an ITU-T G.722 bitstream. Stereo parameters are estimated from $L_{WB}(k)$ and $R_{WB}(k)$ and written into the bitstream. The left and right SHB signals $l_{SHB}(n)$ and $r_{SHB}(n)$ are first down-mixed in the time domain to generate mono signal $m_{SHB}(n)$ and difference signal $d_{SHB}(n)$, which are then transformed into modified discrete cosine transform (MDCT) domain as $M_{SHB}(k)$ and $D_{SHB}(k)$, respectively. The MDCT domain coefficients of left and right channels SHB $L_{SHB}(k)$ and $R_{SHB}(k)$ are generated from $M_{SHB}(k)$ and $D_{SHB}(k)$. $L_{SHB}(k)$ and $R_{SHB}(k)$ are used to estimated SHB stereo parameters. After gain correction, the MDCT domain mono SHB signal $\hat{M}_{SHB}(k)$ is encoded by the ITU-T G.722 Annex B encoder.

D.6.2 WB Stereo Encoder

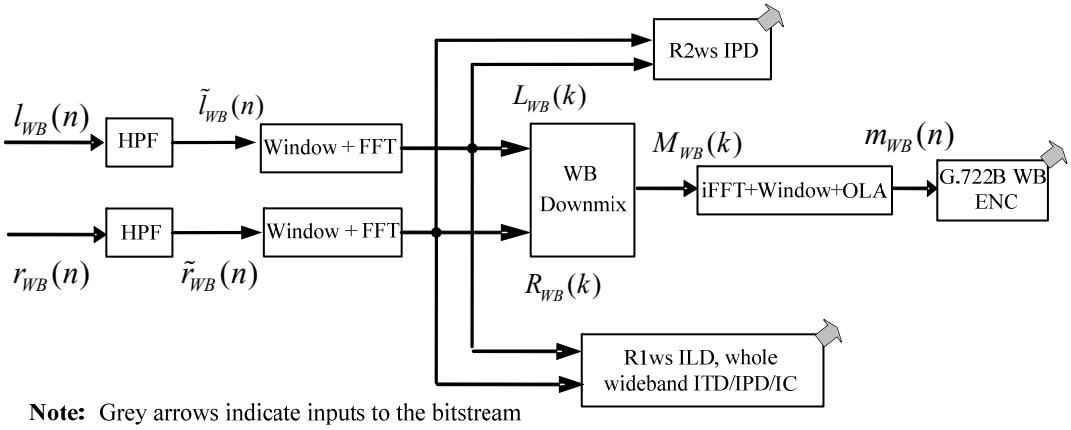


Figure D.6-1 – High-level encoder block diagram of ITU-T G.722 WB stereo extension

D.6.2.1 Pre-processing high-pass filter

The pre-processing filter to remove 0-50 Hz components applied to the 16-kHz sampled input signal is defined as

$$\tilde{s}_{WB}(n) = 0.984375 \cdot \tilde{s}_{WB}(n-1) + s_{WB}(n) - s_{WB}(n-1), \qquad n = 0, \cdots, 79, \qquad \text{(D.6-1)}$$

where $s_{WB}(n)$ is either $l_{WB}(n)$ or $r_{WB}(n)$, and high-pass filtered output $\tilde{s}_{WB}(n)$ is either $\tilde{l}_{WB}(n)$ or $\tilde{r}_{WB}(n)$.

D.6.2.2 FFT

The pre-processed time domain input 16-kHz left and right channel signals $\tilde{l}_{WB}(n)$ and $\tilde{r}_{WB}(n)$ are transformed to frequency domain by FFT. A complex FFT on 80-point is used to obtain a set of 160-point real FFT coefficients. Note that in the following, for brevity, $s_{16}(n)$ represents either $\tilde{l}_{WB}(n)$ or $\tilde{r}_{WB}(n)$. The same buffer $x(n)$ is used for both FFT input and output. The input buffer is composed of 58 samples from the previous frame, together with the 80 samples from the current frame. $s_{16}(n)$ takes its indices between -58 and -1 for the previous frame and between 0 and 79 for

the current frame. The samples are multiplied by the window $w_{FFT}(n)$ before being sent to the input buffer.

$$x(n)=\begin{cases}0 & n=0,\cdots,10\\ w_{FFT}(n)\cdot s_{16}(n-69), & n=11,\cdots,148,\\ 0 & n=149,\cdots,159\end{cases} \qquad \text{(D.6-2)}$$

The window is defined as follows:

$$w_{FFT}(n)=\begin{cases}0 & 0\le n<11\\ \sin((n-11+0.5)\pi/116) & 11\le n<69\\ 1 & 69\le n<91\\ \sin((n-33+0.5)\pi/116) & 91\le n<149\\ 0 & 149\le n<160\end{cases} \qquad \text{(D.6-3)}$$

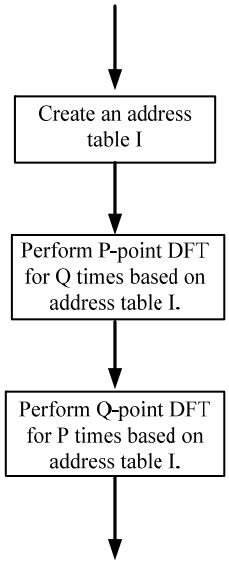


Figure D.6-2 – Flowchart of the FFT

Figure D.6-2 is a flowchart of the FFT. The method includes the following steps:

The 160-point input signal $x(n)$ is used to generate an 80-point complex data signal $v(n)$.

$$\begin{aligned}\mathrm{Re}\{v(n)\}&=x(2n),\\ \mathrm{Im}\{v(n)\}&=x(2n+1)\end{aligned} \qquad n=0,\cdots,79, \qquad \text{(D.6-4)}$$

A complex DFT on 80 points is applied to the complex data $v(n)$:

$$r(k)=\sum_{n=0}^{79}W_{80}^{nk}\cdot v(n), \qquad k=0,\cdots,79, \qquad \text{(D.6-5)}$$

where $W_{80}=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{80}}=\cos(\frac{2\pi}{80})-j\sin(\frac{2\pi}{80})$.

Here, a simple power-2 DFT is not suitable so it is implemented with the following low complexity 2-dimensional ($P\times Q=80$) DFT, where P=5 and Q=16 are coprime positive integers:

To reduce complexity, an address table is introduced. It is calculated by:

$$I(n_1, n_2) = \left(\mathrm{K}_1 \cdot n_1 + \mathrm{K}_2 \cdot n_2\right) \bmod 80, \qquad \begin{matrix} n_1 = 0, \cdots, P-1; \\ n_2 = 0, \cdots, Q-1 \end{matrix}, \qquad \text{(D.6-6)}$$

where K_1 and K_2 are coprime positive integers and satisfy the condition $\left(\mathrm{K}_1\mathrm{K}_2\right) \bmod 80 = 0$. The length of the address table is equal to 80.

Here, $\mathrm{K}_1 = 96, K_2 = 65$. In order to save complexity, the address table *I* is calculated ahead of time and stored as one-dimensional array.

It is used to indicate which samples are used for P-point DFT or Q-point DFT following:

(a) Applying P-point DFT to $v(n)$ for Q times based on the address table *I*.

The input data to the *i*-th ($\mathrm{i} = 0, ..., Q-1$) P-point DFT is found by seeking their addresses stored in the address table *I*. For the *i*-th P-point DFT, the addresses of the input data are the P continuous elements starting from the $\mathrm{i} * \mathrm{P}$'s element in table *I*. For every time of P-point DFT, the resulting data need to be applied a circular shift with a step of m_1. m_1 is the re-ordered index, which satisfies

$$\left(m_1 \cdot \left(\left(\frac{\mathrm{K}_1^2}{\mathrm{Q}}\right) \bmod P\right)\right) \bmod P = 1, \qquad \text{(D.6-7)}$$

The output of step (a) is:

$$w(k) = DFT\_P(v(I + iP))_{m_1}, \qquad i = 0, \cdots, Q-1, \qquad \text{(D.6-8)}$$

For the *i*-th ($\mathrm{i} = 0, ..., 15$) 5-point DFT, the addresses of the input data are the 5 continuous elements starting from $I(5i)$, and the results are circular shifted with m_1=1. Here is an example of circular shift, the original vector is $Z = [z_0\ z_1\ z_2\ z_3\ z_4]$, the new vector with 2 circular shifted is $Z = [z_0\ z_2\ z_4\ z_1\ z_3]$.

(b) Applying Q-point DFT to $w(k)$ for P times based on the address table *I*.

The input data to the *i*-th ($\mathrm{i} = 0, ..., P-1$) Q-point DFT is found by seeking their addresses stored in the address Table *I*. For the *i*-th Q-point DFT, the addresses of the input data are the Q elements starting from the *i*-th element in table *I* each of which are separated by a step of P. For every time of Q-point DFT, the resulting data need to be applied a circular shift with a step of m_2. m_2 is the re-ordered index, which satisfies $\left(m_2 \cdot \left(\left(\frac{\mathrm{K}_2^2}{\mathrm{P}}\right) \bmod Q\right)\right) \bmod Q = 1$.

The output of step (b) is:

$$r(k) = DFT\_Q(w(I + i))_{\mathrm{m}_2}, \qquad i = 0, \cdots, P-1, \qquad \text{(D.6-9)}$$

For the *i*-th ($\mathrm{i} = 0, ..., 4$) 16-point DFT, the addresses of the input data are the 16 continuous elements starting from $I(i)$ each of which are separated by a step of 5, and the results are circular shifted with m_2=5.

Here, *x* is also used to indicate the output buffer. The first 80 points in the output buffer are the real part of $X_{WB}(k)$, and the last 80 points in the output buffer are the image part of $X_{WB}(k)$.

$$\begin{matrix} x(0) = \mathrm{Re}\{r(0)\} + \mathrm{Im}\{r(0)\} \\ x(159) = \mathrm{Re}\{r(0)\} - \mathrm{Im}\{r(0)\} \end{matrix}, \qquad i = 1, \ldots, 79, \qquad \text{(D.6-10)}$$

$$
\begin{aligned}
x(i) &= (\mathrm{Re}\{r(i)\} + \mathrm{Re}\{r(80-i)\} \\
&\quad ... + \mathrm{cosw}(i)(\mathrm{Im}\{r(i)\} + \mathrm{Im}\{r(80-i)\}) \\
&\quad ... - \mathrm{sinw}(i)(\mathrm{Re}\{r(i)\} + \mathrm{Re}\{r(80-i)\}))/2 \\
x(80+i) &= (\mathrm{Im}\{r(i)\} - \mathrm{Im}\{r(80-i)\} \\
&\quad ... - \mathrm{sinw}(i)(\mathrm{Im}\{r(i)\} + \mathrm{Im}\{r(80-i)\}) \\
&\quad ... - \mathrm{cosw}(i)(\mathrm{Re}\{r(i)\} - \mathrm{Re}\{r(80-i)\}))/2
\end{aligned}
$$

where sinw(i) and cosw(i) are defined as follows:

$$
\begin{aligned}
\mathrm{sinw}(i) &= \sin(i\pi/80) \\
\mathrm{cosw}(i) &= \cos(i\pi/80)
\end{aligned}
\text{,} \qquad \text{(D.6-11)}
$$

D.6.2.3 Inter-channel cues

D.6.2.3.1 Overview of inter-channel cues

The inter-channel difference describes the differences between two channels, which includes inter-channel time difference (ITD), inter-channel phase difference (IPD), inter-channel coherence (IC), and inter-channel level difference (ILD).

ITD represents the time difference between two channels and is associated with inter-aural time difference, i.e., the difference in arrival time of a sound between two ears. It is important for the localization of sounds, as it provides a cue to identify the direction or angle of incidence of the sound source (relative to the head). In this recommendation, it is defined that if a waveform to the left ear comes first, ITD is positive, otherwise, it is negative. If the sound source is directly in front of the listener, the waveform arrives at the same time at both ears and ITD is zero.

IPD specifies the relative phase difference between the stereo input channels. A sub-band IPD is generally used as an estimate of the sub-band ITD.

IC is defined as the normalized inter-channel cross-correlation or inter-channel cross-coherence coefficient after phase alignment according to the IPD/ITD. IC represents the width of the stereo image.

ITD, IPD and IC are important parameters for parametric stereo codecs. ITD covers the range of audible delays (between -1.5 ms to 1.5 ms), IPD covers the full range of phase difference (between $-\pi$ to π) and IC covers the range of correlation (between 0 and 1).

ITD, IPD and IC are estimated in the frequency domain, But due to the bit budget limitation for very low bitrate parametric stereo audio-coding scheme, there are not enough bits to transmit all sub-band inter-channel differences in one frame. Whole wideband ITD, IPD and IC are then transmitted to reduce the bitrate. These whole wideband parameters represent a single inter-channel difference for the complete spectrum. In this recommendation, even whole wideband ITD, IPD and IC cannot be transmitted at the same time. A selection on what to transmit is performed based on the signal characteristics.

D.6.2.3.2 Cross-spectrum computation and smoothing

As described in the previous clause, the inter-channel parameters, such as the whole wideband ITD, IPD and IC, are estimated in frequency domain and the estimation is based on cross correlation spectrum.

In the first step, a cross-spectrum $c(k)$ is computed for each frequency bin, from the left $L_{WB}(k)$ and right $R_{WB}(k)$ signals as:

$$
c(k) = L_{WB}(k) \cdot R^{*}_{WB}(k)\text{,} \qquad \text{(D.6-12)}
$$

The ranges of the frequency bins used to estimate the whole wideband ITD, IPD and the IC are [3,12], [2, 6], [1, 10], corresponding to the frequency region [300 Hz, 1 200 Hz], [200 Hz, 600 Hz] and [100 Hz, 1 000 Hz] respectively, while individual IPDs are calculated for frequency bins 0 to 80 (resp.70) in SWB (resp. WB) stereo modes.

In the second step, in order to make the estimation more stable, smoothed cross spectrum is used to compute the whole wideband ITD and IPD. Two smoothed cross-spectrum, one strongly smoothed $c_{strong}^{(i)}(k)$ and another weakly smoothed $c_{weak}^{(i)}(k)$, are calculated from the cross-spectrum $c(k)$ computed in the first step as:

$$\begin{aligned} c_{strong}^{(i)}(k) &= W_{strong} \cdot c_{strong}^{(i-1)}(k) + (1 - W_{strong}) \cdot c(k), && k = 3,...,12, \\ c_{weak}^{(i)}(k) &= W_{weak} \cdot c_{weak}^{(i-1)}(k) + (1 - W_{weak}) \cdot c(k) && k = 1,...,12 \end{aligned} \quad \text{(D.6-13)}$$

with $W_{strong} > W_{weak}$, the strong smoothing coefficient W_{strong} being equal to 0.9844 whereas the weak smoothing coefficient W_{weak} being equal to 0.75; and *i* is the current frame index. The weaker smoothing is an almost instantaneous estimation of the cross correlation thus reducing the memory effect and is used to follow fast changes of the parameter. The strong smoothing keeps the estimation as stable as possible, which eliminates almost all the unwanted fluctuation when the stereo image is stable.

D.6.2.3.3 Whole wideband Inter-channel time difference estimation

The weakly smoothed cross-spectrum $c_{weak}^{(i)}(k)$ and the strongly smoothed cross-spectrum $c_{strong}^{(i)}(k)$ are used to estimate two versions of the whole wideband ITD. Among the two estimations, the best one is kept, and the decision is based on a quality metric of the estimated ITD_{weak}.

The estimation of the whole wideband ITD is illustrated in Figure D.6-3.

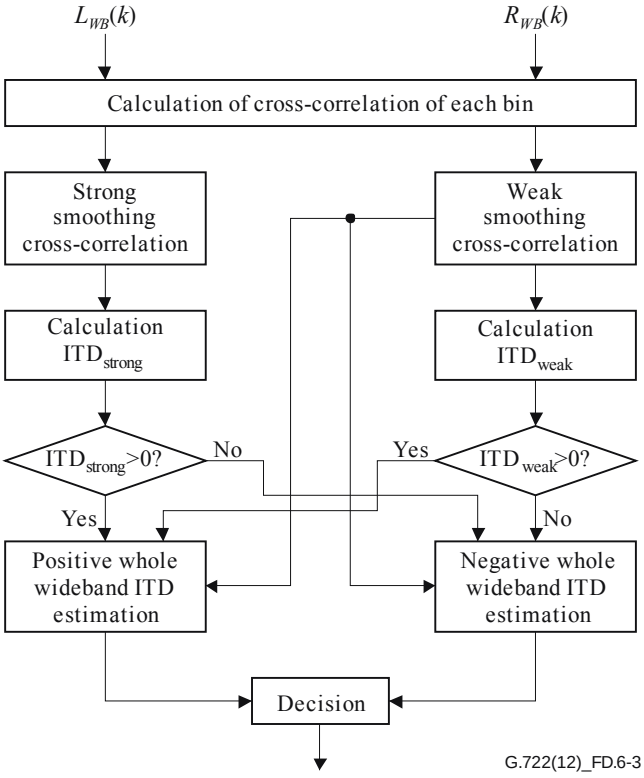


Figure D.6-3 – High level of description of the whole wideband ITD estimation

In the following, for sake of conciseness, the subscript *sm* is used to be either *strong* or *weak*.

First inter-channel phase and time difference, $IPD_{sm}(k)$ and $ITD_{sm}(k)$, are calculated from the smoothed cross-spectrum c_{sm} in the frequency bin ranges [3,12] as:

$$IPD_{sm}(k) = \angle c_{sm}^{(i)}(k),$$
$$ITD_{sm}(k) = \frac{160}{k\pi} \angle c_{sm}^{(i)}(k) \qquad k = 3,...,12, \qquad \text{(D.6-14)}$$

For the weakly smoothing case, the first two $IPD_{weak}(k)$ (k = 1 and 2) are also computed in order to be used in the next sub-clauses as $IPD_{weak}(k)$ from bins 2 to 6 are used in sub-clause D.6.2.3.4 to compute the whole wideband IPD and $IPD_{weak}(k)$ from bins 1 to 10 are used in sub-clause D.6.2.3.5 to compute the whole wideband IC.

Then, if $ITD_{sm}(k)$ is positive, i.e., $ITD_{sm}(k) > 0$, it is sent to the positive whole wideband ITD estimation block; otherwise (if $ITD_{sm}(k)$ is negative, i.e., $ITD_{sm}(k) < 0$), it is sent to the negative whole wideband ITD estimation block. The estimation blocks are described in the next sub-clause.

D.6.2.3.3.1 Positive and negative whole wideband ITD estimation

Within the positive and negative whole wideband ITD estimation block, whole wideband positive and negative ITD are estimated. In this clause, only the ITD estimation of the positive block is detailed. For the negative blocks, the same algorithm as the one for positive ITD is used. In this case, the notation of "+"(positive) should be replaced by "-"(negative).

The mean of ITD, μ_{ITDsm+} is calculated over all the bins in the interval $k \in$ [3, 12]. The frequency bins in which ITD is positive and such that IPD is below 0.3203, form the positive set. This set of such indices *k* is denoted k_+ :

$$\mu_{ITDsm+} = \frac{1}{N_{+sm}} \sum_{k \in k_{+sm}} ITD_{sm}(k) \qquad k_{sm+} = \{k \in [3,12] \cap \bigcap ITD_{sm}(k) > 0 \cap IPD_{sm}(k) < 0.3203\}, \quad \text{(D.6-15)}$$

where N_{sm+} is the cardinal of k_{sm+}, i.e., the number of positive bins to be taken into account for the mean calculation.

For the strongly smoothed coefficients, if $N_{sm+} > 0$, the standard deviation of ITDs is then calculated over all the positive bins:

$$\sigma_{ITDsm+} = \frac{1}{N'_{sm+}} \sqrt{\sum_{k \in k'_{sm+}} (ITD_{sm}(k) - \mu_{ITDsm+})^2} \qquad k'_{sm+} = \{k \in [3,12] \cap \bigcap ITD_{sm}(k) > 0\}, \qquad \text{(D.6-16)}$$

where N'_{sm+} is the cardinal of k'_{sm+}. Note that if $N_{sm+} = 0$, then $\sigma_{ITDsm+} = 7$. If the energies of left and right channels are very low, N'_{sm+} is directly set to 0, and the ITD standard deviation σ_{ITDsm+} becomes 7.

The details are given in Figure D.6-4.

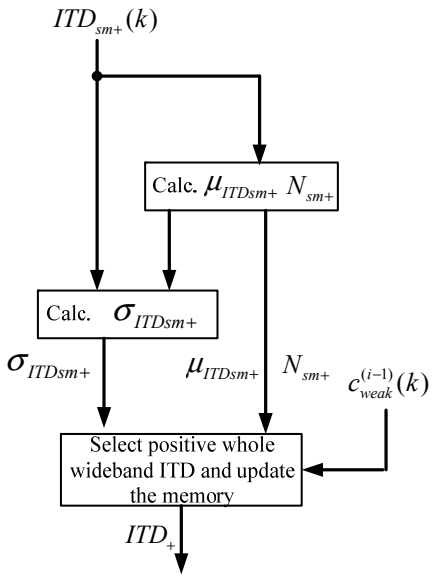


Figure D.6-4 – Positive whole wideband ITD estimation

The selection between the outputs of the two positive smoothing blocks depends on the weak smoothing positive standard deviation σ_{ITDw+} and the number N_{w+} of positive bins taken into account in the average. If σ_{ITDw+} is below a threshold set to 2.0 or if N_{w+} is equal to 10, then weak positive smoothing is selected, otherwise strong positive smoothing is selected.

$$\begin{cases} ITD_+ = \mu_{ITDw+} \\ \sigma_{ITD+} = \sigma_{ITDw+} \\ N_+ = N_{w+} \end{cases} \quad \text{if } \sigma_{ITDw+} < 2.0 \text{ and } N_{w+} = 10, \qquad \text{(D.6-17)}$$

$$\begin{cases} ITD_+ = \mu_{ITDs+} \\ \sigma_{ITD+} = \sigma_{ITDs+} \\ N_+ = N_{s+} \end{cases} \quad \text{otherwise}$$

Moreover, if weak smoothing is selected, the memory of the strong smoothing cross spectrum is updated by the current frame weak smoothing cross spectrum: i.e., if weak smoothing is selected either by the positive or the negative whole wideband ITD estimation blocks: $c_{strong}^{(i)} = c_{weak}^{(i)}$ otherwise $c_{strong}^{(i)}$ is unchanged.

The state variables, i.e., memory of the positive whole wideband ITD estimation block, N_+, σ_{ITD+} and ITD_+ are also updated with the parameters of the selected positive smoothing.

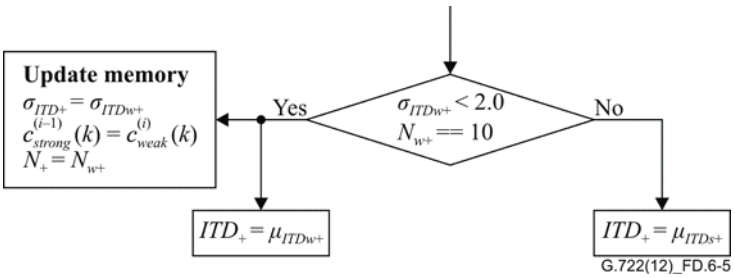


Figure D.6-5 – Whole wideband positive ITD selection and memory update

As mentioned above, the negative whole wideband ITD estimation block is similar to the positive block, N_{sm-} is the cardinal of $k_{sm-} = \{k \in [3,12] \cap \bigcap ITD_{sm}(k) < 0 \cap IPD_{sm}(k) < 0.3203\}$. N'_{sm-} is the cardinal of $k'_{sm-} = \{k \in [3,12] \cap \bigcap ITD_{sm}(k) < 0\}$. N_-, σ_{ITD-} and ITD_- are selected similarly to the positive whole ITD estimation block. Moreover, if weak smoothing is selected, the memory of the strong smoothing cross spectrum is updated by the current frame weak smoothing cross spectrum.

Finally, the memory of the weak smoothing cross spectrum $c_{weak}^{(i)}$ is updated and stored for the next frame.

In order to make the whole wideband ITD estimation more stable, ITD_+ and ITD_- are smoothed between two consecutive frames,

$$\begin{aligned} ITDsm_+^{(i)} &= W_{strong} \cdot ITDsm_+^{(i-1)} + (1 - W_{strong}) \cdot ITD_+, \\ ITDsm_-^{(i)} &= W_{strong} \cdot ITDsm_-^{(i-1)} + (1 - W_{strong}) \cdot ITD_- \end{aligned} \tag{D.6-18}$$

N_+, N_- and σ_{ITD+}, σ_{ITD-} are also smoothed between two consecutive frames,

$$\begin{aligned} Nsm_+^{(i)} &= W_{strong} \cdot Nsm_+^{(i-1)} + (1 - W_{strong}) \cdot N_+, \\ Nsm_-^{(i)} &= W_{strong} \cdot Nsm_-^{(i-1)} + (1 - W_{strong}) \cdot N_- \end{aligned} \tag{D.6-19}$$

$$\begin{aligned} \sigma sm_{ITD+}^{(i)} &= W_{strong} \cdot \sigma sm_{ITD+}^{(i-1)} + (1 - W_{strong}) \cdot \sigma_{ITD+}, \\ \sigma sm_{ITD-}^{(i)} &= W_{strong} \cdot \sigma sm_{ITD-}^{(i-1)} + (1 - W_{strong}) \cdot \sigma_{ITD-} \end{aligned} \tag{D.6-20}$$

If the weak smoothing is selected, the memory $Nsm_+^{(i-1)}$ and $Nsm_-^{(i-1)}$ are updated by N_{w+} and N_{w-}, respectively, otherwise the memory $Nsm_+^{(i-1)}$ and $Nsm_-^{(i-1)}$ are updated by N_{s+} and N_{s-}.

The memory update for the standard deviation is the same. If the weak smoothing is selected, the memory $\sigma sm_{ITD+}^{(i-1)}$ and $\sigma sm_{ITD-}^{(i-1)}$ are updated by σ_{ITDw+} and σ_{ITDw-}, respectively, otherwise the memory $\sigma sm_{ITD+}^{(i-1)}$ and $\sigma sm_{ITD-}^{(i-1)}$ are updated by σ_{ITDs+} and σ_{ITDs-}.

The non-meaningful $ITDsm_+^{(i)}$ and $ITDsm_-^{(i)}$ are set to zero based on $Nsm_+^{(i)}$, $Nsm_-^{(i)}$, the whole wideband ITD in the previous frame pre_ITD and the smoothed standard deviation of ITD $\sigma sm_{ITD+}^{(i)}$ and $\sigma sm_{ITD-}^{(i)}$. This verification of $ITDsm_+^{(i)}$ and $ITDsm_-^{(i)}$ is performed before being used for the final selection.

If $Nsm_+^{(i)} \le 7 \,\&\&\, pre\_ITD \ne 0$ or $Nsm_+^{(i)} \le 8 \,\&\&\, pre\_ITD == 0$ or $\sigma sm_{ITD+}^{(i)} \ge 3$, $ITDsm_+^{(i)} = 0$

For the negative part, if $Nsm_{-}^{(i)} \le 7 \&\& pre\_ITD \ne 0$ or $Nsm_{-}^{(i)} \le 8 \&\& pre\_ITD == 0$ or $\sigma sm_{ITD-}^{(i)} \ge 3$, $ITDsm_{-}^{(i)} = 0$.

D.6.2.3.3.2 Whole wideband inter-channel time difference estimation

In the decision part of Figure D.6-6, the final whole wideband ITD is selected from final positive ITD and final negative ITD based on the positive and negative standard deviations and number of positive ITD and negative ITD.

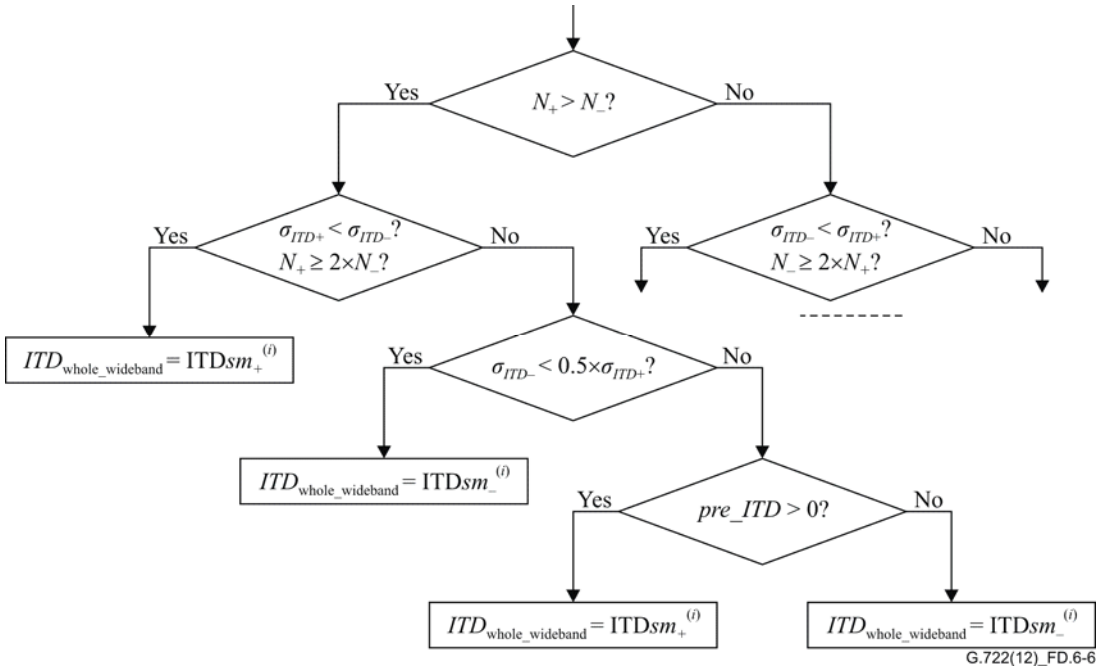


Figure D.6-6 – Flow chart of ITD selection algorithm between positive negative ITD

1. If $N_+ > N_-$: the standard deviation and number of positive and negative ITD will be compared.

   1.1 If $\sigma_{ITD+} < \sigma_{ITD-}$ or $N_+ \ge 2 \cdot N_-$: ITD will be selected as the mean of positive ITD. Otherwise, the relation between positive and negative ITD will be further checked.

      1.1.1 If $\sigma_{ITD-} < 0.5 \cdot \sigma_{ITD+}$: the opposite value of negative ITD mean will be selected as output ITD. Otherwise, ITD form previous frame (*pre_ITD*) will be checked.

         1.1.1.1 If *pre_ITD*>0: ITD becomes the mean of positive ITD. Otherwise, the ITD is selected at the opposite value computed as the mean of negative ITD.

2. Else, the selection procedure is the same except "positive" should be replaced by "negative" and vice versa.

D.6.2.3.4 Whole wideband inter-channel phase difference estimation

The whole wideband IPD is estimated from the inter-channel phase differences (IPDs) calculated bin by bin in the region [2,6] based on the weakly smoothed cross-spectrum $c_{weak}^{(i)}(k)$. The weakly smoothed cross-spectrum $c_{weak}^{(i)}(k)$ is calculated in sub-clause D.6.2.3.2 whereas the inter-channel phase differences $IPD_{weak}(k)$, $(IPD_{weak}(k) = \angle c_{weak}^{(i)}(k))$ in sub-clause D.6.2.3.2. In the following, the notation *weak* is omitted in order to simplify the notation.

The whole wideband IPD is a sum of the weighted IPDs in the frequency region between the bins 2 and 6. The weighting coefficients are calculated as follows:

1. The current energy $e^{(i)}(k)$ of each frequency bin in the region [2,6] is calculated from the left $L_{WB}(k)$ and right $R_{WB}(k)$ signals as:

$$e^{(i)}(k) = L_{WB}(k) \cdot L_{WB}^{*}(k) + R_{WB}(k) \cdot R_{WB}^{*}(k), \qquad k = 2,...,6, \qquad \text{(D.6-21)}$$

where *i* is the current frame index

2. The energies of these five WB bins are added to calculate the WB energy:

$$E = \sum_{k=2}^{6} e^{(i)}(k), \qquad \text{(D.6-22)}$$

3. The energy of the frequency bin is normalized by this energy sum:

$$e_N^{(i)} = \frac{e^{(i)}}{E}, \qquad \text{(D.6-23)}$$

4. The weighting coefficients are then calculated by smoothing the normalized energy between consecutive frames.

$$e_w^{(i)}(k) = SM_1 \cdot e_w^{(i-1)}(k) + (1 - SM_1) \cdot e_N^{(i)}(k), k = 2,...,6 \qquad \text{(D.6-24)}$$

where SM_1=0.984375 is the smoothing factor.

NOTE – At the first frame (*i*=0), no smoothing is performed and the normalized energy is used as the weighting coefficient $e_w^{(i)}(k) = e_N^{(i)}(k)$.

5. The weighted IPD of each bin is calculated by multiplying $IPD(k)$ with the corresponding energy weighting factor.

$$IPD_w(k) = e_w^{(i)}(k) \cdot IPD(k), k = 2,...,6 \qquad \text{(D.6-25)}$$

6. The whole wideband IPD is calculated by adding the five weighted IPDs:

$$IPD_{whole\_wideband} = \sum_{k=2}^{6} IPD_w(k), \qquad \text{(D.6-26)}$$

Due to the phase wrap problem, further check is needed. For each IPD, when $IPD_{whole\_wideband} > 0$, IPD will be refined by the following equation,

$$IPD(k) = \begin{cases} IPD(k) + 2\pi, & IPD(k) < 0 \,\&\&\, \left| IPD(k) - IPD_{whole\_wideband} \right| > \frac{\pi}{4} \\ IPD(k), & others \end{cases} \quad k = 2,...,6, \qquad \text{(D.6-27)}$$

when $IPD_{whole\_wideband} < 0$, IPD will be refined by the following equation

$$IPD(k) = \begin{cases} IPD(k) - 2\pi, & IPD(k) > 0 \,\&\&\, \left| IPD(k) - IPD_{whole\_wideband} \right| > \frac{\pi}{4} \\ IPD(k) & others \end{cases} \quad k = 2,...,6, \qquad \text{(D.6-28)}$$

After refinement, the whole wideband IPD $IPD_{whole\_wideband}$ will be recalculated again based the refined IPD.

In order to keep the stability of $IPD_{whole\_wideband}$ as much as possible, fast change of the $IPD_{whole\_wideband}$ is not allowed within 10 frames. If the absolute difference between $IPD_{whole\_wideband}$ in current and previous frame is higher than 1.5π, and the frame counter is lower than 10, then the

current $IPD_{whole\_wideband}$ will be replaced by the one in the previous frame, and frame counter is increased by 1, otherwise, the frame counter is set to zero, and the current $IPD_{whole\_wideband}$ is kept.

Also due to the phase wrap problem, the actual phase distance between π and $-\pi$ is small. A phase region counter reg_num is used to indicate how often the phase is near to the π or $-\pi$. reg_num is limited in [0, 70], if reg_num is lower than 0 or higher than 70, it will be set to 0 or 70 respectively. If the absolute value of $IPD_{whole\_wideband}$ is higher than 2.5, reg_num is increased by 1, otherwise, reg_num is decreased by 1.

reg_num is also smoothed frame by frame by using the following equation.

$$reg\_num\_sm^{(1)} = SM_1 * reg\_num\_sm^{(-1)} + (1 - SM_1) * reg\_num, \quad \text{(D.6-29)}$$

where SM_1=0.984375 is the smoothing factor

In order to measure the change of $IPD_{whole\_wideband}$ during the past 10 frames, the mean of $IPD_{whole\_wideband}$, $phase\_mean$, over the last 10 frames is calculated. The absolute distance $phase\_dis$ between the mean $phase\_mean$ and the $IPD_{whole\_wideband}$ in current frame is computed. The distant is also smoothed between two frames,

$$phase\_dis\_sm^{(1)} = SM_1 * phase\_dis\_sm^{(-1)} + (1 - SM_1) * phase\_dis, \quad \text{(D.6-30)}$$

If the smoothed distance $phase\_dis\_sm^{(1)}$ higher than 0.08, and $reg\_num\_sm^{(1)}$ lower than 50, the $IPD_{whole\_wideband}$ will be set to zero, otherwise there is no change to the $IPD_{whole\_wideband}$.

The whole wideband IPD estimation is shown in Figure D.6-7.

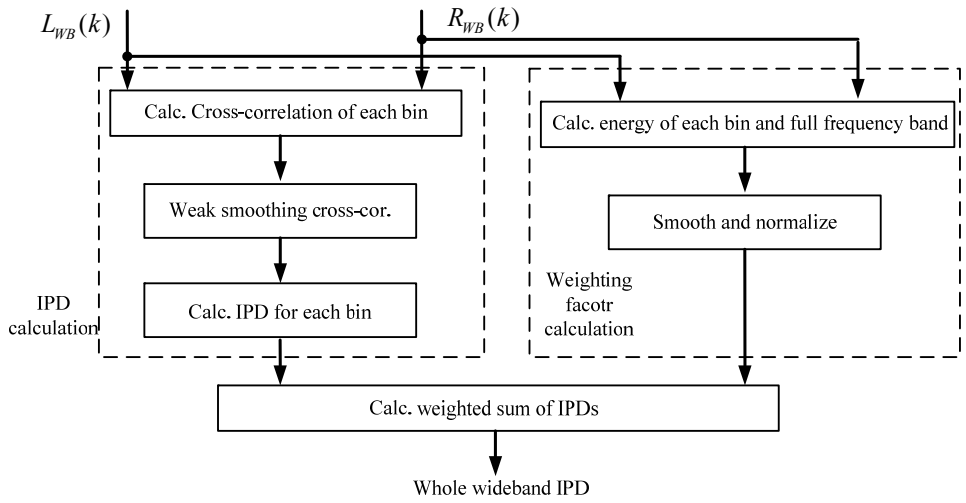


Figure D.6-7 – Flow chart of whole wideband IPD estimation algorithm

D.6.2.3.5 Whole wideband inter-channel coherence estimation

The whole wideband inter-channel coherence is estimated in the following way as illustrated in Figure D.6-8.

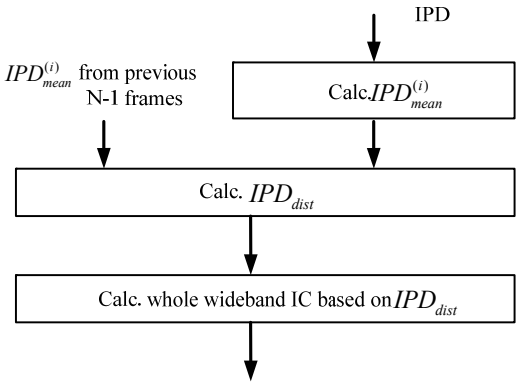


Figure D.6-8 – Flow chart of the whole wideband IC estimation algorithm

The inter-channel phase differences $IPD(k)$ (k=1,…,10) are calculated as defined in clause D.6.2.3.3 using the weak smoothing.

1. The averaged IPD ($IPD_{mean}^{(i)}$) over the frequency bins in the range [1, 10] is also computed as defined in the following equation:

$$IPD_{mean}^{(i)} = \frac{1}{9}\sum_{k=1}^{10} IPD(k), \tag{D.6-31}$$

2. Then, a long term average of the $IPD_{mean}^{(i)}$ is computed as the average over the last 10 frames as:

$$IPD_{meanLT} = \frac{1}{10}\sum_{m=0}^{9} IPD_{mean}^{(i-m)}, \tag{D.6-32}$$

3. The whole wideband IC is then estimated based on the $IPD_{mean}^{(i)}$ and IPD_{meanLT}. In order to evaluate the stability of the IPD parameter, the distance IPD_{dist} between $IPD_{mean}^{(i)}$ and IPD_{meanLT} is computed. This distance shows the evolution of the IPD over the last 10 frames. It is defined as the absolute value of the difference between the local (current frame) mean $IPD_{mean}^{(i)}$ and the long term average IPD_{meanLT}

$$IPD_{dist} = \left| IPD_{mean}^{(i)} - IPD_{meanLT} \right|, \tag{D.6-33}$$

   If the $IPD_{mean}^{(i)}$ parameter is stable over the last frames, the distance IPD_{dist} is close to 0. The distance is then equal to zero when the phase difference is stable over the time. This distance represents an estimation of the similarity of the channels.

4. In order to increase the stability of the parameter estimation, IPD_{dist} is smoothed over two consecutive frames as.

$$IPD_{dist\_sm}^{(i)} = \begin{cases} IPD_{dist}^{(i)} & \text{if } IPD_{dist}^{(i-k)} < 0.003 \text{ for all } k = -4,...,0 \\ W_{dist} \cdot IPD_{dist\_sm}^{(i-1)} + (1 - W_{dist}) \cdot IPD_{dist}^{(i)} & \text{otherwise} \end{cases} \tag{D.6-34}$$

   where W_{dist} is the smoothing factor set to 0.9922.

5. Finally, the whole wideband IC is calculated based on $IPD_{dist\_sm}^{(i)}$, since it has an indirect inverse relation with IC. IC is close to 1 when the channels are similar and $IPD_{dist\_sm}^{(i)}$ becomes equal to 0 in that case. If $IPD_{dist\_sm}^{(i)} < 0.2$ and if ITD and IPD are equal to 0, IC is

set equal to 1. In other cases, the mapping table between $IPD_{dist\_sm}^{(i)}$ and the IC index is given in Table D.6-1, the quantized IC values being given in Table D.6-2.

Table D.6-1 – mapping table between $IPD_{dist\_sm}^{(i)}$ and IC index

	$IPD_{dist\_sm}^{(i)}$ **region**	***IC index***
0	$0.67 \le IPD_{dist\_sm}^{(i)}$	0
1	$0.52 \le IPD_{dist\_sm}^{(i)} < 0.67$	1
2	$0.36 \le IPD_{dist\_sm}^{(i)} < 0.52$	2
3	$0.2 \le IPD_{dist\_sm}^{(i)} < 0.36$	3

Table D.6-2 – Quantized IC value

IC index	***IC***
0	0
1	0.4
2	0.7
3	0.9

D.6.2.3.6 Inter-channel cue selection

So far, the whole wideband ITD, IPD and IC are estimated and quantized. However, due to the bit budget limitation, the whole wideband ITD, IPD and IC cannot be all transmitted at the same time. The quantized value of the most perceptually important parameter is included in the bitstream together with ITD/IPD flags instructing the decoder to de-quantize the transmitted parameter and set the other parameters to the default value (the default values for each parameter are ITD = 0, IPD = 0 and IC = 1). Hence, the bitstream structure indicates to the decoder the value which must be used for all wideband inter-channel cues (ITD, IPD and IC). The selection of the most important parameter is based on the characteristics of the signal.

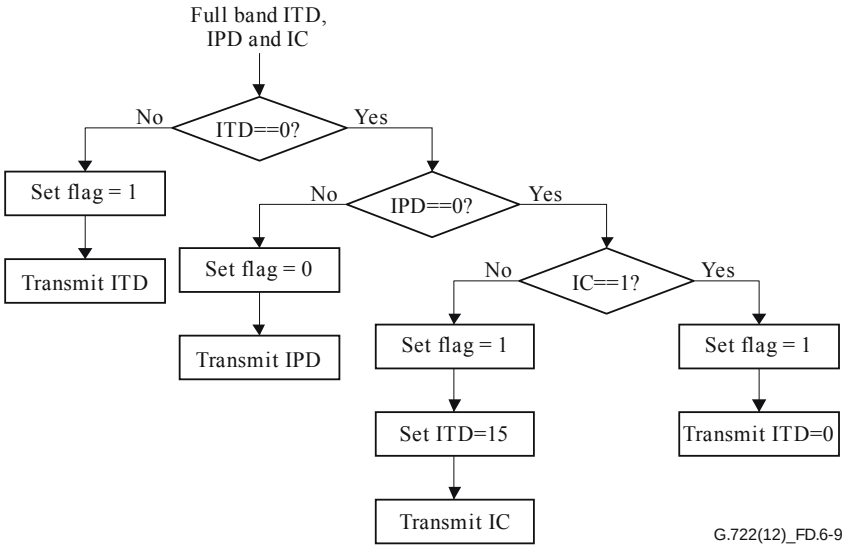


Figure D.6-9 – Flow chart of inter-channel cue selection

1. Check if ITD is not equal to zero.

   1.1 If No, set ITD/IPD flag to 1 indicating that the quantized ITD is transmitted with 4 bits. Five bits (1 bit for ITD/IPD flag and 4 bits for the quantized ITD index) are written into the bitstream.

   1.2 If Yes, check if IPD is equal to zero.

      1.2.1 If No, set ITD/IPD flag to 0 indicating that the quantized IPD is transmitted with 4 bits. Five bits (1 bit for ITD/IPD flag and 4 bits for the quantized IPD index) are written into the bit steam.

      1.2.2 If Yes, check if IC is equal to one.

         1.2.2.1 If No, set ITD/IPD flag to 1, set the following four bits to 15 indicating that the next bits represent IC information (IC flag), and finally the last 2 bits represent the quantized IC. In that case, seven bits (1 bit for ITD/IPD flag, 4 bits for IC flag and 2 bits for the quantized IC index) are written into the bitstream.

         1.2.2.2 If Yes, set ITD/IPD flag to 1, the quantized ITD=0 is transmitted with 4 bits. Only five bits (1 bit for ITD/IPD flag and 4 bits for quantized ITD) are written into the bitstream.

The whole wideband ITD and IPD are all uniform quantized in 4 bits. The whole wideband ITD is first added to 7 before being quantized. The region of the scalar quantizer is 0 to 14. The value 15 is reserved to indicate that the IC is transmitted. The region of the 4 bits uniform scalar quantizer for the whole wideband IPD is from -7*π/8 to π.

D.6.2.3.7 Bit allocation of inter-channel cue

The bitstream structures which are used to indicate the selection information are shown in Figure D.6-10.

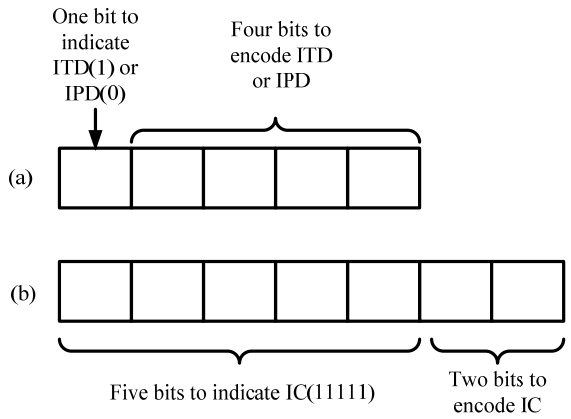


Figure D.6-10 – Bitstream structures

In Figure D.6-10 (a), one bit is used to indicate the following 4 bits are ITD information or IPD information. In Figure D.6-10 (b), 5 bits (which are all 1) indicate that the following two bits are IC information.

D.6.2.4 Inter-channel level differences

The 80 FFT coefficients in the 0~8000 Hz frequency range are split into 20 sub-bands. Table D.6-3 defines the sub-band boundaries. The b-th sub-band comprises $N_{wbcf}(b)$ coefficients with $bands(b) \le k < bands(b+1)$.

Table D.6-3 – Sub-band boundaries and number of coefficients per sub-band

b	***bands(b)***	$N_{wbcf}(b)$
0	0	1
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	2
8	9	2
9	11	2
10	13	3
11	16	3
12	19	4
13	23	4
14	27	4
15	31	6
16	37	7
17	44	8
18	52	9
19	61	20
20	81	Not applicable

Inter-channel level differences (ILDs) reflect the energy relationship between the left and right channels and ILD is calculated for each sub-band. ILDs are defined as the logarithmic energy ratio per sub-band in the frequency domain as.

$$ILD(b) = 10\log_{10}\frac{\sum_{k=bands(b)}^{bands(b+1)-1} L_{WB}(k)L_{WB}^{*}(k)}{\sum_{k=bands(b)}^{bands(b+1)-1} R_{WB}(k)R_{WB}^{*}(k)}, \qquad b = 0,...,19, \qquad \text{(D.6-35)}$$

where *bands*(*b*) is given in Table D.6-3.

D.6.2.4.1 Stereo attack detection in WB

The WB sub-bands are grouped into two groups. The low frequency part starts from sub-band 0, and ends at sub-band 13, the high frequency part starts at sub-band 14 and ends at sub-band 19. The sum of the ILDs of the current frame *i* in each frequency part is calculated as follows:

$$ILD_{WBlow}^{(i)} = \sum_{b=0}^{13} ILD_{WBsub}(b), \qquad \text{(D.6-36)}$$

$$ILD_{WBhigh}^{(i)} = \sum_{b=14}^{19} ILD_{WBsub}(b)$$

Then, long term averages of the ILD sums ($ILD_{WBlowLT}$ and $ILD_{WBhighLT}$) are calculated as the average of the ILD sums over the last *seven* frames:

$$\begin{aligned} ILD_{WBlowLT} &= \frac{1}{N_{att}} \sum_{m=0}^{6} ILD_{WBlow}^{(i-m)}, \\ ILD_{WBhighLT} &= \frac{1}{N_{att}} \sum_{m=0}^{6} ILD_{WBhigh}^{(i-m)} \end{aligned} \quad \text{(D.6-37)}$$

with N_{att} = 7. In order to evaluate the stability of the ILD parameter, the distances, ΔILD_{WBlow} and ΔILD_{WBhigh}, between the ILD sums of the current frame, $ILD_{WBlow}^{(i)}$ and $ILD_{WBhigh}^{(i)}$, and their long term average, $ILD_{WBlowLT}$ and $ILD_{WBhighLT}$, for both frequency parts are calculated. Those distances give the evolution of the ILD during the last seven frames. They are defined as the absolute value of the difference between the current frame and the long term average:

$$\begin{aligned} \Delta ILD_{WBlow} &= \left| ILD_{WBlow}^{(i)} - ILD_{WBlowLT} \right|, \\ \Delta ILD_{WBhigh} &= \left| ILD_{WBhigh}^{(i)} - ILD_{WBhighLT} \right| \end{aligned} \quad \text{(D.6-38)}$$

ΔILD_{WBlow} and ΔILD_{WBhigh} are compared with the thresholds $ILD\_thr_{low} = 30$ and $ILD\_thr_{high} = 11$ respectively.

If one of the distances is higher than its threshold, the input signal is classified as WB transient stereo, otherwise the signal is classified as normal stereo (non-transient). The classification information is written in the bitstream as 1 bit. The ILD classification (transient and non-transient modes) is used as a classification of the WB input signal. Different ILD quantization methods are used depending on this classification.

D.6.2.4.2 Inter-channel level differences quantization

20 ILDs, one per sub-band are calculated. However, due to the bit budget limitation, the ILDs of all sub-bands in a frame cannot be transmitted at the same time. Therefore, the 20 sub-band ILDs are split into different categories and only ILDs with the sub-band index which belong to one category are quantized and transmitted in a frame. The number of categories depends on the input signal characteristics (transient stereo or normal stereo). The category selection depends on the frame index.

If the input signal is classified as WB transient stereo, the 20 sub-band ILDs are split in two categories; otherwise the input signal is classified as WB normal stereo and the 20 sub-band ILDs are split in four categories.

The category information index is transmitted: one bit is needed in transient frame (two-frame mode), two bits in normal frames (four-frame mode). The category index is reset to 0 at the first frame when input signal characteristics change between normal and transient stereo, i.e., when the frame mode switches.

All the ILDs in a selected category are quantized with scalar piece-wise uniform scalar quantizers. The first ILD in the selected category is quantized into 5 bits. For the other ILDs in the selected category, differential quantization is used: the difference between an ILD and the previously quantized ILD in the selected category is calculated and quantized. The ILDs which are not in the

selected category are neither quantized nor transmitted; they are simply replaced by their values from the previous frame in the decoder.

Before describing further the two classes, i.e., transient stereo and normal stereo, the splitting of ILDs in categories and the quantization of the ILDs in the selected category; the piece-wise uniform quantization codebooks are given.

Symmetric mid-thread (i.e., with level 0) quantizers are used with different bitrates: 5 bits, 4 bits, 3 bits. The 5-bit scalar quantizer has 31 levels and 5 uniform pieces of steps: 5, 3, 2 (around level 0), 3, and 5. The 4-bit quantizer is a subset of the 5-bit scalar quantizer: it has 15 levels and 3 uniform pieces of steps: 3, 2 (around level 0) and 3. The 3-bit scalar quantizer is a subset of the 4-bit scalar quantizer: it has 7 levels and 3 uniform pieces of steps: 8, 4 (around level 0), and 8. These ILD quantization codebooks are given in the tables below.

Table D.6-4 – 5-bit ILD quantization codebook

i	***Codebook(i)***
0	–50
1	–45
2	–40
3	–35
4	–30
5	–25
6	–22
7	–19
8	–16
9	–13
10	–10
11	–8
12	–6
13	–4
14	–2
15	0
16	2
17	4
18	6
19	8
20	10
21	13
22	16
23	19
24	22
25	25

Table D.6-4 – 5-bit ILD quantization codebook

i	***Codebook(i)***
26	30
27	35
28	40
29	45
30	50

Table D.6-5 – 4-bit ILD quantization codebook

i	***Codebook(i)***
0	–16
1	–13
2	–10
3	–8
4	–6
5	–4
6	–2
7	0
8	2
9	4
10	6
11	8
12	10
13	13
14	16

Table D.6-6 – 3-bit ILD quantization codebook

i	***Codebook(i)***
0	–16
1	–8
2	–4
3	0
4	4
5	8
6	16

D.6.2.4.2.1 Two-frame mode quantization

If the input signal is classified as transient stereo, two-frame mode quantization is selected. The ILDs are divided into two categories, each one containing ten sub-band ILDs. The splitting is performed according to the parity of the sub-band ILD index. The first category contains the ILDs with even sub-band indices, the second category the ILDs with odd sub-band indices. The two sets, $Cat_{tr}(0)$ and $Cat_{tr}(1)$, give the indices of the ILDs for the two categories:

$$\begin{aligned} Cat_{tr}(0) &= \{0,2,4,\ldots,18\} = \{2n\}, \\ Cat_{tr}(1) &= \{1,3,5,\ldots,19\} = \{2n+1\} \end{aligned} \qquad n = 0,\ldots,9, \qquad \text{(D.6-39)}$$

As mentioned previously, one category is selected in a frame and only the ILDs with sub-band indices in the selected category are quantized and transmitted. The selection of the category is made based on the frame index, noted i_{cat}, in the succession of consecutive transient frames: when this frame index i_{cat} is even, the ILDs with indices in $Cat_{tr}(0)$ – even indices – are quantized, when the frame index i_{cat} is odd, the ILDs with indices in $Cat_{tr}(1)$ – odd indices – are quantized. This means that the ILD with even and odd indices are quantized alternately. i_{cat} is set to 0 when the encoder switch from four-frame mode to two-frame mode.

The ILD of the first sub-band in the selected category, namely sub-band 0 or 1, is quantized using a 5-bit scalar quantizer. The quantization codebook is given in Table D.6-4

For the other nine ILDs in the selected category, the differences of ILDs between two adjacent sub-bands in this category are calculated:

$$d_{ILD}^{tr}(n) = ILD(2n + i'_{cat}) - \hat{I}LD(2n - 2 + i'_{cat}), \qquad n = 1,\ldots,9, \qquad \text{(D.6-40)}$$

where $\hat{I}LD(n)$ is the quantized value of $ILD(n)$, and i'_{cat} the parity of i_{cat} :

$$i'_{cat} = i_{cat} \bmod 2 \qquad \text{(D.6-41)}$$

The first five differences, $d_{ILD}^{tr}(n)$, with *n* from 1 to 5, are quantized by a 4-bit scalar quantizer, its quantization codebook is given in Table D.6-5.

The last four differences, $d_{ILD}^{tr}(n)$, with *n* from 6 to 9, are quantized by a 3-bit scalar quantizer, its quantization codebook is given in Table D.6-6.

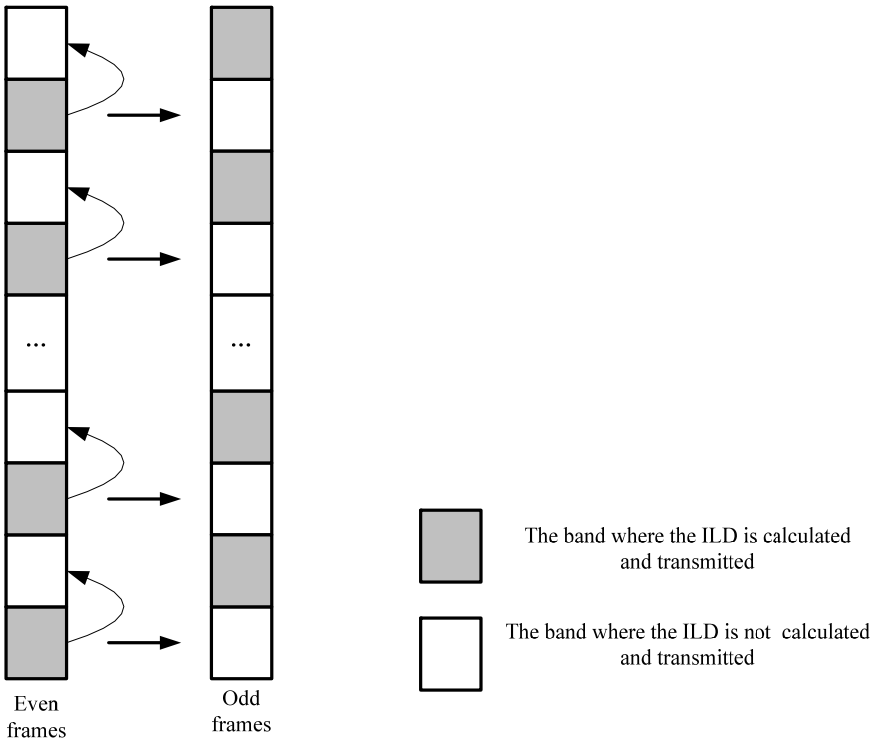


Figure D.6-11 – Two-frame mode

D.6.2.4.2.2 Four-frame mode quantization

If the input signal is classified as normal stereo, four-frame mode quantization is selected. The indices of ILDs are divided into four groups, each with five sub-band ILDs. The indices of the ILDs in these four groups are given in the four sets, $Cat_{norm}(i'_{cat})$, $i'_{cat} = 0,...,3$, given below:

$$\begin{aligned} Cat_{norm}(0) &= \{0,2,4,11,13\}, \\ Cat_{norm}(1) &= \{5,7,9,16,18\}, \\ Cat_{norm}(2) &= \{1,3,10,12,14\}, \\ Cat_{norm}(3) &= \{6,8,15,17,19\} \end{aligned} \tag{D.6-42}$$

As mentioned previously, one group is selected and only the ILDs with indices in the selected group are quantized and transmitted. The selection of the group is made according to the frame index, noted i_{cat}, in the succession of consecutive normal frames: $Cat_{norn}(i'_{cat})$ is selected when $i'_{cat} = i_{cat} \bmod 4$. i_{cat} is set to 0 when the encoder switch from two-frame mode to four-frame mode. Then i_{cat} is incremented for each new four-frame mode frame.

The splitting of 20 sub-bands in these four groups is performed as follows:

First, the 20 sub-bands are divided into 4 groups of 5 consecutive sub-bands. The first and the third are the odd group, the second and the fourth are the even group: the first group includes sub-bands 0 to 4; the second group includes sub-bands 5 to 9, the third group includes sub-bands 10 to 14; the fourth group includes sub-bands 15 to 19.

The indices within each group are then divided into two categories, even and odd. Within each group, even index and odd index ILDs are transmitted separately. For frames with even index i_{cat} (i'_{cat} = 0 or 2), ILDs are taken in even groups (groups 0 and 2), whereas for frames with odd index

i_{cat} (i'_{cat}= 1 or 3), ILDs are taken in odd groups (groups 1 and 3). The selection of indices in an even group, respectively odd group, is as follows:

– For frame index i_{cat} with i'_{cat} even, the indices in the selected category are the even indices of group index i'_{cat} and the odd indices of the other even group;

– For frame index i_{cat} with i'_{cat} odd, the indices in the selected category are the odd indices of group index i'_{cat} and the even indices of the other odd group.

The transmission of four consecutive normal frames is illustrated in the figure below:

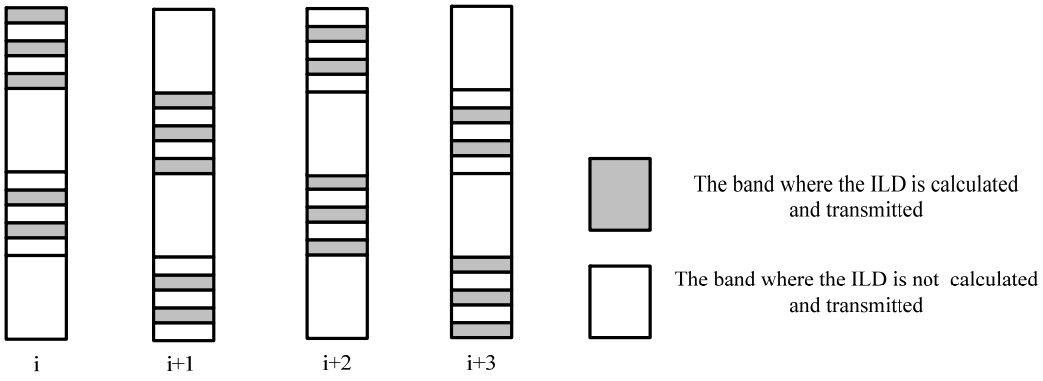


Figure D.6-12 – Four-frame mode

As already mentioned, one category is selected and only the ILDs with indices in the selected category are quantized and transmitted (as given in grey sub-bands in Figure D.6-12).

The bit allocation for the quantization of the five ILDs with indices in the selected category set $Cat_{norm}(0)$ and $Cat_{norm}(1)$ is: 5,4,4,5,4; and the bit allocation for the category set $Cat_{norm}(2)$ and $Cat_{norm}(3)$ is: 5,4,5,4,4. Thus, two sub-band ILDs are quantized with five bits: the indices of these two sub-bands are the first and the fourth indices when i'_{cat} is equal to 0 or 1 and the first and the third indices when i'_{cat} is equal to 2 or 3. When five bits are allocated to quantize a sub-band ILD, it is directly quantized using the 5-bit scalar quantizer. The quantization codebook is given in Table D.6-4. For the other three ILDs in the selected category, differential quantization in 4 bits is used. The differences of ILDs between two adjacent sub-bands in the selected category are calculated:

$$d_{ILD}^{norm}(n) = ILD(ind^{i'_{cat}}{}_{norm}(n)) - \hat{I}LD(ind^{i'_{cat}}{}_{norm}(n-1)), \qquad n = 1, 2+\lfloor i'_{cat}/2 \rfloor, 4, \quad \text{(D.6-43)}$$

where $\hat{I}LD(n)$ is the locally de-quantized value of $ILD(n)$, and $ind^{i'_{cat}}{}_{norm}(n)$ is the n^{th} index in the selected category set, $Cat_{norm}(i'_{cat})$.

Then the differences are quantized with the 4-bits scalar quantization codebook given in Table D.6-5.

D.6.2.4.3 Inter-channel level differences refinement

When the input signal is classified as normal stereo, the stereo image does not change very much during two consecutive frames. However, in some cases ILDs of a few sub-bands change faster than normal. Therefore, *m* bits are used to improve the performance, and this number of bits *m* depends on the WB frame classification (transient or normal) and WB/SWB coding mode (bandwidth flag). The number of bits used for each configuration is given in row "ILD refinement" of Table D.5-4 and Table D.5-5.

The number of sub-bands in each group whose ILD is not transmitted is either 2, 3 or 5 sub-bands. This number depends on the group index and also on the current ILD category index i'_{cat}. This relation is shown in Table D.6-7.

Table D.6-7 – Number of non-transmitted ILD in each group

	Frame index i'_{cat}			
Group index k_{gr}	**0**	**1**	**2**	**3**
0	2	5	3	5
1	5	2	5	3
2	3	5	2	5
3	5	3	5	2

As explained in the previous sub clause, the 20 sub-band indices are divided into 4 groups with 5 sub-bands in each group. Within each group of the four groups, for all non-transmitted sub-band, the differences between the ILD in the current frame and de-quantized ILD in the previous frames are calculated. The differences are added together, and normalized by the number of the frequency bins of the sub-bands in order to calculate the average distortion for each bin. This average distortion is selected as group distortion.

$$dis(k_{gr}) = \frac{\sum_{b \in B_{gr}(k_{gr}, i'_{cat})} \frac{\left| ILD^{(0)}(b) - \hat{ILD}^{-1}(b) \right|}{N_{wbcf}(b)}}{w_1(k_{gr}, i'_{cat})}, \tag{D.6-44}$$

where k_{gr} is the group index, $B_{gr}(k_{gr}, i'_{cat})$ is the set of indices of non-transmitted sub-band ILDs within group k_{gr} for the category index i'_{cat}, $N_{wbcf}(b)$ is the number of the frequency bins in sub-band b, which is given in D.6.2.4, $w_1(k_{gr}, i'_{cat})$ is the cardinal of $B_{gr}(k_{gr}, i'_{cat})$.

The group with the biggest distortion is selected to be enhanced:

$$k_{grmax} = \arg\max_{k}(dist(k)) \tag{D.6-45}$$

Two bits are used to code the index k_{grmax}, which are written in the bitstream together with the ILD refinement bits.

The coding of the ILDs within the selected group (group of index k_{grmax}) is improved by quantization of some differences of its non-transmitted ILDs. The number of quantized ILD differences and the bit allocation depend on k_{grmax} and i'_{cat}.

The improvement is performed from the low frequencies to the high frequencies. If the bit budget is not enough to quantize all the differences, the differences in the low frequency sub-bands are first quantized. The bit allocation is given in Table D.6-8.

Table D.6-8 – ILD refinement bit allocation configuration

$w_1(k_{gr}, i'_{cat})$	WB stereo		SWB stereo	
	Whole wideband ITD/IPD	**Whole wideband IC**	**Whole wideband ITD/IPD**	**Whole wideband IC**
2	4,3	3,2	3,3	2,2
3	3,2,2	3,2	2,2,2	2,2
5	3,2,2	3,2	2,2,2	2,2

The scalar quantizers used for ILD refinement are the same as the ones given in Tables D.6-5 and D.6-6 for 4 and 3 bits respectively. For 2-bit scalar quantizer, the codebook is given in the Table D.6-9.

Table D.6-9 – 2-bit ILD refinement quantization codebook

i	***Codebook(i)***
0	–4
1	0
2	1
3	4

D.6.2.5 Inter-channel phase differences

The inter-channel phase differences (IPDs) are calculated per bin based on the cross-spectrum $c(k)$ (see sub-clause D.6.2.3.2):

$$IPD(k) = \angle c(k)\,, \tag{D.6-46}$$

The range and number of WB inter-channel phase differences calculated depends on the stereo bandwidth:

– for WB stereo, there are 71 IPDs calculated from bin 0 to 70;

– for SWB stereo: there are 81 IPDs calculated from bin 0 to 80;

The number of quantized and transmitted IPDs depends on the stereo layer:

– for WB stereo, eight IPDs are quantized (the IPDs from bin 2 to 9);

– for SWB stereo, seven IPDs are quantized (the IPDs from bin 2 to bin 8) in SL1, the next 16 IPDs (the IPDs of bin 9 to 24) being quantized in SL2.

All the transmitted IPDs are quantized using a 5-bit uniform scalar quantizer, except in SHB two-frame mode, where IPD(8) is quantized using a 4-bit uniform scalar quantizer. The bit "stolen" from the quantization is used to indicate the frame index.

The region of the 5-bit uniform scalar quantizer is from $-15\pi/16$ to π. The region of the 4-bit uniform scalar quantizer is from $-7\pi/8$ to π.

D.6.2.6 Down-mix computation

Since down-mixing in time domain does not finely control the phase differences between channels, and does not preserve the energy per frequency regions. Down-mixing is performed in frequency domain.

The left and right stereo channels $L_{WB}(k)$ and $R_{WB}(k)$ are down-mixed to mono signal $M_{WB}(k)$ using the following equation:

$$M_{WB}(k) = |M_{WB}(k)| e^{j\angle M_{WB}(k)}, \qquad k = 0,...,80, \qquad \text{(D.6-47)}$$

where $|M_{WB}(k)|$ and $\angle M_{WB}(k)$ are the amplitude and the phase of $M_{WB}(k)$ in each frequency bin respectively.

The phase $\angle M_{WB}(k)$ is calculated for each bin in the following function based on the locally de-quantized whole wideband stereo parameter $\hat{IPD}_{whole\_wideband}$ the locally de-quantized sub-band $\hat{ILD}$ and the individual $IPD(k)$ as follows:

$$\angle M_{WB}(k) = \begin{cases} \angle L_{WB}(k) - \dfrac{1}{1+f(b)}(\hat{IPD}(k) - \hat{IPD}_{whole\_wideband}) & 2 \le k \le k_{IPD} \\ \angle L_{WB}(k) - \dfrac{1}{1+f(b)}(IPD(k) - \hat{IPD}_{whole\_wideband}) & 0 \le k < 2, k_{IPD} < k \le 80 \end{cases}, \qquad \text{(D.6-48)}$$

where b is the sub-band of frequency bin k, $f(b) = 10^{\hat{ILD}(b)/20}$ and k_{IPD} is the index of the last bin whose IPD is quantized. k_{IPD} depends on the bitrate as defined in Table D.6-10

Table D.6-10 – The index of the last IPD quantized bin k_{IPD}

bitrate modes	k_{IPD}
R1ws	1
R2ws	9
R2ss, R3ss and R4ss	8
R5ss	24

When available, locally de-quantized parameters are used in order to synchronize with the decoder.

The IPDs in the region $2 \le k \le k_{IPD}$ are the only locally de-quantized version, and outside this frequency region, i.e., $0 \le k < 2$ and $k_{IPD} \le k < 80$, no IPD information is transmitted to the decoder and un-quantized IPD parameters are used.

$\angle M_{WB}(k)$ is located for all *k* between the phase of left $\angle L_{WB}(k)$ and right channel $\angle R_{WB}(k)$.

The amplitude $|M_{WB}(k)|$ is calculated as the following way:

$$|M_{WB}(k)| = (|L_{WB}(k)| + |R_{WB}(k)|)/2, \qquad k = 0,...,80, \qquad \text{(D.6-49)}$$

The down-mix is computed per frequency bin. The real and imaginary parts of the mono down-mix signal are given by the following equations:

$$\begin{aligned} \mathrm{Re}\{M_{WB}(k)\} &= |M_{WB}(k)| \cos(\angle M_{WB}(k)) \\ \mathrm{Im}\{M_{WB}(k)\} &= |M_{WB}(k)| \sin(\angle M_{WB}(k)) \end{aligned}, \qquad k = 0,...,80, \qquad \text{(D.6-50)}$$

Note that as stated in the previous clause, in WB stereo modes, the frequency range is [0, 80].

The frequency domain down-mixed mono signal $M_{WB}(k)$ is converted back to the time domain by inverse FFT.

D.6.2.7 Inverse FFT and OLA

The frequency domain down-mixed mono signal $M_{WB}(k)$ is converted back to the time domain by inverse FFT and OLA.

For the inverse FFT, the first 80 points in the input buffer are the real part of $X_{WB}(k)$, and the last 80 points in the input buffer are the imaginary part of $X_{WB}(k)$.

The 160-points input data $x(n)$ is used to generate an 80-points complex data.

$$\begin{aligned}
&\mathrm{Re}\{v(0)\} = (x(0) + x(80))/2 \\
&\mathrm{Im}\{v(0)\} = -(x(0) - x(80))/2 \\
&\mathrm{Re}\{v(Idx(i))\} = ((1 - \mathrm{sinw}(i)) \cdot x(i) - \mathrm{cosw}(i) \cdot (x(81+i) + x(161-i)) \\
&\qquad \ldots + (1 + \mathrm{sinw}(i)) \cdot x(80-i))/2 \\
&\mathrm{Im}\{v(Idx(i))\} = -(\mathrm{cosw}(i) \cdot (x(i) - x(80-i)) + (1 - \mathrm{sinw}(i)) \cdot x(81+i) \\
&\qquad \ldots - (1 + \mathrm{sinw}(i)) \cdot x(161-i))/2
\end{aligned} \qquad i = 1,\ldots,79, \qquad \text{(D.6-51)}$$

sinw(i) and cosw(i) are defined in clause D.6.2.2

Then as described in D.6.2.2, a P-point DFT to $v(n)$ for Q times is applied to obtain $w(k)$ followed by the application of a Q-point DFT to $w(k)$ for P times to get $r(k)$.

The output buffer is composed of L=80 (= N/2) samples from the current frame, together with the L samples from the previous frame.

$$\begin{aligned}
&x(2 \cdot i) = \mathrm{Re}\{r(i)\}/80 \\
&x(2 \cdot i + 1) = -\mathrm{Im}\{r(i)\}/80
\end{aligned} \qquad i = 0,\ldots,79, \qquad \text{(D.6-52)}$$

D.6.3 SWB stereo encoder

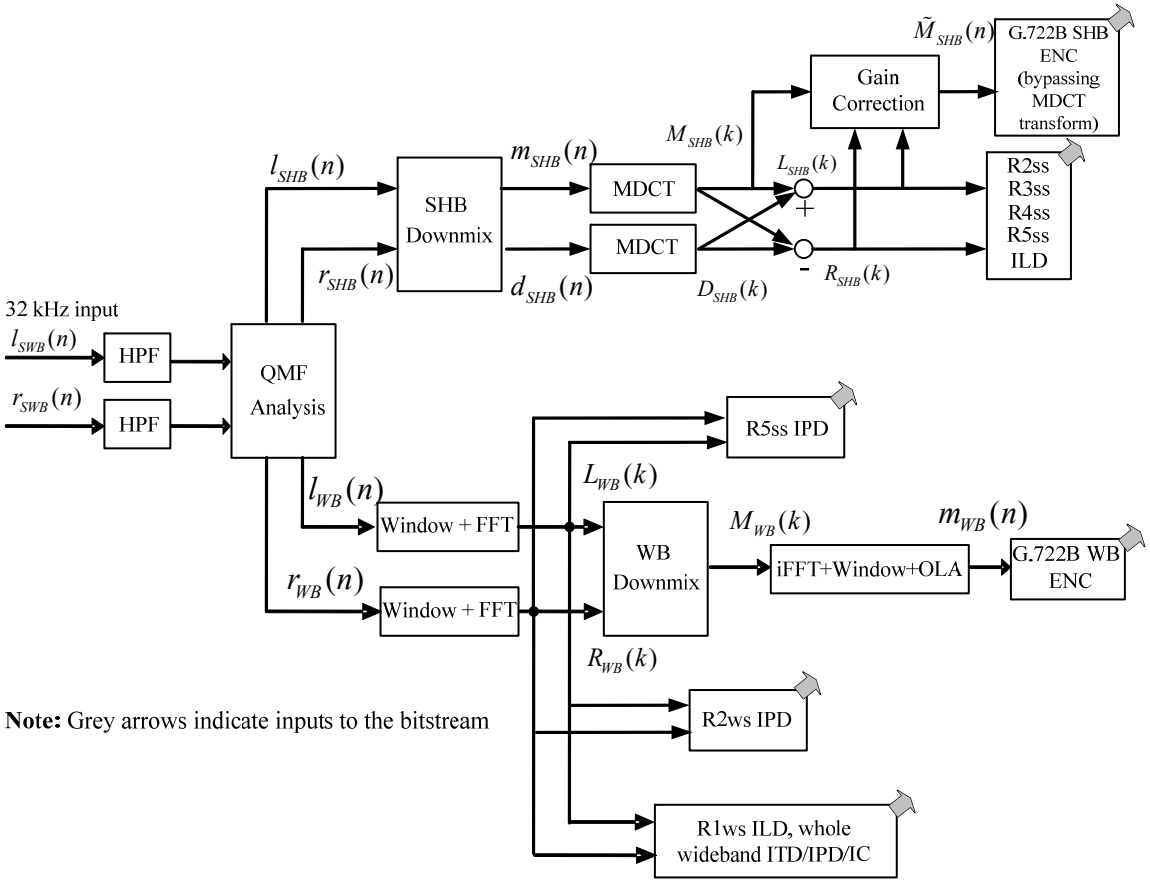


Figure D.6-13 – The high level description of ITU-T G.722 SWB stereo encoder

The WB part encoder is identical to the one described in the clause D.6.2

D.6.3.1 Pre-processing high-pass filter

The pre-processing filter to remove 0-50 Hz components applied to the 32-kHz sampled input signal is defined as:

$$\tilde{s}_{SWB}(n) = 0.9921875 \cdot \tilde{s}_{SWB}(n-1) + s_{SWB}(n) - s_{SWB}(n-1), \qquad n = 0,...,159, \qquad \text{(D.6-53)}$$

where $s_{SWB}(n)$ is either $l_{SWB}(n)$ of $r_{SWB}(n)$ and high-pass filtered output $\tilde{s}_{SWB}(n)$ is either $\tilde{l}_{SWB}(n)$ or $\tilde{r}_{SWB}(n)$.

D.6.3.2 Analysis QMF

An analysis QMF, h^{qmfA}, is applied to the high-pass filtered input signal $\tilde{l}_{SWB}(n)$ and $\tilde{r}_{SWB}(n)$ in order to split it into two 16-kHz-sampled signals; WB signal $l_{WB}(n)$ $r_{WB}(n)$ and SHB signal $l_{SHB}(n)$ and $r_{SHB}(n)$. Note that in the following, for brevity of notation, $x(n)$ is either $l(n)$ or $r(n)$. The 32-kHz-sampled WB signal lower band $x_{SWBL}(n)$ is obtained by filtering the 32 kHz sampled pre-processed signal $\tilde{x}_{SWB}(n)$ through a symmetric FIR low-pass filter with 32 coefficients given by:

$$x_{SWBL}(n) = \sum_{i=0}^{31} h_L^{qmfA}(i)\,\tilde{x}_{SWB}(n-i), \qquad n = 0,\cdots,159, \qquad \text{(D.6-54)}$$

where $h_L^{qmfA}(i)$ are the filter coefficients. The 16 kHz sampled WB signal $x_{WB}(n)$ is then obtained by decimating $x_{SWBL}(n)$ by a factor of 2:

$$x_{WB}(n) = x_{SWBL}(2n+1), \qquad n = 0,\cdots,79, \qquad \text{(D.6-55)}$$

Similarly, the 16-kHz-sampled SHB signal $x_{SHB}(n)$ is obtained by filtering the 32 kHz sampled pre-processed signal $\tilde{x}_{SWB}(n)$ through a FIR high-pass filter with 32 coefficients, then decimating the filtered output signal $x_{SWBH}(n)$ by a factor of 2:

$$x_{SWBH}(n) = \sum_{i=0}^{31} h_H^{qmfA}(i)\,\tilde{x}_{SWB}(n-i), \qquad n = 0,\cdots,159, \qquad \text{(D.6-56)}$$

$$x_{SHB}(n) = x_{SWBH}(2n+1), \qquad n = 0,\cdots,79, \qquad \text{(D.6-57)}$$

where $h_H^{qmfA}(i)$ are the filter coefficients. $x_{SHB}(n)$ is then obtained by decimating $x_{SWBH}(n)$ by a factor of 2.

The high-pass and low-pass filter coefficients, $h_H^{qmfA}(i)$ and $h_L^{qmfA}(i)$, have the following relationship:

$$h_H^{qmfA}(i) = (-1)^i\, h_L^{qmfA}(i), \qquad i = 0,\cdots,31, \qquad \text{(D.6-58)}$$

Therefore, $x_{WB}(n)$ and $x_{SHB}(n)$ can be directly computed as follows:

$$\begin{aligned} x_{WB}(n) &= \sum_{i=0}^{15} h_0^{qmf}(i)\,\tilde{x}_{SWB}(2(n-i)) + \sum_{i=0}^{15} h_1^{qmf}(i)\,\tilde{x}_{SWB}(2(n-i)+1), \\ x_{SHB}(n) &= -\sum_{i=0}^{15} h_0^{qmf}(i)\,\tilde{x}_{SWB}(2(n-i)) + \sum_{i=0}^{15} h_1^{qmf}(i)\,\tilde{x}_{SWB}(2(n-i)+1) \end{aligned} \qquad n = 0,\cdots,79, \qquad \text{(D.6-59)}$$

where

$$\begin{aligned} h_0^{qmf}(i) &= h_H^{qmfA}(2i) \\ h_1^{qmf}(i) &= h_L^{qmfA}(2i+1) \end{aligned}, \qquad i = 0,\cdots,15, \qquad \text{(D.6-60)}$$

Table D.6-11 gives the values of the coefficients h_0^{qmf} and h_1^{qmf}.

Table D.6-11 – QMF coefficients

i	$h_0^{qmf}(i)$	$h_1^{qmf}(i)$
0	0.00064087	–0.00134277
1	–0.00125122	0.00415039
2	0.00143433	–0.0093689
3	–0.00018311	0.0178833
4	–0.00411987	–0.03115845
5	0.01446533	0.05291748
6	–0.03924561	–0.09979248
7	0.128479	0.46646118
8	0.46646118	0.128479
9	–0.09979248	–0.03924561
10	0.05291748	0.01446533
11	–0.03115845	–0.00411987
12	0.0178833	–0.00018311
13	–0.0093689	0.00143433
14	0.00415039	–0.00125122
15	–0.00134277	0.00064087

D.6.3.3 SHB down-mix

The SHB down-mix is performed in the time domain. The left and right channel signals $l_{SHB}(n)$ and $r_{SHB}(n)$ are delayed by one frame to generate the mono down-mix signal $m_{SHB}(n)$, in order to synchronize with the WB down-mix. The mono signal $m_{SHB}(n)$ is simply generated by adding the delayed left and right channel signal together and then dividing by 2:

$$m_{SHB}(n) = 0.5 \cdot \left(l_{SHB}(n) + r_{SHB}(n)\right), \qquad n = 0, \cdots, 79, \qquad \text{(D.6-61)}$$

The difference signal is generated by subtracting the right channel from the left channel and also dividing by 2.

$$d_{SHB}(n) = 0.5 \cdot \left(l_{SHB}(n) - r_{SHB}(n)\right), \qquad n = 0, \cdots, 79, \qquad \text{(D.6-62)}$$

The SHB signals $m_{SHB}(n)$ and $d_{SHB}(n)$ are spectrally folded using equation D.6-63. $x_{SHB}(n)$ is either $m_{SHB}(n)$ or $d_{SHB}(n)$.

$$x_{SHB}^{fold}(n) = (-1)^n x_{SHB}(n), \qquad n = 0, \cdots, 159, \qquad \text{(D.6-63)}$$

D.6.3.4 MDCT

MDCT is applied to the mono $m_{SHB}^{fold}(n)$ and the difference $d_{SHB}^{fold}(n)$ signals. Note that in the following, for brevity of notation, $x_{SHB}^{fold}(n)$ is either $m_{SHB}^{fold}(n)$ or $d_{SHB}^{fold}(n)$, and $X_{SHB}(k)$ corresponds to MDCT coefficients $M_{SHB}(k)$ or $D_{SHB}(k)$. The SHB signal $x_{SHB}^{fold}(n)$ is transformed into frequency domain by the modified discrete cosine transform (MDCT) with a frame length of 5-ms and an

analysis window of 10-ms length. The MDCT coefficients $X_{SHB}(k)$ of the signal $x_{SHB}^{fold}(n)$ are given by:

$$X_{SHB}(k) = \sqrt{\frac{2}{80}} \sum_{n=0}^{159} w_{TDAC}(n) \cos\left(\frac{\pi}{80}(n+40.5)(k+0.5)\right) x_{SHB}^{fold}(n), \qquad k = 0,...,79, \qquad \text{(D.6-64)}$$

where $w_{TDAC}(n)$ is the analysis weighting window given by:

$$w_{TDAC}(n) = \sin\left(\frac{\pi}{160}(n+0.5)\right), \qquad n = 0,...,159, \qquad \text{(D.6-65)}$$

The MDCT coefficients of left and right SHB $L_{SHB}(k)$ and $R_{SHB}(k)$ are directly computed from the MDCT coefficients of $m_{SHB}^{fold}(n)$ and $d_{SHB}^{fold}(n)$ as

$$\begin{aligned} L_{SHB}(k) &= M_{SHB}(k) + D_{SHB}(k) \\ R_{SHB}(k) &= M_{SHB}(k) - D_{SHB}(k) \end{aligned}, \qquad k = 0,...,79, \qquad \text{(D.6-66)}$$

D.6.3.5 SHB inter-channel level differences

In the SHB part, there are two sub-bands. The sub-band boundaries configuration is defined in Table D.6-12. The b-th sub-band comprises $N_{swbcf}(b)$ coefficients with $bands_{SHB}(b) \le k < bands_{SHB}(b+1)$.

Table D.6-12 – Sub-band boundaries and number of coefficients per sub-band in SHB

b	$bands_{SHB}(b)$	$N_{swbcf}(b)$
0	0	20
1	20	40
2	60	Not applicable

For each band b (b=0, 1), the energies of the left channel $E_{SHBsubL}(b)$ and the right channel $E_{SHBsubR}(b)$ are computed.

$$\begin{aligned} E_{SHBsubL}(b) &= \sum_{k=bands_{SHB}(b)}^{k=bands_{SHB}(b+1)-1} L_{SHB}(k)^2, \\ E_{SHBsubR}(b) &= \sum_{k=bands_{SHB}(b)}^{k=bands_{SHB}(b+1)-1} R_{SHB}(k)^2 \end{aligned}, \qquad b = 0,1, \qquad \text{(D.6-67)}$$

Inter-channel level differences (ILDs) are defined as the ratio between the sub-band energies of the left $E_{SHBsubL}(b)$ and right $E_{SHBsubR}(b)$ channels in the log domain. The equation of $ILD_{SHBsub}(b)$ is as follows:

$$ILD_{SHBsub}(b) = 10\log_{10}\frac{E_{SHBsubL}(b)}{E_{SHBsubR}(b)} \qquad b = 0,1, \qquad \text{(D.6-68)}$$

In the SHB part, stereo attack detection is also performed. The sum of the ILD is calculated.

$$ILD_{SHB}^{(i)} = ILD_{SHBsub}(0) + ILD_{SHBsub}(1), \qquad \text{(D.6-69)}$$

Then, a long term average of the ILD sum ILD_{SHBLT} is computed as the average of ILD sums over the last seven frames:

$$ILD_{SHBLT} = \frac{1}{7}\sum_{m=0}^{6} ILD_{SHB}^{(i-m)}, \quad \text{(D.6-70)}$$

In order to evaluate the stability of the ILD parameter, the distance ΔILD_{SHB} between the ILD sums of the current *i*-th frame, $ILD_{SHB}^{(i)}$, and its long term average ILD_{SHBLT} is calculated. This distance shows the evolution of the ILD over the last seven frames. As in the WB encoder, this distance is defined as the absolute value of the difference between the local (current frame) and the long term average:

$$\Delta ILD_{SHB} = \left| ILD_{SHB}^{(i)} - ILD_{SHBLT} \right|, \quad \text{(D.6-71)}$$

This distance ΔILD_{SHB} is compared with the threshold $ILD\_thr_{SHB} = 5$. If the distance is higher than the threshold, the input SHB signal is classified as transient stereo, otherwise the signal is classified as normal stereo. The classification information is written in the bitstream.

Because of the bit budget limitation, the two sub-band ILDs of the SHB cannot be transmitted at the same time. Based on the characteristics of the signal (SHB transient stereo or SHB normal stereo), different quantization methods are used.

If the SHB signal is classified as transient stereo, SHB one-frame mode is selected, only one whole SHB ILD ($ILD_{SHBwhole}^{(i)}$) is calculated and transmitted, which is calculated in the following way:

$$ILD_{SHBwhole}^{(i)} = 10\log_{10}\frac{E_{SHBsubL}(0) + E_{SHBsubL}(1)}{E_{SHBsubR}(0) + E_{SHBsubR}(1)}, \quad \text{(D.6-72)}$$

If the SHB signal is classified as normal stereo, SHB two-frame mode is selected. The two super higher sub-bands ILDs calculated are split in two categories, and only one super higher sub-band ILD is transmitted in a frame.

The selection of the category is made according to the frame index in the succession of consecutive SHB transient frames: when this frame index is even (resp. odd), the category index is 0 (resp. 1) and $ILD_{SHB}(0)$ (resp. $ILD_{SHB}(1)$) is quantized. One bit is used to indicate the category index which is written in the bitstream. The category index is reset to 0 at the first frame when input signal characteristics change between normal and transient (i.e., when the frame mode switches).

Every SHB ILD is quantized by a 5-bit scalar quantizer, the quantization codebook is defined in Table D.6-4.

When switching from one-frame mode to two-frame mode, the ILD in the second SHB sub-band is equal to the whole SHB ILD of previous frame. This is done by:

$$ILD_{SHBsub}(1) = ILD_{SHBwhole}^{(i-1)}, \quad \text{(D.6-73)}$$

D.6.3.6 SHB down-mix gain adjustment

When the left and right channels are not in phase, a simple addition results in coloration of the sound due to the time varying inter-channel signal statistics. To mitigate this problem, the gain of the mono down-mix signal is first adjusted before being sent to the ITU-T G.722 Annex B mono core encoder.

For each sub-band *b* (*b*=0, 1), the energies of the mono signal $E_{SHBsubM}(b)$ is computed.

$$E_{SHBsubM}(b) = \sum_{k=bands_{SHB}(b)}^{k=bands_{SHB}(b+1)-1} M_{SHB}(k)^2, \qquad b = 0, 1, \qquad \text{(D.6-74)}$$

The gain adjustment is performed in the MDCT domain. The gains are calculated in each sub-band of the SHB.

$$g(b) = \frac{E_{SHBsubL}(b) + E_{SHBsubR}(b)}{2 \cdot E_{SHBsubM}(b)}, \qquad b = 0, 1, \qquad \text{(D.6-75)}$$

$g(b)$ is limited in the boundary of [1,2]. If $g(b)$ is higher than 1 or lower than 2, $g(b)$ is limited to 1 and 2 respectively.

The adjustment gain is performed along the frequency bins by piece-wise linear interpolation with three segments. The segment bounds are the four bins: 0, 12, 45 and 60. The gain values for the segment bounds are 1, $g(0)$, $g(1)$ and 1 respectively. The adjustment gain for the frequency bins is linearly interpolated between the gains of these segment bounds. This interpolation is illustrated in the following equation and in Figure D.6-14.

$$g_a(k) = \begin{cases} 1 + \dfrac{g(0)-1}{12} k & 0 \le k < 12 \\ g(0) + \dfrac{g(1)-g(0)}{33}(k-12) & 12 \le k < 45 \\ g(1) + \dfrac{1-g(1)}{15}(k-45) & 45 \le k < 60 \end{cases}, \qquad \text{(D.6-76)}$$

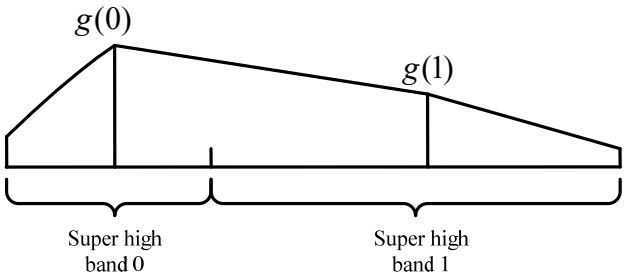


Figure D.6-14 – The linear interpolation of gain

The gain adjustment is performed as:

$$\tilde{M}_{SHB}(k) = g_a(k) \cdot M_{SHB}(k), \qquad k = 0, \ldots, 59, \qquad \text{(D.6-77)}$$

$\tilde{M}_{SHB}(k)$ is directly sent to the ITU-T G.722 annex B SHB encoder, therefore, the MDCT of ITU-T G.722 Annex B SHB encoder is bypassed.

D.7 Functional description of the decoder

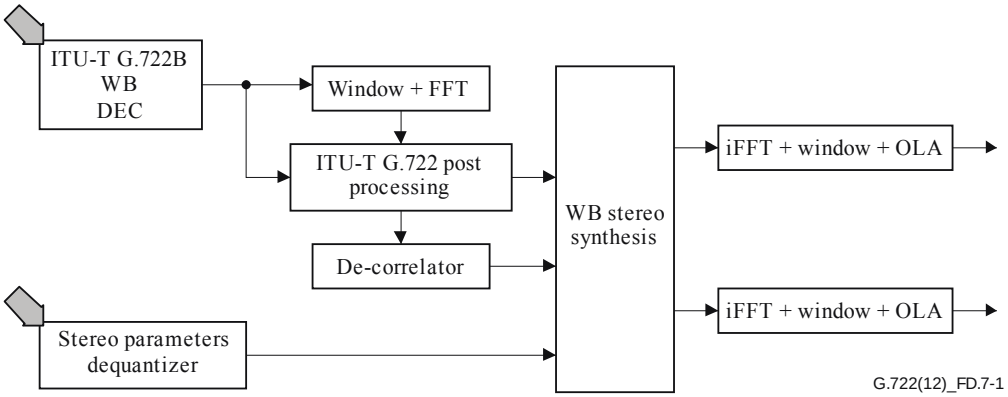
D.7.1 Decoder overview

Figure D.7-1 gives the high-level block diagram of the ITU-T G.722 WB stereo decoder. The whole bitstream is de-multiplexed into two parts: ITU-T G.722 compatible core bitstream and stereo parameters bitstreams (SL0w and SL1w). R1ws mode is based on ITU-T G.722 core at 56 kbit/s WB layer, and synthesizes the stereo signal based on the stereo layer SL0 on top of the enhanced

version of WB signal obtained using G722EL0 as defined in Annex B. R2ws bitrate mode is based on ITU-T G.722 core at 64 kbit/s, and synthesizes the further enhanced stereo signal using the stereo layer SL1w.

Figure D.7-4 shows the high-level block diagram of the ITU-T G.722 SWB stereo decoder. The whole bitstream is de-multiplexed into three parts: ITU-T G.722 compatible core bitstream, ITU-T G.722 Annex B bitstream and stereo parameters bitstream. R2ss bitrate mode is based on ITU-T G.722 core at 56 kbit/s WB layer, and synthesizes the SWB stereo signal using the stereo layers SL0s and SL1s on top of R1sm. R3ss bitrate mode are based on ITU-T G.722 core at 64 kbit/s, and synthesizes the enhanced SWB stereo signal using the stereo layers SL0s and SL1s on top of R2sm. R4ss bitrate mode is based on ITU-T G.722 core at 64 kbit/s, and synthesizes the further enhanced SWB stereo signal using the stereo layers SL0s and SL1s on top of R3sm. Finally, R5ss bitrate mode is based on ITU-T G.722 core at 64 kbit/s, and synthesizes the SWB stereo signal using the stereo layers SL0s, SL1s and SL2 on top of R3sm. Meanwhile, in order to improve the performance of WB signal, post processing of the higher band signal is performed.

D.7.2 WB stereo decoder



NOTE – Grey arrows indicate inputs from the bitstream

Figure D.7-1 – High level description of ITU-T G.722 WB stereo decoder

ITU-T G.722 compatible bitstream is first fed to ITU-T G.722 decoder to produce a time domain mono decoded signal $\hat{m}_{WB}(n)$. On the other hand, the stereo parameters which are WB ILDs, whole wideband ITD, whole wideband IPD, whole wideband IC and the sub-band IPD information, contained in ITU-T G.722 stereo bitstream (layer SL0w for R1ws and layers SL0w and SL1w for R2ws) are decoded. Then $\hat{m}_{WB}(n)$ is converted to the frequency domain coefficients $\hat{M}_{WB}(k)$ using the FFT. In the second step, $\hat{M}_{WB}(k)$ is post processed to generate $\hat{M}_{WB}^{pp}(k)$ and sent to the de-correlator to produce the de-correlated signal. $\hat{M}_{WB}^{pp}(k)$. The de-correlated signal and the decoded stereo parameters are fed to the WB stereo synthesis module to generate the reconstructed left and right channel signals.

D.7.2.1 FFT

The decoded time domain down-mix mono signal $\hat{m}_{WB}(n)$ is transformed to the frequency domain to obtain the mono frequency domain signal $\hat{M}_{WB}(k)$. The time to frequency transform is performed with the 80-point complex FFT described in sub-clause D.6.2.2.

D.7.2.2 ITU-T G.722 higher band post processor

For frames whose mono part is classified as non-TRANSIENT, a FFT domain post-processor is applied to improve the quality of the signal generated by ITU-T G.722 decoder in 4.4 to 8 kHz

frequency range. The post processing gains $g_{pp}^{norm\_sm}(k)$ are calculated in the MDCT domain as described in B.7.2.6.

The post processing gains are directly applied to the FFT domain mono signal.

$$\hat{M}_{WB}^{pp}(k) = \begin{cases} g_{pp}^{norm\_sm}(k) \cdot \hat{M}_{WB}(k), & \text{if } g_{pp}^{norm\_sm}(k) < 1.15 \\ \hat{M}_{WB}(k), & \text{otherwise} \end{cases}, \qquad k = 44,...,79, \qquad \text{(D.7-1)}$$

If the mono signal in the previous frame is TRANSIENT, $g_{pp}^{norm\_sm}(k)$ is set to $g_{pp}^{norm}(k)$ before applying it. For the frames where this post-processor is not applied, $g_{pp}^{norm\_sm}(k)$ is simply set to 1.

D.7.2.3 Stereo synthesis with inter-channel level differences

If WB frame mode bit indicates that the current frame is classified as WB transient stereo, two-frame mode quantization is used for ILD. The category index i'_{cat} is read from the bitstream, the next five bits are read to directly decode $I\hat{L}D(i'_{cat})$ using the codebook defined in Table D.6-4. For the other nine sub-bands, the ILDs are differentially quantized either on four or three bits as explained in sub-clause D.6.2.4.2, and the differences $\hat{diff}(n)$ (n=1,…,9) are decoded and the nine $I\hat{L}D(2n + i'_{cat})$ are recovered using the following equation:

$$I\hat{L}D(2n + i'_{cat}) = I\hat{L}D(2n + i'_{cat} - 2) + \hat{diff}(n), \qquad n = 1,...,9, \qquad \text{(D.7-2)}$$

The non-transmitted ILDs are replaced by the decoded ILDs from the previous frame:

$$I\hat{L}D^{(0)}(2n + 1 - i'_{cat}) = I\hat{L}D^{(-1)}(2n + 1 - i'_{cat}), \qquad n = 1,...,9, \qquad \text{(D.7-3)}$$

For the first frame of two-frame mode after a four-frame mode frame (switch from WB normal stereo to WB transient stereo frame), the non-transmitted ILDs are replaced by the adjacent sub-band ILD:

$$I\hat{L}D(2n+1) = I\hat{L}D(2n), \qquad n = 0,...,9, \qquad \text{(D.7-4)}$$

If WB frame mode bit indicates that the current frame is classified as WB normal stereo, four-frame mode quantization is used for ILD. The category index i'_{cat} is read from the bitstream (2 bits). As explained in sub-clause D.6.2.4.2, the bit allocation is determined from the parity of the category index i'_{cat}. The absolute value of two sub-band ILDs quantized in 5 bits are decoded using the codebook defined in Table D.6-4. The other three sub-band ILDs have been differentially quantized as explained in sub-clause D.6.2.4.2 and the differences $\hat{diff}(n)$ (n=1, 2+(i'_{cat} mod 2), 4) are decoded and the three $I\hat{L}D(i_k)$ (i_k being the n^{th} in $Cat_{norm}(i'_{cat})$) are recovered using the following equation:

$$I\hat{L}D(ind^{i'_{cat}}{}_{norm}(n)) = I\hat{L}D(ind^{i'_{cat}}{}_{norm}(n-1)) + \hat{d}_{ILD}^{norm}(n), \qquad = 1,\ 2+(i'_{cat} \bmod 2),\ 4, \qquad \text{(D.7-5)}$$

The non-transmitted ILDs are replaced by ILDs from the previous frame.

$$I\hat{L}D^{(0)}(n) = I\hat{L}D^{(-1)}(n), \qquad \text{(D.7-6)}$$

where n is the index of the sub-band for which ILDs are not transmitted.

In the four-frame mode, ILD refinement information is also decoded from the bitstream. First, 2-bit are read from the bitstream in order to obtain the index of the refined group. Then the number of bits used for ILD refinement is determined, based on the inter-channel difference selection information and WB/SWB coding mode.

Based on the bit allocation scheme defined in Table D.6-9, the differences of current ILD and ILD in the previous frame $\hat{diff}_{\text{interframe}}$ are de-quantized. Then the decoded differences are added to the corresponding ILDs in current frame to get the final refined ILD.

$$\hat{ILD}^{(0)}(n) = \hat{ILD}^{(-1)}(n) + \hat{diff}_{\text{interframe}}(n), \tag{D.7-7}$$

where n is the index of refined ILD.

The amplitudes of the left $|\hat{L}_{WB}(k)|$ and right $|\hat{R}_{WB}(k)|$ stereo signals are recovered from the decoded and post processed mono down-mix signal $|\hat{M}^{pp}_{WB}(k)|$ by using inter-channel level differences:

$$\begin{aligned} |\hat{L}_{WB}(k)| &= |\hat{M}^{pp}_{WB}(k)| \frac{f(b)}{1+f(b)}, \\ |\hat{R}_{WB}(k)| &= |\hat{M}^{pp}_{WB}(k)| \frac{1}{1+f(b)} \end{aligned} \qquad k = 0,...,79, \tag{D.7-8}$$

where b is the sub-band of frequency bin k and $f(b) = 10^{\hat{ILD}(b)/20}$.

D.7.2.4 Stereo synthesis with inter-channel delay and phase

In the four-frame mode, the whole wideband ITD and whole wideband IPD are decoded from the bitstream. First, the ITD/IPD flag is read from the bitstream. If the ITD/IPD flag is equal to 1, $\hat{ITD}_{whole\_wideband}$ is set to zero and the following four bits are the whole wideband ITD information. If the value for the 4-bit is idx (idx≠15), the whole wideband ITD is calculated in the following way:

$$\hat{ITD}_{whole\_wideband} = idx - 7, \tag{D.7-9}$$

If *idx* is equal to 15, $\hat{ITD}_{whole\_wideband}$ is also set to zero and the next two bits are the IC index, which is decoded using the Table D.6-2.

If ITD/IPD flag is equal to 0, $\hat{ITD}_{whole\_wideband}$ is also set to zero and the following four bits are the whole wideband IPD index. The 4-bits codebook which is described in clause D.6.2.5 is used to decode $\hat{IPD}_{whole\_wideband}$.

In the four-frame mode, the phases can be synthesized by modifying the left and right channel spectra. Based on the decoded whole wideband ITD and whole wideband IPD, the average phase difference between channels are calculated as follows.

$$\bar{IPD}(k) = \frac{-2\pi k \hat{ITD}_{whole\_wideband}}{160} + \hat{IPD}_{whole\_wideband}, \qquad k_{IPD} < k \le 43, \tag{D.7-10}$$

For the frequency bins in the region $k_{IPD} < k \le 43$, the phases of left and right channels $\angle\hat{L}_{WB}(k)$ and $\angle\hat{R}_{WB}(k)$ are recovered from the phase of the post processed mono signal based on the average of the phase difference:

$$\begin{aligned} \angle\hat{L}_{WB}(k) &= \angle\hat{M}^{pp}_{WB}(k) + \frac{f(b)}{1+f(b)} \bar{IPD}(k), \\ \angle\hat{R}_{WB}(k) &= \angle\hat{M}^{pp}_{WB}(k) - \frac{1}{1+f(b)} \bar{IPD}(k) \end{aligned} \qquad k_{IPD} < k \le 43, \tag{D.7-11}$$

k_{IPD} is the index of the last bin whose IPD is quantized, as described in clause D.6.2.6.

If only SL0 is received (R1ws mode), the frequency bins outside this region (i.e., for the bins $k < 2, k > 43$), the phases of left and right channels are set to the phase of the post processed mono signal:

$$\begin{aligned} \angle \hat{L}_{WB}(k) &= \angle \hat{M}^{pp}_{WB}(k), \\ \angle \hat{R}_{WB}(k) &= \angle \hat{M}^{pp}_{WB}(k) \end{aligned} \qquad k < 2, k > 43, \qquad \text{(D.7-12)}$$

Otherwise, if more layers (for modes other than R1ws) are received, the IPD of some bins between left and right channels are transmitted and decoded to reconstruct the phase of left and right channel signals more precisely as described in the next sub-clause.

D.7.2.5 Stereo synthesis with channel phase differences

For modes other than R1ws, the IPD of some bins between left and right channels are transmitted. There are (k_{IPD}-1) transmitted frequency bin IPDs between left and right channels, where k_{IPD} is the index of the last quantized bin, defined in clause D.6.2.6. The decoded $\hat{IPD}(k)$ are used to improve the reconstruction of the phase of left and right channel signals. The difference between decoded $\hat{IPD}(k)$ and the whole wideband $\hat{IPD}_{whole\_wideband}$ is calculated first in order to be in line with the down-mix:

$$IPD_{diff}(k) = \hat{IPD}(k) - \hat{IPD}_{whole\_wideband}, \qquad 2 \le k \le k_{IPD}, \qquad \text{(D.7-13)}$$

Then the phases of left and right channel, in which IPDs are transmitted, are calculated as follows:

$$\begin{aligned} \angle \hat{L}_{WB}(k) &= \angle \hat{M}^{pp}_{WB}(k) + \frac{f(b)}{1+f(b)} \cdot IPD_{diff}(k), \\ \angle \hat{R}_{WB}(k) &= \angle \hat{L}_{WB}(k) - \hat{IPD}(k) \end{aligned} \qquad 2 \le k \le k_{IPD}, \qquad \text{(D.7-14)}$$

The frequency bins outside the region of transmitted *IPD* (i.e., for the bins $k < 2, k > 43$), the phases of left and right channels are set to the phase of the post processed mono signal:

$$\begin{aligned} \angle \hat{L}_{WB}(k) &= \angle \hat{M}^{pp}_{WB}(k), \\ \angle \hat{R}_{WB}(k) &= \angle \hat{M}^{pp}_{WB}(k) \end{aligned} \qquad k < 2, k > 43, \qquad \text{(D.7-15)}$$

D.7.2.6 Inter-channel coherence synthesis

Figure D.7-2 summarizes the processing of synthesizing left and right channel signals $\hat{l}_{WB}(n)$ and $\hat{r}_{WB}(n)$ from the decoded mono down-mixed signal $\hat{M}^{pp}_{WB}(k)$. The processing of one sub-band is shown in detail in the figure. All other sub-bands are processed similarly.

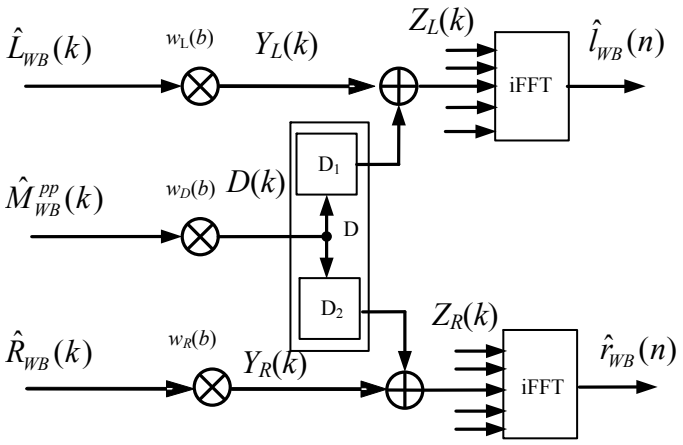


Figure D.7-2 – Inter-channel coherence synthesis

Scale factors $w_L(b)$, $w_R(b)$, and $w_D(b)$ are applied to $\hat{L}_{WB}(k)$, $\hat{R}_{WB}(k)$ and $\hat{M}^{pp}_{WB}(k)$ to generate the frequency coefficients of the left correlated signal $Y_L(k)$, right correlated signal $Y_R(k)$, and left-right un-correlated signal $D(k)$ respectively. A de-correlator D which include D_1 and D_2 are applied to $D(k)$ to generate two de-correlated signals $D_1(k)$ and $D_2(k)$, which are added to $Y_L(k)$and $Y_R(k)$ respectively to generate the stereo output left and right signals $Z_L(k)$, and $Z_R(k)$.

In order to save complexity, D_1 and D_2 just use different delays to generate de-correlation signals. The delays for D_1 and D_2 are 10 ms and 20 ms respectively. Since the delays are a multiple of the 5-ms frame size, the delay is introduced in the frequency domain by just storing the FFT coefficients of $\hat{M}^{pp}_{WB}(k)$ of previous four frames.

$Z_L(k)$, and $Z_R(k)$ are converted back to the time domain by using an inverse FFT to generate the final reconstructed left and right time domain signals $\hat{l}_{WB}(n)$ and $\hat{r}_{WB}(n)$ respectively.

The relative powers of the left and right channels are $P_L(b)$ and $P_R(b)$ are computed from the decoded ILD $\hat{ILD}(b)$ and defined as follows:

$$P_L(b) = \frac{d(b)}{1+d(b)}, \qquad b = 0,....19, \qquad \text{(D.7-16)}$$

$$P_R(b) = \frac{1}{1+d(b)}$$

where $d(b) = 10^{\frac{\hat{ILD}(b)}{10}}$.

Given the whole wideband coherence $IC_{wholewideband}$, the amount of diffuse sound in the left and right channels, $P_D(b)$ can be computed as follows

$$P_D(b) = \frac{P_L(b) + P_R(b) - \sqrt{(P_L(b) + P_R(b))^2 - 4(1 - \hat{IC}^2_{wholewideband})P_L(b)P_R(b)}}{2}, \quad b = 0,....19, \quad \text{(D.7-17)}$$

$P_D(b)$ is clipped between 0 and the minimum of $P_L(b)$ and $P_R(b)$.

The scale factors are computed such that the resulting three signals $Y_L(k)$, $Y_R(k)$, and $D(k)$ have power equal to $P_D(b)$, $P_R(b)$, and $P_D(b)$, i.e.,

$$w_L(b) = 2\sqrt{(P_L(b) - P_D(b)) \cdot g(b)}, \qquad b = 0,....19, \qquad \text{(D.7-18)}$$

$$w_R(b) = 2\sqrt{(P_R(b) - P_D(b)) \cdot g(b)}\,,$$

$$w_D(b) = 2\sqrt{P_D(b) \cdot g(b)}$$

where $g(b) = \dfrac{1 + d(b)}{(1 + f(b))^2}$

The three scaled signals are defined by:

$$\begin{aligned} Y_L(k) &= w_L(b) \cdot \hat{L}_{WB}(k) && b = 0,....19\,, \\ Y_R(k) &= w_R(b) \cdot \hat{R}_{WB}(k)\,, && k = bands(b), \\ D(k) &= w_D(b) \cdot \hat{M}^{pp}_{WB}(k) && ..., bands(b+1) - 1 \end{aligned} \qquad \text{(D.7-19)}$$

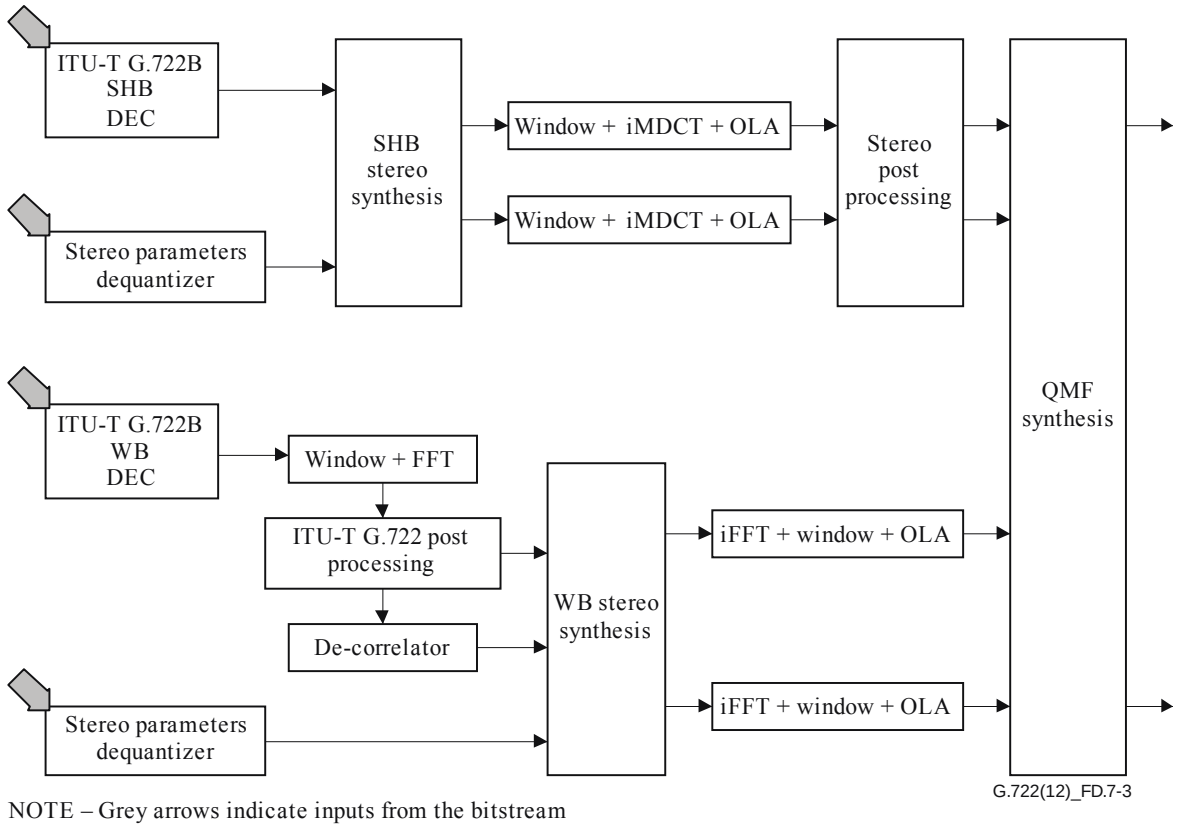
Finally, the decoded stereo channels including the de-correlated signals are given by:

$$\begin{aligned} Z_L(k) &= Y_L(k) + D^{(-2)}(k) \\ Z_R(k) &= Y_R(k) + D^{(-4)}(k) \end{aligned}, \qquad k = 0,...,79\,, \qquad \text{(D.7-20)}$$

D.7.2.7 Inverse FFT and OLA

See clause D.7.2.1.

D.7.3 SWB stereo decoder



NOTE – Grey arrows indicate inputs from the bitstream

Figure D.7-3 – High level description of ITU-T G.722 SWB stereo decoder

The WB decoder is the same as described in D.7.2.

D.7.3.1 Stereo synthesis with SHB inter-channel level differences

The MDCT coefficients of super high band left and right signals $\hat{L}_{SHB}(k)$ and $\hat{R}_{SHB}(k)$ are recovered from the MDCT coefficients of decoded down-mix signal $\hat{M}_{SHB}(k)$ by using decoded SHB inter-channel level differences.

$$\begin{aligned} \hat{L}_{SHB}(k) &= \hat{M}_{SHB}(k)\cdot\frac{f(b)}{1+f(b)}, \\ \hat{R}_{SHB}(k) &= \hat{M}_{SHB}(k)\cdot\frac{1}{1+f(b)} \end{aligned} \qquad \begin{aligned} &b = 0,1, \\ &k = bands_{SHB}(b), \\ &\ldots, bands_{SHB}(b+1)-1 \end{aligned} \qquad \text{(D.7-21)}$$

where $f(b) = 10^{I\hat{L}D_{SHB}(b)/20}$.

D.7.3.2 Inverse MDCT and OLA

Before applying the inverse MDCT, the last 20 SHB frequency coefficients are set to zero to obtain 14 kHz bandwidth output. Then, the SHB frequency coefficients are transformed to the time domain by inverse MDCT transform. Again, for brevity of notation, $x(n)$ is either $l(n)$ or $r(n)$:

$$\hat{x}_{SHB}^{(0)}(n) = \sqrt{2}\sum_{k=0}^{79}\cos\left(\frac{\pi}{80}(k+0.5)(n+40.5)\right)\hat{X}_{SHB}(k), \qquad n = 0,\ldots,159, \qquad \text{(D.7-22)}$$

The SHB signal is obtained by the following OLA operation:

$$\hat{x}'_{SHB}(n) = w_{TDAC}(n+80)\cdot\hat{x}_{SHB}^{(-1)}(n) + w_{TDAC}(n)\cdot\hat{x}_{SHB}^{(0)}(n), \qquad n = 0,\cdots,79, \qquad \text{(D.7-23)}$$

where $w_{TDAC}(n)$ is the synthesis weighting window:

$$w_{TDAC}(n) = \sin\left(\frac{\pi}{160}(n+0.5)\right), \qquad n = 0,\ldots,159, \qquad \text{(D.7-24)}$$

and $\hat{x}_{SHB}^{(-1)}(n)$ is obtained from the previous inverse MDCT transform, which is updated as:

$$\hat{x}_{SHB}^{(-1)}(n) = \hat{x}_{SHB}^{(0)}(80+n), \qquad n = 0,\ldots,79, \qquad \text{(D.7-25)}$$

D.7.3.3 Stereo time envelope post-processor

D.7.3.3.1 Overview of stereo time envelope post-processor

If the down-mix SHB mono signal is classified as transient, some pre-echo artefacts may be heard in the reconstructed stereo signal. To avoid this, post processing is applied to at least one channel in order to improve the quality. As the time envelopes of the two channels are not transmitted, this post processing takes advantage of the time envelope of the decoded mono down-mix signal, weighted by channel weighting factors w_{left} and/or w_{right}. The weighting factors depend on the decoded SHB ILD. If the SHB signal is classified as transient stereo, the post processing is applied to only one channel; otherwise (the SHB signal is classified as normal stereo), the post processing is applied to both channels. In this case, one channel uses the weighted mono time envelop delayed by $I\hat{T}D_{whole\_wideband}$ samples. The post processing is illustrated in the Figure D.7-4.

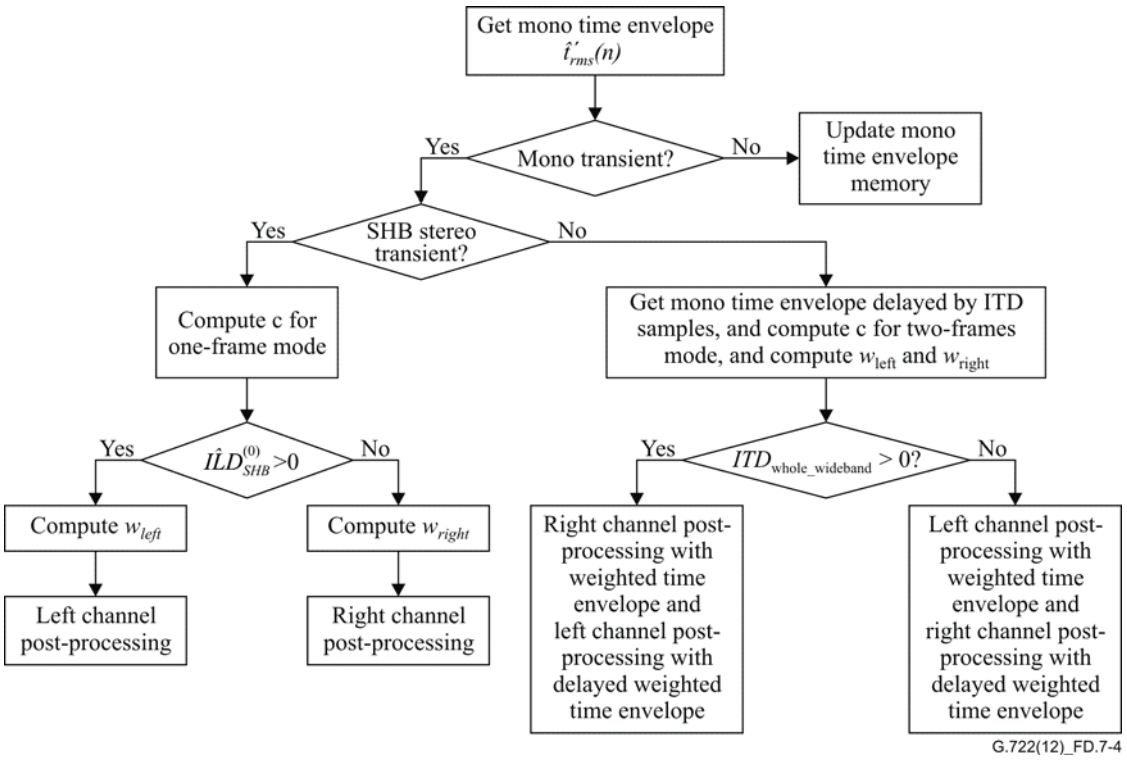


Figure D.7-4 – Flow chart of stereo post processing

First, the decoded time envelope $\hat{t}'_{rms}(n)$ of the decoded mono down-mixed signal is calculated as described in clause B.7.6.

If the mono signal is classified as normal (non-TRANSIENT), the memory of the decoded time envelope $\hat{t}'_{rms}(n)$ is simply updated. Otherwise (the mono signal is classified as transient), the SHB stereo time envelope post processing is applied.

D.7.3.3.2 Stereo time envelope post-processor in one-frame SHB mode

When the SHB stereo signal is classified as stereo transient, one-frame mode is used and only one SHB ILD is received. The weighting parameter f is computed from the decoded $I\hat{L}D^{(0)}_{SHB}$ as: $f = 10^{I\hat{L}D^{(0)}_{SHB}/20}$.

If only one channel is transient, the energy of this channel is higher than the energy of the other channel. Therefore, the energy information is used to identify which channel is transient. If $I\hat{L}D^{(0)}_{SHB}$ is positive (resp. negative), the energy of the left (resp. right) channel is higher than the energy of the right (resp. left) channel, the stereo time envelope post processing is only applied to the left (resp. right) channel using the weighted mono time envelope.

Thus, if $I\hat{L}D^{(0)}_{SHB} > 0$, the stereo time envelope post processing is applied to the left channel, the post processed left signal $\hat{l}^{pp}_{SHB}(n)$ is computed as follows:

$$\hat{l}^{pp}_{SHB}(n) = \hat{l}_{SHB}(n) \cdot \hat{t}'_{rms}(n) \cdot w_{left}, \qquad n = 0, \cdots, 79, \qquad \text{(D.7-26)}$$

where the left weighting factor is calculated from the weighting parameter f as $w_{left} = \frac{2f}{1+f}$.

Otherwise ($\hat{ILD}_{SHB}^{(0)} \leq 0$), the stereo time envelope post processing is applied to the right channel, the post processed right signal $\hat{r}_{SHB}^{pp}(n)$ is computed as follows:

$$\hat{r}_{SHB}^{pp}(n) = \hat{r}_{SHB}(n) \cdot \hat{t}'_{rms}(n) \cdot w_{right}, \qquad n = 0, \cdots, 79, \qquad \text{(D.7-27)}$$

where the right weighting factor is calculated from the weighting parameter f as $w_{right} = \frac{2}{1+f}$.

D.7.3.3.3 Stereo time envelope post-processor in two-frame SHB mode

When the SHB stereo signal is classified as normal stereo, two-frame mode is used and two SHB ILDs are received. The weighting parameter f is computed from the decoded ILD of the first SHB $\hat{ILD}_{SHB}(0)$ as: $f = 10^{\hat{ILD}_{SHB}(0)/20}$.

The post processing is applied to both channels. The inter-channel time difference is given by the decoded $\hat{ITD}_{whole\_wideband}$. If $\hat{ITD}_{whole\_wideband}$ is negative (resp. positive), the left (resp. right) channel comes first, the weighted mono time envelope is delayed by $\hat{ITD}_{whole\_wideband}$ samples before being applied to the right (resp. left) channel; the time envelope of the left (resp. right) channel is directly recovered by using the weighted mono time envelope from the mono down-mixed signal.

Thus, $\hat{ITD}_{whole\_wideband} > 0$

$$\begin{aligned} \hat{l}_{SHB}^{pp}(n) &= \hat{l}_{SHB}(n) \cdot \hat{t}'_{rms}(n) \cdot w_{left}, \\ \hat{r}_{SHB}^{pp}(n) &= \hat{r}_{SHB}(n) \cdot \hat{t}'_{rms}(n - ITD_{whole\_wideband}) \cdot w_{right} \end{aligned} \qquad n = 0, \cdots, 79, \qquad \text{(D.7-28)}$$

otherwise,

$$\begin{aligned} \hat{l}_{SHB}^{pp}(n) &= \hat{l}_{SHB}(n) \cdot \hat{t}'_{rms}(n - ITD_{whole\_wideband}) \cdot w_{left}, \\ \hat{r}_{SHB}^{pp}(n) &= \hat{r}_{SHB}(n) \cdot \hat{t}'_{rms}(n) \cdot w_{right} \end{aligned} \qquad n = 0, \cdots, 79, \qquad \text{(D.7-29)}$$

where the left and right weighting factors are calculated from the weighting parameter f as in the previous sub-clause D.7.3.3.2.

This post processing is also applied if at the previous frame both the mono and SHB stereo signals were classified as transient.

D.7.3.4 Frame erasure concealment

In case of frame erasure, the FERC algorithm described in clause B.7.8 is used to extrapolate missing samples of the down-mixed mono signal.

For the stereo parameters, the decoded ILD from the previous (i-1)-th frame $\hat{ILD}^{(i-1)}(b)$, b=0,..,19, and $\hat{ILD}_{SHB}^{(i-1)}(b)$, b=0,1, are used to recover the amplitude of left and right channel signals.

For WB stereo

$$\begin{aligned} |\hat{L}'_{WB}(k)| &= |\hat{M}'_{WB}(k)| \frac{2f(b)}{1+f(b)}, \\ |\hat{R}'_{WB}(k)| &= |\hat{M}'_{WB}(k)| \frac{2}{1+f(b)} \end{aligned} \qquad \begin{aligned} b &= 0, ..., 19, \\ k &= bands(b), \\ &..., bands(b+1)-1 \end{aligned} \qquad \text{(D.7-30)}$$

where $f(b) = 10^{I\hat{L}D^{(-1)}(b)/20}$,

and for SHB

$$\begin{aligned} \hat{L}'_{SHB}(k) &= \hat{M}'_{SHB}(k)\frac{2f(b)}{1+f(b)}, \\ \hat{R}'_{SHB}(k) &= \hat{M}'_{SHB}(k)\frac{2}{1+f(b)} \end{aligned} \qquad \begin{aligned} b &= 0,1, \\ k &= bands_{SHB}(b), \\ &\ldots, bands_{SHB}(b+1)-1 \end{aligned} \qquad \text{(D.7-31)}$$

where $f(b) = 10^{I\hat{L}D^{(-1)}_{SHB}(b)/20}$.

The phases of left and right channels are set equal to the phase of the mono signal after FERC decoding.

For WB stereo:

$$\begin{aligned} \angle\hat{L}'_{WB}(k) &= \angle\hat{M}'_{WB}(k), \\ \angle\hat{R}'_{WB}(k) &= \angle\hat{M}'_{WB}(k) \end{aligned} \qquad k = 0,\ldots,79, \qquad \text{(D.7-32)}$$

and for SHB stereo:

$$\begin{aligned} \angle\hat{L}'_{SHB}(k) &= \angle\hat{M}'_{SHB}(k), \\ \angle\hat{R}'_{SHB}(k) &= \angle\hat{M}'_{SHB}(k) \end{aligned} \qquad k = 0,\ldots,60, \qquad \text{(D.7-33)}$$

If the number of consecutive lost frames is greater than 10, all the memories of stereo parameters are set to the initial values such that a monaural synthesis is obtained.

D.7.3.5 Bandwidth switching

Same as clause B.7.9 when switching from SWB to WB.

D.7.3.6 Switching between mono and stereo

When switching from stereo to mono, the stereo parameters of previous frame (except sub-band IPDs which are set to zeros) are used as the stereo parameters of current frame, and the current mono frame will be decoded as stereo frame. After 10 frames, all the memories of stereo parameters are set to the initial values such that a monaural synthesis is obtained.

D.8 Bit-exact description of the ITU-T G.722 stereo extension coder

The description of the coding algorithm of this Recommendation is made in terms of bit-exact fixed-point mathematical operations. The ANSI C code indicated in clause 8, which constitutes an integral part of this Recommendation, reflects this bit-exact, fixed-point descriptive approach. The mathematical descriptions of the encoder and decoder can be implemented in other fashions, possibly leading to a codec implementation not complying with this Recommendation. Therefore, the algorithm description of the ANSI C code of clause 8 shall take precedence over the mathematical descriptions whenever discrepancies are found. A non-exhaustive set of test signals, which can be used with the ANSI C code, is available as an electronic attachment to this Recommendation.

D.8.1 Use of the simulation software

The C code consists of two main programs, encoder.c and decoder.c which simulate the main encoder and main decoder, respectively.

The command line for the WB stereo encoder is as follows:

encoder -stereo –wb [-options] <infile> <codefile> [rate]

where

rate	is the desired encoding bitrate in kbit/s: either 64 or 80 (64 for R1ws or 80 for R2ws)
infile	is the name of the input file to be encoded
codefile	is the name of the output bitstream file

Options:

-quiet	quiet processing

The command line for the SWB stereo encoder is as follows:

encoder -stereo [-options] <infile> <codefile> [rate]

rate	is the desired encoding bitrate in kbit/s: (80 for R2ss, 96 for R3ss,112 for R4ss and 128 for R5ss)
infile	is the name of the input file to be encoded
codefile	is the name of the output bitstream file

Options:

-quiet	quiet processing

The command line for the WB and SWB stereo decoder are the same, as follows:

decoder [-options] <codefile> <outfile> [rate]

where

rate	is the desired decoding bitrate in kbit/s: 64 R1ws, 80 for R2ws or R2ss, 96 for R3ss,112 for R4ss and 128 for R5ss
codefile	is the name of the input bitstream file
outfile	is the name of the decoded output file

Options:

-quiet	quiet processing
-bitrateswitch [bsflag]	bsflag is 1 for ITU-T G.722 core at 56kbit/s and 0 for ITU-T G.722 core at 64 kbit/s

The encoder input and the decoder output files are sampled data files containing 16-bit PCM signals. The encoder output and decoder input files follow the ITU-T G.192 bitstream format.

D.8.2 Organization of the simulation software

Table D.8-1 – Summary of stereo encoder specific routines

Filename	Description
encoder.c	ITU-T G.722-stereo encoder interface routine
g722_stereo_enc.c	ITU-T G.722-stereo main encoder
get_interchannel_difference.c	Routines for whole wideband inter-channel differences (IPD, ITD and IC) estimation

Table D.8-2 – Summary of stereo decoder specific routines

Filename	Description
decoder.c	ITU-T G.722-stereo decoder interface routine
g722_stereo_dec.c	ITU-T G.722-stereo main decoder

Table D.8-3 – Summary of common routines

Filename	Description
fft.c	FFT routine
stereo_tools.c	Routines for read/write stereo bitstream and phase calculation
table_stereo.c	Tables for stereo encoder and decoder

Appendix I

Networking aspects

(This appendix does not form an integral part of this Recommendation.)

The purpose of this appendix is to give a broad outline of the interaction of 64 kbit/s (7 kHz) audio-coding with other parts of the digital network. Some general guidance is also offered.

The establishment of the connection is beyond the scope of this Recommendation.

I.1 Network characteristics

This Recommendation is applicable to systems operating in networks which exhibit each of the following characteristics:

i) availability of network octet timing at the terminals;

NOTE – Octet timing may also be derived from control signals within the frame structure defined in Recommendation ITU-T G.725;

ii) plesiochronous networking where the reference clocks meet the timing requirements given in Recommendation ITU-T G.811, or synchronous networking;

iii) 64 kbit/s connection types having either of the following characteristics:

- full 64 kbit/s transparency,
- pulse density restriction as described in Recommendation ITU-T G.802.

NOTE – 64 kbit/s (7 kHz) audio-coding can also operate in networks where there is substitution of a signalling bit for the 8th bit of the octet as described in Recommendation ITU-T G.704, clause 3.1 or where there is 56 kbit/s transparency only. However, a reduction of the audio bit rate and auxiliary data channel capacity occurs and only two modes of operation, denoted 1 *bis* (unframed) and 3 *bis,* are possible as follows:

– Mode 1 *bis* : 56 kbit/s for audio-coding and no data channel;

– Mode 3 *bis*: 48 kbit/s for audio-coding, a 6.4 kbit/s data channel and 1.6 kbit/s for service channel framing and mode control.

I.2 Integration into the telecommunications network

It is foreseen that the 64 kbit/s (7 kHz) audio-coding system will be used for point-to-point, multipoint and broadcast applications. Examples of particular uses are: commentary quality channels for broadcasting purposes and high quality speech for audio and video conferencing applications.

The coding system can operate over any 64 kbit/s bearer channel (see clause I.1), e.g., the public switched telephone network, leased circuits or over an ISDN.

Processes such as digital speech interpolation, echo control and digital pads must be disabled for the transmission of 64 kbit/s (7 kHz) audio-coding. The disabling protocol is not the subject of this Recommendation.

It should be noted however that signal processing may occur in a multipoint conference unit (see clause I.7).

I.3 Audio performance of the 64 kbit/s (7 kHz) audio-coding system

I.3.1 Speech

The speech performance of the 64 kbit/s (7 kHz) audio-coding system has been quantified in terms of Q_{w}-values, where Q_w is a measure of the signal-to-correlated noise ratio of the wideband system, measured in dB. Detailed information on *Q*-value measurements may be found in

Recommendation P.81. This Recommendation, although primarily intended for telephony bandwidth applications, has been used for the evaluation of wideband systems – signified by the subscript W – by use of an appropriate filter (50-7 000 Hz).

For guidance purposes only, a Q_w value of 38 dB corresponds approximately to a 128 kbit/s (7 kHz) PCM system (sampling frequency 16 kHz, coding law as in Recommendation ITU-T G.711), whereas a Q_w value of 45 dB is approximately equivalent to the audio parts of the coder interconnected without the intermediate SB-ADPCM coding process.

Table I.1 indicates the relative performance in Q_w values for nominal input values.

Table I.1 –Relative levels of speech performance (Q_w values)

Mode of operation	Transcodings		
	1	4 Analogue according to Fig. I.1	4 Digital according to Fig. I.2
1 (64 kbit/s)	45	38	41
1 (56 kbit/s)	43	36	38
1 (48 kbit/s)	38	29	34

The performance of the 64 kbit/s (7 kHz) audio-coding system has been found to be substantially unaffected by randomly distributed errors at BER levels as high as $1 \cdot 10^{-4}$. High error ratios approaching $1 \cdot 10^{-3}$ produce perceptible degradation which may be considered tolerable in certain applications.

No particular problems have been experienced in the multiple talker condition and hence correct operation under normal conference conditions can safely be assumed.

The performance under conditions of mode mismatch (i.e., where the variant used in the decoder for a given octet does not correspond to the mode of operation) is considered in clause I.5.

I.3.2 Music

Although primarily designed for speech, no significant distortions may be expected when coding a wide range of music material in Mode 1. Further study on the effects on music signals is a matter of Study Group CMTT.

I.4 Audio performance when interconnected with other coding systems on an analogue basis

I.4.1 64 kbit/s PCM

Informal subjective tests carried out over a path consisting of an analogue interconnected combination of a 64 kbit/s PCM link conforming to Recommendation ITU-T G.711 and a 64 kbit/s (7 kHz) audio-coding link has indicated that no interworking problems will occur. However, the performance of the combination will not be better than that of 64 kbit/s PCM.

Interconnection of the two coding systems on a digital basis is the subject of clause I.8.

I.4.2 32 kbit/s ADPCM

An analogue interconnected combination of a 32 kbit/s ADPCM link conforming to Recommendation ITU-T G.721 and a 64 kbit/s (7 kHz) audio-coding link is not expected to pose any interworking problems. However, the performance of the combination will not be better than that of 32 kbit/s ADPCM.

Interconnection of the two coding systems at a digital level is the subject of further study.

I.5 Audio performance under mode switching

It is recommended that mode switching should be performed synchronously between the transmitter and the receiver to maximize the audio performance. However, asynchronous mode switching may be considered since the condition of mode mismatch will probably be of limited duration and hence the corresponding performance is likely to be acceptable. Although not desirable, operation under permanent mode mismatch may be contemplated in exceptional circumstances. Table I.2 indicates the relative performance under all mode mismatch combinations for nominal input levels.

Table I.2 – Relative speech performance under mode mismatch (Q_W values)

Bit rate used for audio reception	Bit rate used for audio transmission	
	56 kbit/s	48 kbit/s
64 kbit/s	41	35
56 kbit/s	–	36

NOTE – The bits not used for audio-coding have been replaced by bits of a pseudorandom sequence.

I.6 Auxiliary data channel performance

The available combinations of audio and data channel bit rates depends on the connection types described in clause I.1, numeral iii.

The data channel is unaffected by the characteristics of the audio signal since the audio and data channels are effectively decoupled. The transparency of the data channel is limited only by the choice of signalling sequences which could be used to derive the terminal identification. If these sequences are chosen to be of a suitable format, the possibility of their simulation by audio or data bits can be made extremely low. Hence, for all practical purposes, the data channel may be assumed to be transparent.

The control of the data channel capacity is considered in Recommendation ITU-T G.725.

Although the format of the data channel is not part of this Recommendation, it may be noted that the use of two completely independent 8 kbit/s data channels when the total data channel capacity is 16 kbit/s is not prohibited by the algorithm.

Under transmission error conditions the data channel is not subject to error multiplication due to the audio-coding algorithm.

NOTE – It might be possible to obtain additional data channel capacity by substituting data for the two bits normally allocated to the higher sub-band with the consequent penalty of a reduction in the audio bandwidth. However, such an approach is likely to require a more stringent specification for the receive filter characteristics in order to minimize aliasing effects.

I.7 Multi-point conference configuration

The specific features of a multipoint conference arrangement including control of the data channel, echo control, and handling of control messages between terminals, are beyond the scope of this Recommendation. However, the audio-coding algorithm has been chosen to maintain maximum flexibility for multipoint conference arrangements which are likely to emerge. There are a number of general guidelines which should be noted:

– To maximize audio performance, the highest audio bit rate possible, consistent with the transmitted data channel bit rate requirement, should be used for transmission into and out of the signal summing facility of the multipoint conference unit.

NOTE – The signal summation must be carried out on a linear representation of the signals.

– The transmit and receive modes of a terminal or port of a multipoint conference unit do not necessarily have to be the same.

– Signal summing at the sub-band uniform PCM level is preferred for the following reasons:

i) the hardware is minimized in the multipoint conference unit (MCU) by eliminating the need for quadrature mirror filters,

ii) signal quality is maximized and additional signal delay is eliminated by avoiding additional filtering,

iii) echo control is likely to be simpler to perform at the sub-band level.

Figure I.3 indicates a possible arrangement at the multipoint conference bridge with signal summing at the sub-band level;

– For reasons of audio performance, the number of tandem connected multipoint conference units interconnected with 64 kbit/s (7 kHz) audio-coding is limited to three, see Figure I.4).

– In the case where the multipoint conference unit includes 64 kbit/s PCM ports, digital transcoding principles equivalent to that described in clause I.8 should be used to derive the higher and lower sub-band signals.

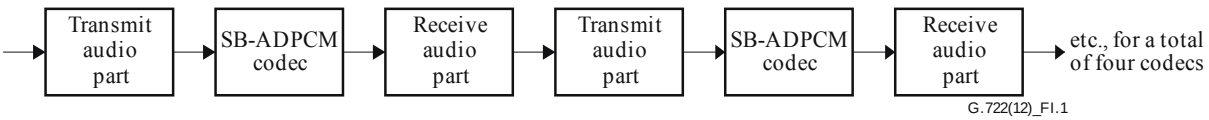


Figure I.1 – Four transcodings (analogue interconnection)

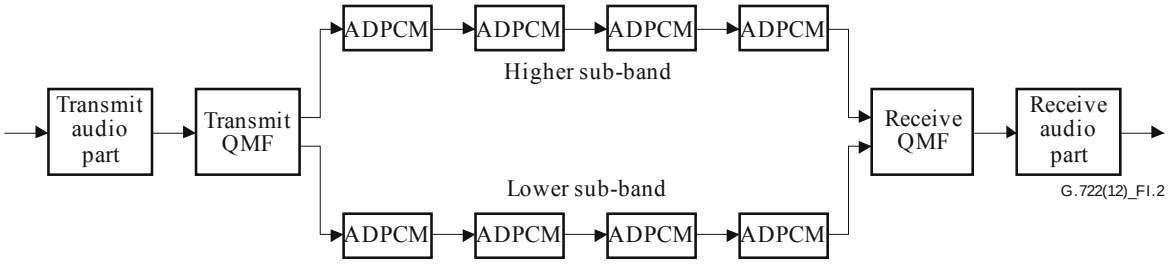


Figure I.2 – Four transcodings (digital interconnection)

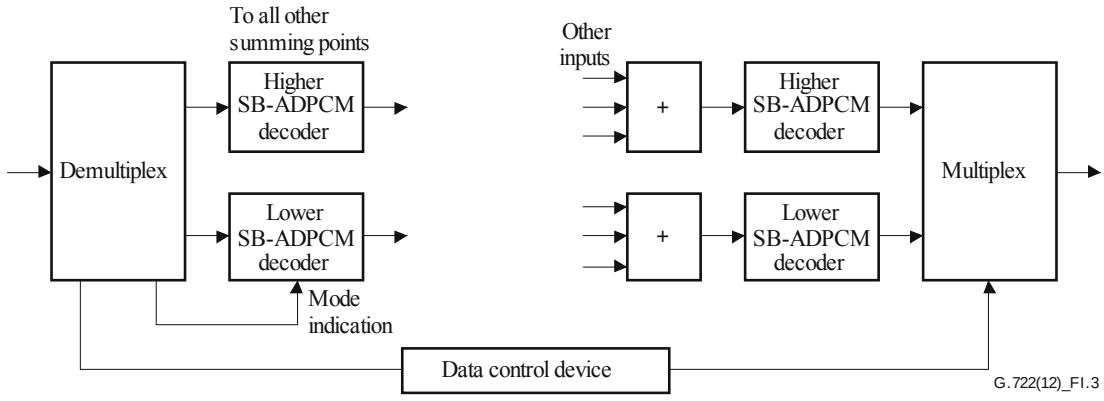


Figure I.3 – Possible arrangement at a multipoint conference unit

T Terminal
MCU Multipoint conference unit

Figure I.4 – Tandem connected multipoint conference units

I.8 Digital transcoding between the 64 kbit/s (7 kHz) audio-coding system and 64 kbit/s PCM

Figure I.5 indicates the method recommended for the digital interconnection of the 64 kbit/s (7 kHz) audio-coding system and 64 kbit/s PCM to Recommendation ITU-T G.711.

The principle of transcoding from 64 kbit/s PCM to 64 kbit/s (7 kHz) audio-coding involves the conversion from A-law or μ-law PCM to uniform PCM and the insertion of interleaved alternate samples of zero amplitude to the 8 kHz sampled uniform PCM signal to form a 16 kHz sampled signal. This signal is then passed through a digital low pass filter sampled at 16 kHz which does not significantly modify the baseband frequency response up to 3.4 kHz and which attenuates the frequency components above 4.6 kHz. The resulting signal is then applied to the sub-band ADPCM encoder as shown in Figure I.3.

It should be noted that the use of the lower sub-band alone to carry the information in a signal emanating from a 64 kbit/s PCM link to Recommendation ITU-T G.711 should be avoided.

An alternative method of deriving two sub-band signals from a 64 kbit/s PCM signal using the low pass (LP) and high pass (HP) QM filter designs already employed for the 64 kbit/s (7 kHz) audio-coding scheme is given in Figure I.6. The objective is to generate a higher sub-band signal which will eventually cancel the aliasing distortion introduced into the lower sub-band signal. The 64 kbit/s PCM signal is converted to uniform PCM and upsampled to 16 kHz by inserting alternate zero-valued samples. The factor 2 multiplier is inserted to preserve unity gain. The lower sub-band signal is derived by two identical stages of HP QM filtering following by 2:1 subsampling. The higher sub-band signal is derived by two filtering stages, HP followed by LP, a factor 1/2 gain reduction, sign inversion, followed by 2:1 subsampling. When these two signals are input to the QM synthesis filter of Recommendation ITU-T G.722, an appropriate 7 kHz form of the original PCM is obtained.

Note that the upsampling and subsampling process should be synchronized so that instants of sample deletion correspond to the instants of zero-sample insertion.

Transcoding from 64 kbit/s (7 kHz) audio-coding to 64 kbit/s PCM can be achieved by taking the output signal from the sub-band ADPCM decoder and performing the following three processes in turn:

– digital low pass filtering (16 kHz sampling), which does not significantly modify the baseband frequency response up to 3.4 kHz and which attenuates the frequency components above 4.6 kHz;
– the deletion of alternate samples from the resulting 16 kHz sampled signal;
– conversion from the resulting 8 kHz sampled uniform PCM signal to A-law or μ-law PCM.

NOTE – The derivation of a 64 kbit/s PCM signal solely from the lower sub-band of the 64 kbit/s (7 kHz) signal is subject to further study.

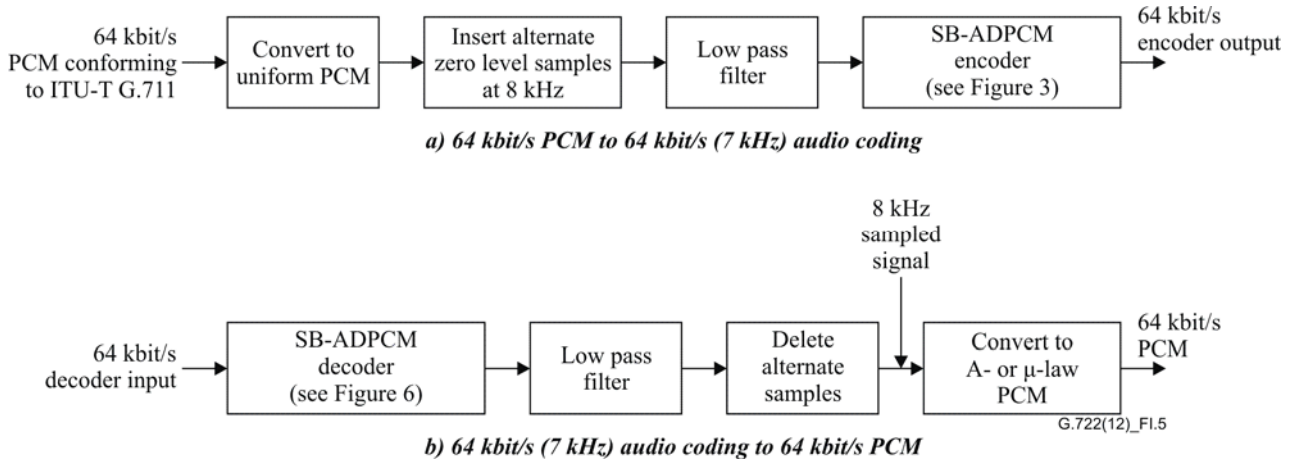


Figure I.5 – Digital transcoding between the 64 kbit/s (7 kHz) audio-coding system and 64 kbit/s PCM conforming to Recommendation ITU-T G.711

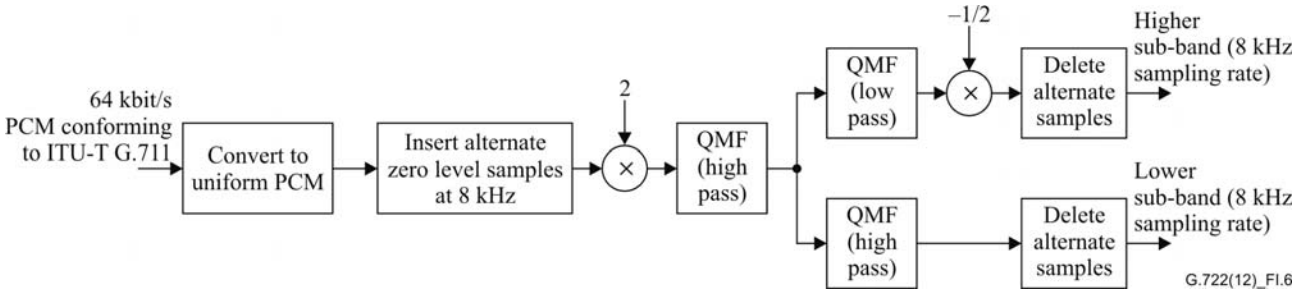


Figure I.6 – An alternative method for digital transcoding between 64 kbit/s PCM conforming to Rec. ITU-T G.711 and 64 kbit/s (7 kHz) audio-coding

Appendix II

Digital test sequences

(This appendix does not form an integral part of this Recommendation.)

This appendix gives information concerning the digital test sequences which should be used to aid verification of implementations of the ADPCM codec part of the wideband coding algorithm. Copies of the sequences are available as an electronic attachment to ITU-T G.722, as well as from the ITU-T Test Signal Database (see clause II.4).

II.1 Input and output signals

Table II.1 defines the input and output signals for the test sequences. It contains some signals (indicated by #) peculiar to these test sequences in order to facilitate the interface between the test sequence generator/receiver and the encoder/decoder. 16-bit word formats for these input and output signals are shown in Figures II.1, II.2 and II.3.

II.2 Configurations for the application of test sequences

Two configurations (Configuration 1 and Configuration 2) are appropriate for use with test sequences. In both configurations, a TEST signal is used to make the encoder and decoder ready to be tested with the digital test sequences. When the TEST signal is provided, the QMFs are by-passed and the test sequences are applied directly to the ADPCM encoders or decoders. An RSS signal is extracted from the input test sequences X # (I # in decoder) and results in a reset signal RS for the encoder and decoder. The RS signal will be used to initialize state variables (those indicated by * in Table 13 to zero or specific values.

II.2.1 Configuration 1

Configuration 1 shown in Figure II.4 is a simplified version of Figures 4 and 5. The encoder input signals, XL and XH, are described in Table 12. These input signals are directly fed to the respective lower and higher sub-band ADPCM encoders, by-passing the QMF. The encoder output signals, IL and IH, are defined in the sub-block QUANTL and QUANTH, respectively.

This sequence is used for testing the quantizer/predictor feedback loop in the encoder.

Table II.1 – Description of input and output signals for test sequence

Name	Description
XL	15-bit uniformly quantized input signal to the lower sub-band encoder
XH	15-bit uniformly quantized input signal to the higher sub-band encoder
X #	Input test sequence with 16-bit word format as shown in Figure II.1
IL	6-bit lower sub-band ADPCM codeword
ILR	Received 6-bit lower sub-band ADPCM codeword
IH	2-bit higher sub-band ADPCM codeword
I #	Output (in Configuration 1) and Input (en Configuration 2) test sequence with 16-bit word format as shown in Figure II.2
RL	15-bit uniformly quantized output signal from the lower sub-band decoder
RH	15-bit uniformly quantized output signal from the higher sub-band decoder
RL #	Output test sequence with 16-bit word format as shown in Figure II.3
RH#	Output test sequence with 16-bit word format as shown in Figure II.3
RSS	Reset/synchronization signal
VI	Valid data indication signal

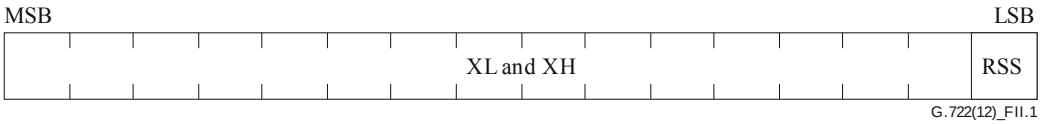
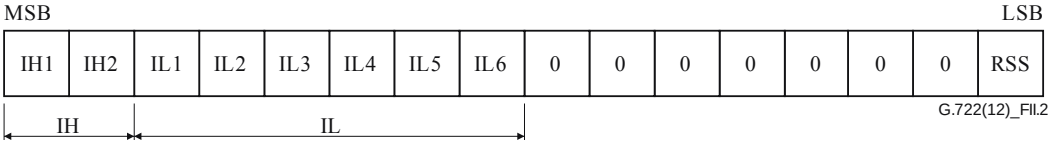


Figure II.1 – Word format of X#



NOTE 1 – IH1 and IL1 are MSBs of IH and IL, respectively.
NOTE 2 – IL is read as ILR in configuration 2.

Figure II.2 – Word format of I#

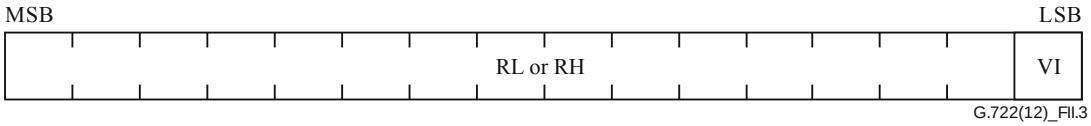


Figure II.3 – Word format of RL# and RH#

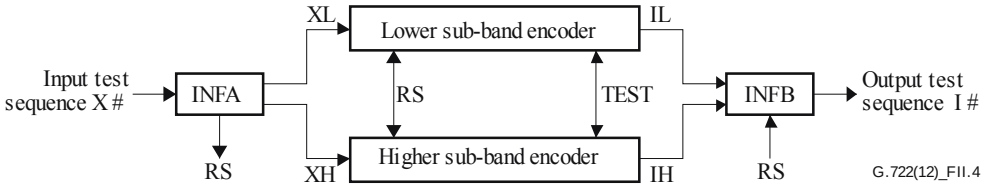


Figure II.4 – Configuration 1 – encoder only

II.2.2 Configuration 2

Configuration 2 shown in Figure II.5 is a simplified version of Figures 7 and 8. The test signals, ILR and IH, and the MODE signal are described in Table 12. The corresponding decoder output signals, RL and RH, are defined in the sub-blocks LIMIT in clauses 6.2.1.6 and 6.2.2.5. For the lower sub-band, the ADPCM decoder output signals are derived for three basic modes of operation

(Modes 1, 2 and 3). By-passing the QMF, the output signals, RL and RH, are separately obtained from the lower and higher sub-band ADPCM decoders, respectively.

Configuration 2 is used for testing the inverse quantizer operation and the predictor adaptation without a quantizer/predictor feedback loop in the decoder.

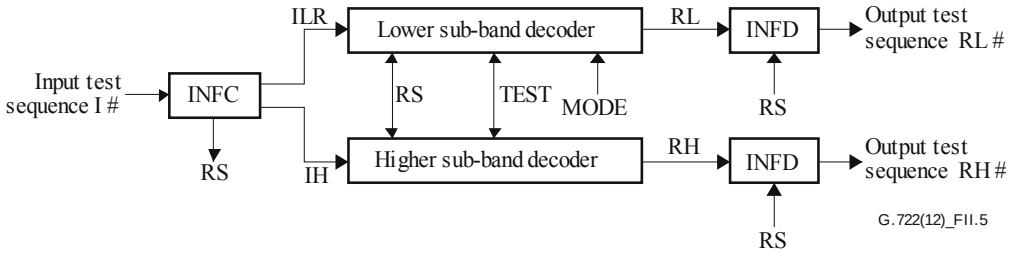


Figure II.5 – Configuration 2 – decoder only (RL and RL# are derived for Modes 1, 2 and 3)

II.2.3 Reset/synchronization signal (RSS) and valid data indication (VI)

All memory states in the two test configurations must be initialized to the exact states specified in this Recommendation prior to the start of an input test sequence in order to obtain the correct output values for the test.

In Configuration 1, the input test sequence, X#, is composed of encoder input test signals and the reset/synchronization signal (RSS) as shown in Figure II.1. The RSS signal is located at the first LSB of the input sequence. If RSS is "1", the lower and higher sub-band encoders are initialized, and the outputs of the encoders are set to "0", i.e., IH = "0" and IL = "0". This normally forbidden output code is used to indicate "non-valid data" of the outputs. After the RSS signal goes to "0", the input test sequence will be valid and the ADPCM algorithm begins to operate.

In Configuration 2, the input test sequence, I #, is composed of the first 8 bits of lower and higher sub-band decoder input codewords, and the last 8 bits consists of 7-bit zeroes and "RSS" in the LSB as shown in Figure II.2. The RSS signal has the same role as in Configuration 1. That is, if the RSS signal equals "1", the lower and higher sub-band decoders are initialized. After the RSS signal goes to "0", the ADPCM algorithm will be in the operational state. The output test sequences, RL# and RH#, are made up of a decoder output signal of 15 bits and a valid data indication signal (VI) as shown in Figure II.3. While the RSS signal to the decoder is "1", the signal "VI" is set to "1" and the decoder output set to "0", which indicates "non-valid data" of the output. When "VI" is "0", the output test sequence is valid.

In order to establish the connection between the test sequence generator/receiver and the encoder/decoder, four sub-blocks, INFA, INFB, INFC, INFD in Figures II.4 and II.5 are provided. A detailed expansion of these sub-blocks is described below using the same notations specified in clause 6.2.

INFA

Input: X#

Outputs: XL, XH, RS

Function: Extract reset/synchronization signal and input signals to lower and higher sub-band ADPCM encoder.

RS = X# & 1	|	Extract RSS signal
XL = S# >> 1	|	Lower sub-band input signal
XH = XL	|	Higher sub-band input signal

INFB

Inputs: IL, IH, RS

Outputs: I#

Function: Create an output test sequence by combining lower and higher sub-band ADPCM encoder output signals and the reset/synchronization signal.

I=	(IH <<< 6) + IL	if RS = = 0	|	Combine IH and IL
	0	if RS = = 1	|	Set output to zero
I# = (I <<< 8) + RS			|	Add RSS signal

INFC

Input: I#

Outputs: ILR, IH, RS

Function: Extract reset/synchronization signal and input signals to lower and higher sub-band ADPCM decoder.

RS = I# & 1	|	Extract RSS signal
ILR = (I # >>> 8) & 63	|	Lower sub-band ADPCM input
IH = I # >>> 14	|	Higher sub-band ADPCM input

INFD

Inputs: RL (RH in higher sub-band), RS

Output: RL#(RH# in higher sub-band)

Function: Create output test sequence by combining lower (higher) sub-band ADPCM decoder output signal and the valid data indication signal.

RLX =	RL << 1	if RS = = 0	|	Scaling by 1-bit shift
	0	if RS = = 1	|	Set output to zero
RL# = RLX + RS			|	Add VI signal

II.3 Test sequences

II.3.1 Input sequences for Configuration 1

For Configuration 1, two types of input test sequences are provided:

1) sequence containing tones, d.c. and white noise,

2) sequence for testing overflow controls in the ADPCM encoders.

The first input sequence contains tones with various frequencies, DC and white noise with two levels. The signal segments and lengths are given in Table II.2.

The tones are used to move the predictor poles over their operating range and to test the stability control. Although the second pole coefficients are settled only in the vicinity of their lower limit for tone inputs, the upper limit is examined at the beginning of the d.c.-positive input. d.c. and white noise are used to vary the quantizer scale factors over their entire range.

The second input sequence permits testing of frequent overflows. The signal segments and lengths are given in Table II.3.

The sequences produces large prediction errors, so it is used to check the overflow controls in pole and zero section output computations.

In Configuration 1, the coefficient values of the zero predictor do not move to the range limits of –2 and +2.

II.3.2 Input sequences for Configuration 2

For Configuration 2, these types of input test sequences are provided:

1) The sequence generated by the encoder is used when applying the input test sequence described in Table II.2;
2) The sequence generated by the encoder is used when applying the input test sequence described in Table II.3;
3) An artificial sequence containing consecutive sub-sequences is used that would not ordinarily emanate from an encoder.

The third test sequence, consisting of 16384 values, is described below.

Table II.2 – Sequence of tones, d.c. and white noise

Signal segments	Length (16 bits words)
3 504 Hz tone	1 024
2 054 Hz tone	1 024
1 504 Hz tone	1 024
504 Hz tone	1 024
254 Hz tone	1 024
1 254 Hz tone	1 024
2 254 Hz tone	1 024
3 254 Hz tone	1 024
4 000 Hz tone	512
d.c, positive, low level	512
d.c., value of zero	512
d.c., negative, low level	512
White noise, low level	3 072
White noise, high level	3 072
Total length of sequence	16 384

Table II.3 – Overflow test sequence

Signal segments	Length (16 bits words)
–16 384, +16 383; repeated	639
0, –10 000, –8192	3
–16 384, +16 383, –16 384; repeated	126
Total length of sequence	768

II.3.2.1 Lower sub-band ADPCM codewords

The 6-bit lower sub-band decoder input sequence consists of an MSB sequence and a distinct sequence of the remaining 5 bits. The MSB sequence consists of eight artificial sub-sequences, each 2 048 bits in length, as follows:

(1) 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0.....

(2) 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1.....

(3) 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1.....

(4) 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1.....

(5) 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1.....

(6) 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0.....

(7) 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0.....

(8) 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1.....

These MSB sequences are used to force the coefficients of the zero predictor to vary across the entire range of ± 2.

The associated 5-bit word sequence consists of 64 concatenated artificial sub-sequences, each 256 values long, as described in Table II.4. This 5-bit word sequence was chosen to exercise the logarithmic quantizer scale factor over its entire range, and the log-to-linear conversion.

The composite sequence of ILR also tests the pole predictor and varies its coefficients over their allowable range. The sequences from sub-sequence numbers (56) to (64) test the conversion from the suppressed codewords, which can occur due to transmission errors, to specified quantizer intervals.

II.3.2.2 Higher sub-band ADPCM codewords

The 2-bit higher sub-band decoder input sequence consists of an MSB sequence and a distinct LSB sequence.

The MSB sequence consists of eight artificial sub-sequences, identical to those used in the MSB sequence for the lower sub-band ADPCM.

The LSB sequence consists of 8 concatenated artificial sub-sequences, each 2048 bits long, as follows:

(1) 1 1 1 1 1 1.....

(2) alternating sixteen 1s, sixteen 0s

(3) 0 0 0 0 0 0.....

(4) alternating eight 1s, eight 0s

(5) 0 0 0 0 0 0.....

(6) alternating four 1s, four 0s

(7) 1 1 1 1 1 1.....

(8) alternating two 1s, two 0s.

The role of the composite sequence formed by appending the l-bit LSB to the l-bit MSB is equivalent to that for the lower sub-band ADPCM codeword described in clause II.3.2.1.

II.4 Format for test sequence distribution

II.4.1 Format

Copies of the digital test sequences are available as an electronic attachment to this Recommendation as well as for download from the ITU-T Test Signal Database at http://itu.int/net/itu-t/sigdb/speaudio/Gseries.htm#G.722.

The files are written in ASCII text in order to be dumped, listed or edited easily, but are also available in binary format.

Table II.4 – Sequence of last 5 bits of ILR

Repetitive pattern, each 256 values long	
(1) 31 31 31 31 31 31	(33) 15 15 15 15 15 15 . . . . .
(2) alternating sixteen 31's, sixteen 30's	(34) alternating sixteen 15's, sixteen 14's
(3) 30 30 30 30 30 30.	(35) 14 14 14 14 14 14 . . . . .
(4) alternating sixteen 30's, sixteen 29's	(36) alternating sixteen 14's, sixteen 13's
(5) 29 29 29 29 29 29	(37) 13 13 13 13 13 13 . . . . .
(6) alternating sixteen 29's, sixteen 28's	(38) alternating sixteen 13's, sixteen 12's
(7) 28 28 28 28 28 28	(39) 12 12 12 12 12 12 . . . . .
(8) alternating sixteen 28's, sixteen 27's	(40) alternating sixteen 12's, sixteen 11's
(9) 27 27 27 27 27 27	(41) 11 11 11 11 11 11 . . . . .
(10) alternating sixteen 27's, sixteen 26's	(42) alternating sixteen 11's, sixteen 10's
(11) 26 26 26 26 26 26	(43) 10 10 10 10 10 10 . . . . .
(12) alternating sixteen 26's, sixteen 25's	(44) alternating sixteen 10's, sixteen 9's
(13) 25 25 25 25 25 25	(45) 9 9 9 9 9 9 . . . . .
(14) alternating sixteen 25's, sixteen 24's	(46) alternating sixteen 9's, sixteen 8's
(15) 24 24 24 24 24 24	(47) 8 8 8 8 8 8 . . . . .
(16) alternating sixteen 24's, sixteen 23's	(48) alternating sixteen 8's, sixteen 7's
(17) 23 23 23 23 23 23	(49) 7 7 7 7 7 7 . . . . .
(18) alternating sixteen 23's, sixteen 22's	(50) alternating sixteen 7's, sixteen 6's
(19) 22 22 22 22 22 22	(51) 6 6 6 6 6 6 . . . . .
(20) alternating sixteen 22's, sixteen 21's	(52) alternating sixteen 6's, sixteen 5's
(21) 21 21 21 21 21	(53) 5 5 5 5 5 5 . . . . .
(22) alternating sixteen 21's, sixteen 20's	(54) alternating sixteen 5's, sixteen 4's
(23) 20 20 20 20 20 20	(55) 4 4 4 4 4 4 . . . . .
(24) alternating sixteen 20's, sixteen 19's	(56) alternating sixteen 4's, sixteen 3's
(25) 19 19 19 19 19 19	(57) 3 3 3 3 3 3 . . . . .
(26) alternating sixteen 19's, sixteen 18's	(58) alternating sixteen 3's, sixteen 2's
(27) 18 18 18 18 18 18	(59) 2 2 2 2 2 2. . . . .
(28) alternating sixteen 18's, sixteen 17's	(60) alternating sixteen 2's, sixteen 1's
(29) 17 17 17 17 17 17 17	(61) 1 1 1 1 1 1 . . . . .
(30) alternating sixteen 17's, sixteen 16's	(62) alternating sixteen 1's, sixteen 0's
(31) 16 16 16 16 16 16	(63) 0 0 0 0 0 0 . . . . .
(32) alternating sixteen 16's, sixteen 15's	(64) alternating sixteen 0's, sixteen 3's

II.4.2 Type of files provided

The test sequences are arranged into 7 files. These 17 files are classified in 3 groups according to the following description:

Class T1: Source files to be input to the ADPCM codec. Class T1 includes 2 files to be used in Configuration 1 (encoder only) and 1 file to be used in Configuration 2 (decoder only).

Class T2: Combined source-comparison files. There are 2 files in class T2. Both are used for comparison purposes at the output of the encoder in Configuration 1. Also they are used as source files to test the decoder in Configuration 2.

Class T3: Comparison files used to check the output of the decoder in different modes. There are 9 files in class T3 to test the lower sub-band decoder and 3 files in the same class to test the higher sub-band decoder. In class T3, the suffix H or L in the file name distinguishes the higher and lower sub-band. Also a number from 1 to 3 in the file name indicates the corresponding mode used for the test.

II.4.3 Directory of the test sequence files

This section gives the name and the content of each file provided for the digital test sequences. Figure II.6 shows which files are to be used in the different configurations of test.

Class T1 file names

T1C1.XMT: 16 416 test values (16-bit words) containing various frequencies, d.c., white noise for encoder test.

T1C2.XMT: 800 test values (16-bit words) containing the artificial sequence to test overflow in the encoder.

T1D3.COD: 16 416 test values (16-bit words) containing the artificial sequence for the decoder test. The most significant 8 bits contain the ADPCM code (IH, IL) and the least significant 8 bits contain the RSS information (reset/synchronisation signal).

Class T2 file names

T2R1.COD: 16 416 test values (16-bit words) containing the output code for the T1C1.XMT file. This file is also used as an input to test the decoder, and consequently has the same structure as the T1D3.COD file.

T2R2.COD: 800 test values (16 bit words) containing the output code for the T1C2.XMT file. This file is also used as source to test the decoder and consequently has the same structure as the T1D3.COD file.

Class T3 file names

T3L1.RC1 16 416 test values (1 6-bit words) containing the output of the lower sub-band decoder in Mode 1 when the file T2R1.COD is used as an input.

T3L1.RC2 same meaning as for T3L1.RC1 file but when Mode 2 is used.

T3L1.RC3 same meaning as for T3L1.RC1 file but when Mode 3 is used.

T3H1.RC0 16 416 test values (16-bit words) containing the output of the higher sub-band decoder when the file T2R1.COD is used as an input.

T3L2.RC1 800 test values (16-bit words) containing the output of the lower sub-band decoder in Mode 1 when the file T2R2.COD is used as an input.

T3L2.RC2 same meaning as for T3L2.RC1 file but when Mode 2 is used.

T3L2.RC3 same meaning as for T3L2.RC1 file but when Mode 3 is used.

T3H2.RC0 800 test values (16-bit words) containing the output of the higher sub-band decoder when the file T2R2.COD is used as an input.

T3L3.RC1 16 416 test values (16-bit words) containing the output of the lower sub-band decoder in Mode 1 when the file T1 D3.COD is used as an input.

T3L3.RC2 same meaning as for T3L3.RC1 file but when Mode 2 is used.

T3L3.RC3 same meaning as for T3L3.RC1 file but when Mode 3 is used.

T3H3.RC0 16 416 test values (16-bit words) containing the output of the higher sub-band decoder when the file T1D3.COD is used as an input.

NOTE – Mode indication must be set by the user of the digital test sequences.

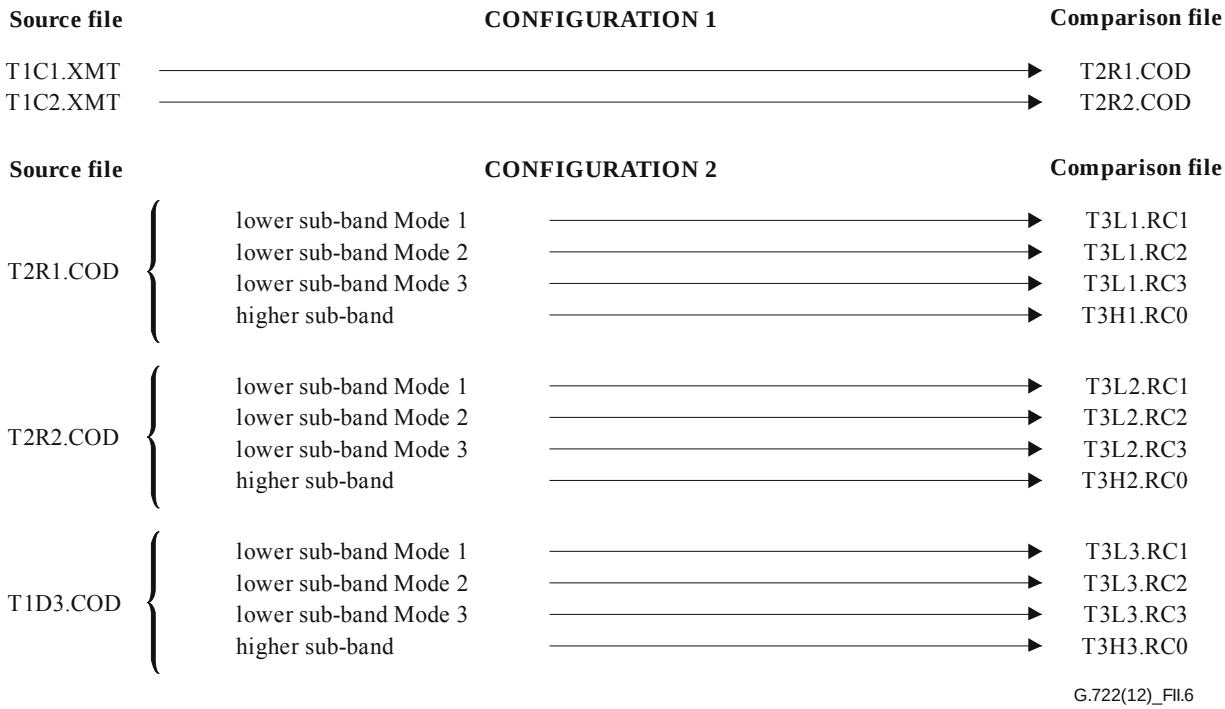


Figure II.6 – Configuration of test

II.4.4 File format description

All the files are available either in ASCII or binary formats.

II.4.4.1 ASCII files

For files written in ASCII with a line structure, the first two lines of each file give some information on the file content. The following format is used for the two first lines:

/* CCITT 64 KBIT/S SB-ADPCM DIGITAL TEST SEQUENCE ITU-T G.722 */

/* FILE NAME: xxxx.eee DATE: mm-dd-yy VERSION: V 1.0 */

For the subsequent lines of the file, 16 test values (16-bit words, 64 hexadecimal characters) are followed by a checksum on 1 byte (2 hexadecimal characters), a carriage return (ASCII code 0D in hexadecimal), and a line feed (ASCII code 0A in hexadecimal). These last two characters are non-printable.

The checksum is the two's complement of the least significant 8 bits of the summation of all the preceding characters (ASCII codes) in the line. If the least significant 8 bits of the summation are all zero, the corresponding two's complement is set all zero.

At the end of each file, a line of comment closes the file. This line is:

/* END OF FILE: xxxx.eee */

II.4.4.2 Binary format files

For the binary files, all test sequence values are put in 16-bit format (without checksums). They are available either in little- or big-endian representation, and appropriate format should be used depending on the CPU architecture where the test software is being run.

II.4.5 Internal line description

II.4.5.1 File with extension .XMT

– 16 words of 16 bits with the LSB set to 1, all others set to zero (RSS = 1: reset mode);

– 16 384 or 768 words of 16 bits of digital test sequence with RSS = 0 (RSS is the LSB of the lower byte of the word);

– 16 words of 16 bits with the LSB set to 1, all others set to 0 (marks for end of test sequence).

II.4.5.2 File with extension .COD

– 16 words of 16 bits with the LSB set to 1, all others set to 0 (RSS = 1: reset mode and the ADPCM code set to 0);

– 16 384 or 768 words of 16 bits of digital test sequence with RSS = 0 (RSS is the LSB of the lower byte of the word and the upper byte is the ADPCM code);

– 16 words of 16 bits with the LSB set to 1, all others set to zero (marks for end of test sequence).

II.4.5.3 File with extension .RCx

– 16 words of 16 bits with the LSB set to 1, all others set to 0 (this means that these words are non-valid data);

– 16 384 or 768 words of 16 bits of digital test sequence with the LSB of the lower byte set to 0 to indicate valid data;

– 16 words of 16 bits with the LSB set to 1, all others set to 0 (marks for end of test sequence).

II.4.6 Distribution of the ITU-T digital test sequences

The digital test sequences are provided as an electronic attachment to ITU-T G.722, as well as from the ITU-T Test Signal Database (http://itu.int/net/itu-t/sigdb/speaudio/Gseries.htm#G.722). The file sizes for the ASCII version of the test sequences are given in Table II.5. Table II.6 provides the size and CRC-32 for the binary version of the test sequences for little endian and big endian formats.

Table II.5 – Digital test sequences in ASCII format: file names and sizes

File name	Number of bytes
T1C1. XMT	69 973
T1C2. XMT	3 605
T1D3. COD	69 973
T2R1. COD	69 973
T2R2. COD	3 605
T3L1. RC1	69 973
T3L1. RC2	69 973
T3L1. RC3	69 973
T3H1. RC0	69 973
T3L2. RC1	3 605
T3L2. RC2	3 605

Table II.5 – Digital test sequences in ASCII format: file names and sizes

File name	Number of bytes
T3L2. RC3	3 605
T3H2. RC0	3 605
T3L3. RC1	69 973
T3L3. RC2	69 973
T3L3. RC3	69 973
T3H3. RC0	69 973

Table II.6 – Digital test sequence in binary format: file names, sizes and CRC-32 for little endian and big endian formats

File name	Size (bytes)	CRC-32 (Little Endian)	CRC-32 (Big Endian)
bt1c1.xmt	32 832	0C3BFCA7	015ACCE4
bt1c2.xmt	1 600	2D604685	EAFC99B4
bt1d3.cod	32 832	7398964F	1C85BE45
bt2r1.cod	32 832	D1DAA1D1	0B904231
bt2r2.cod	1 600	344EA5D0	F928980D
bt3h1.rc0	32 832	E9250851	0BDE9C9C
bt3h2.rc0	1 600	5330AE2E	3A54C7DF
bt3h3.rc0	32 832	3731AD7F	8E8DEE65
bt3l1.rc1	32 832	ED1B3993	90DD2D72
bt3l1.rc2	32 832	8E8C4E2B	FE7C4611
bt3l1.rc3	32 832	B7AA5569	20B5FFC4
bt3l2.rc1	1 600	AF00F31F	D2599DE8
bt3l2.rc2	1 600	9143E92C	84041F43
bt3l2.rc3	1 600	AE855C07	F32628D9
bt3l3.rc1	32 832	A5374659	8C12ED04
bt3l3.rc2	32 832	687B250A	550534A7
bt3l3.rc3	32 832	3605736B	9354E9CF

NOTE – The letter "b" is added to the beginning of all file names, compared to the ASCII version, to denote that these are the binary version of the files and avoid accidental overwriting.

Appendix III

A high-quality packet loss concealment algorithm for ITU-T G.722

(This appendix does not form an integral part of this Recommendation.)

III.1 Scope

This appendix describes a high-quality packet loss concealment algorithm for ITU-T G.722. The statistical analysis of the ITU-T G.722 PLC Selection Test results has demonstrated that the PLC algorithm in this appendix was clearly the best performing PLC among the solutions examined (including the PLC in ITU-T G.722 Appendix IV) in terms of speech quality for applications of ITU-T G.722 in the presence of packet loss. This appendix meets the same complexity requirements as Appendix IV, but with a higher complexity. Due to its high quality, this appendix is suitable for general applications of ITU-T G.722 that may encounter frame erasures or packet loss. As examples, these applications include VoIP, Voice over WiFi, and DECT next generation. The algorithm of this appendix adds a complexity of 2.8 WMOPS worst case and 2 WMOPS on average to the ITU-T G.722 decoder. It is easy to accommodate, except for applications where there is practically no complexity headroom left after implementing the basic ITU-T G.722 decoder without packet loss concealment.

III.2 References

[ITU-T G.191 An.A] Recommendation ITU-T G.191 Annex A (2005), *Software tools for speech and audio coding standardization, Annex A: List of software tools available.*

[ITU-T G.192] Recommendation ITU-T G.192 (1996), *A common digital parallel interface for speech standardization activities*.

III.3 Abbreviations and acronyms

This appendix uses the following abbreviations and acronyms:

ADPCM	Adaptive Differential PCM
DECT	Digital Enhanced Cordless Telecommunications
FIR	Finite Impulse Response
LPC	Linear Predictive Coding
OLA	Overlap-Add
PCM	Pulse Code Modulation
PLC	Packet Loss Concealment
PWE	Periodic Waveform Extrapolation
QMF	Quadrature Mirror Filter
STL2005	Software Tool Library 2005
VoIP	Voice over Internet Protocol
WB	Wideband
WiFi	Wireless Fidelity

III.4 Conventions

For the purposes of this appendix, the terms "frame loss" and "packet loss" are used interchangeably.

This appendix uses the following conventions:

– The PLC operates at an intrinsic frame size of 10 ms, and hence, the algorithm is described for 10-ms frames only. For packets of larger size (multiples of 10 ms) the received packet is decoded in 10-ms sections.
– The discrete time index of signals at the 16-kHz sampling rate level is generally referred to with either "j" or "i".
– The discrete time of signals at the 8-kHz sampling level is typically referred to with "n".
– Low-band signals (0-4 kHz) are identified with a subscript "L".
– High-band signals (4-8 kHz) are identified with a subscript "H".
– This appendix follows the conventions of [ITU-T G.722] where possible.

The following is a list of the most frequently used symbols and their descriptions:

$x_{out}(j)$	16-kHz ITU-T G.722 decoder output
$x_{PLC}(i)$	16-kHz ITU-T G.722 PLC output
$x_{out,\ FGF}(j)$	16-kHz ITU-T G.722 decoder output for the 'first good frame' received after packet loss
$w(j)$	LPC window
$x_w(j)$	Windowed speech
$r(i)$	Autocorrelation
$\hat{r}(i)$	Autocorrelation after spectral smoothing and white noise correction
$\hat{a}_i$	Intermediate LPC predictor coefficients
a_i	LPC predictor coefficients
$d(j)$	16-kHz short-term prediction error signal
avm	Average magnitude of the short-term prediction residual signal
a'_i	Weighted short-term synthesis filter coefficients
$xw(j)$	16-kHz weighted speech
$xwd(n)$	Down-sampled weighted speech (2 kHz)
b_i	60th-order low-pass filter for down-sampling
$c(k)$	Correlation for coarse pitch analysis (2 kHz)
$E(k)$	Energy for coarse pitch analysis (2 kHz)
$c2(k)$	Signed squared correlation for coarse pitch analysis (2 kHz)
cpp	Coarse pitch period
$cpplast$	Coarse pitch period of last frame
$Ei(j)$	Interpolated E(k) (to 16 kHz)
$c2i(j)$	Interpolated c2(k) (to 16 kHz)
$\widetilde{E}(k)$	Energy for pitch refinement (16 kHz)
$\widetilde{c}(k)$	Correlation for pitch refinement (16 kHz)
$ppfe$	Pitch period for frame erasure
$ptfe$	Pitch tap for frame erasure

ppt	Pitch predictor tap
merit	Figure of merit of periodicity
G_r	Scaling factor for random component
G_p	Scaling factor for periodic component
ltring(*j*)	Long-term (pitch) ringing
ring(*j*)	Final ringing (including short-term)
wi(*j*)	Fade-in window
wo(*j*)	Fade-out window
wn(*j*)	Output of noise generator
wgn(*j*)	Scaled output of noise generator
fan(*j*)	Filtered and scaled noise
cfecount	Counter of consecutive 10-ms frame erasures
$w_i(j)$	Window for overlap-add
$w_o(j)$	Window for overlap-add
h_i	QMF filter coefficients
$x_L(n)$	Low-band sub-band signal (8 kHz)
$x_H(n)$	High-band sub-band signal (8 kHz)
$I_L(n)$	Index for low-band ADPCM coder (8 kHz)
$I_H(n)$	Index for high-band ADPCM coder (8 kHz)
$s_{Lz}(n)$	Low-band predicted signal, zero section contribution
$s_{Lp}(n)$	Low-band predicted signal, pole section contribution
$s_L(n)$	Low-band predicted signal
$e_L(n)$	Low-band prediction error signal
$r_L(n)$	Low-band reconstructed signal
$p_{Lt}(n)$	Low-band partially reconstructed truncated signal
$\nabla_L(n)$	Low-band log scale factor
$\Delta_L(n)$	Low-band scale factor
$\nabla_{L,m1}(n)$	Low-band log scale factor, 1st mean
$\nabla_{L,m2}(n)$	Low-band log scale factor, 2nd mean
$\nabla_{L,trck}(n)$	Low-band log scale factor, tracking
$\nabla_{L,chng}(n)$	Low-band log scale factor, degree of change
MPTH	Multiple pitch threshold
MPDTH	Multiple pitch deviation threshold
$\beta_L(n)$	Stability margin of low-band pole section
$\beta_{L,MA}(n)$	Moving average of stability margin of low-band pole section
$\beta_{L,min}$	Minimum stability margin of low-band pole section

$s_{Hz}(n)$ High-band predicted signal, zero section contribution

$s_{Hp}(n)$ High-band predicted signal, pole section contribution

$s_H(n)$ High-band predicted signal

$e_H(n)$ High-band prediction error signal

$r_H(n)$ High-band reconstructed signal

$r_{H,HP}(n)$ High-band high-pass filtered reconstructed signal

$p_H(n)$ High-band partially reconstructed signal

$p_{H,HP}(n)$ High-band high-pass filtered partially reconstructed signal

$\nabla_H(n)$ High-band log scale factor

$\nabla_{H,m}(n)$ High-band log scale factor, mean

$\nabla_{H,trck}(n)$ High-band log scale factor, tracking

$\nabla_{H,chng}(n)$ High-band log scale factor, degree of change

$\alpha_{LP}(n)$ Coefficient for low-pass filtering of high-band log scale factor

$\nabla_{H,LP}(n)$ Low-pass filtered high-band log scale factor

$r_{Le}(n)$ Estimated low-band reconstructed error signal

$es(n)$ Extrapolated signal for time lag calculation of re-phasing

$R_{SUB}(k)$ Sub-sampled normalized cross-correlation

$R(k)$ Normalized cross-correlation

T_{LSUB} Sub-sampled time lag

T_L Time lag for re-phasing

$es_{tw}(n)$ Extrapolated signal for time lag refinement for time-warping

T_{Lwarp} Time lag for time-warping

$X_{warp}(j)$ Time-warped signal (16 kHz)

$es_{ola}(j)$ Extrapolated signal for overlap-add (16 kHz)

III.5 General description of the PLC algorithm

For ease of understanding, six types of frames are defined and referred to in the text:

– Type 1: Received frame, with no lost packets in the preceding eight frames
– Type 2: Lost frames 1 and 2 of packet loss
– Type 3: Lost frames 3 through 6 of packet loss
– Type 4: Lost frames beyond frame 6 of packet loss
– Type 5: First frame received immediately following packet loss
– Type 6: Received frames 2 through 8 following packet loss

This is illustrated with the time-line example in Figure III.1. The PLC algorithm operates on an intrinsic frame size of 10 ms in duration.

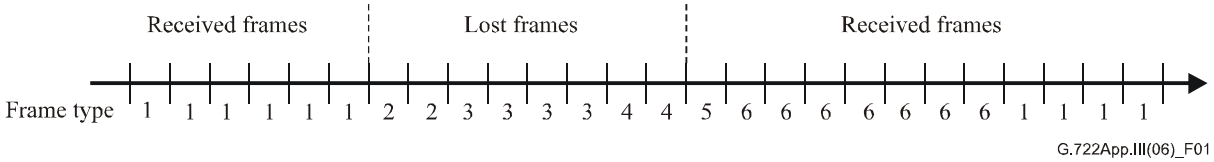


Figure III.1 – Time-line of frame types

Type 1

Type 1 frames are decoded according to [ITU-T G.722] with the addition of maintaining some state memory and processing to facilitate the PLC and associated processing. This is shown in Figure III.2.

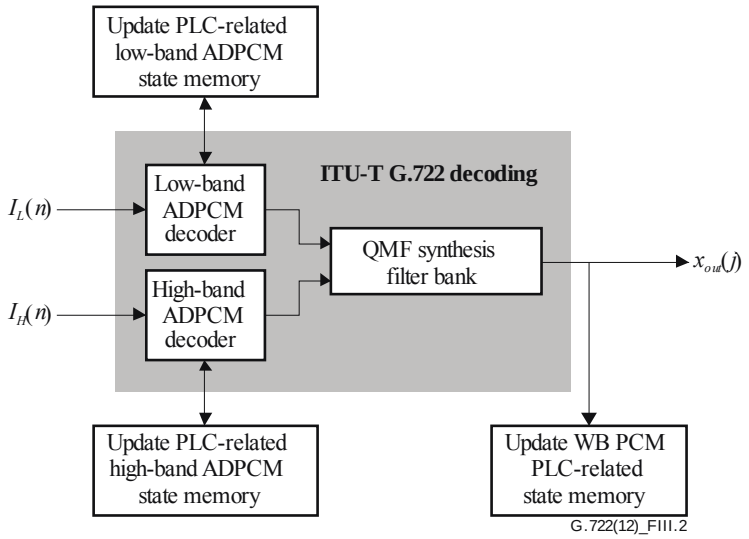


Figure III.2 – Regular ITU-T G.722 decoding (Type 1)

Types 2, 3 and 4

The algorithm performs wideband (WB) PCM PLC in the 16-kHz output speech domain for frames of Types 2, 3 and 4. A block diagram of the WB PCM PLC is shown in Figure III.3. Past output speech of ITU-T G.722 is buffered and passed to the WB PCM PLC. The WB PCM PLC is based on periodic waveform extrapolation (PWE), and pitch estimation is an important component of the WB PCM PLC. Initially, a coarse pitch is estimated based on a down-sampled (to 2 kHz) signal in the weighted speech domain. Subsequently, this estimate is refined at full resolution using the original 16-kHz sampling. The WB PCM PLC output is a linear combination of the periodically extrapolated waveform and noise shaped by LPC. For extended erasures, the output waveform is gradually muted. The muting starts after 20 ms of frame loss and becomes complete after 60 ms of loss.

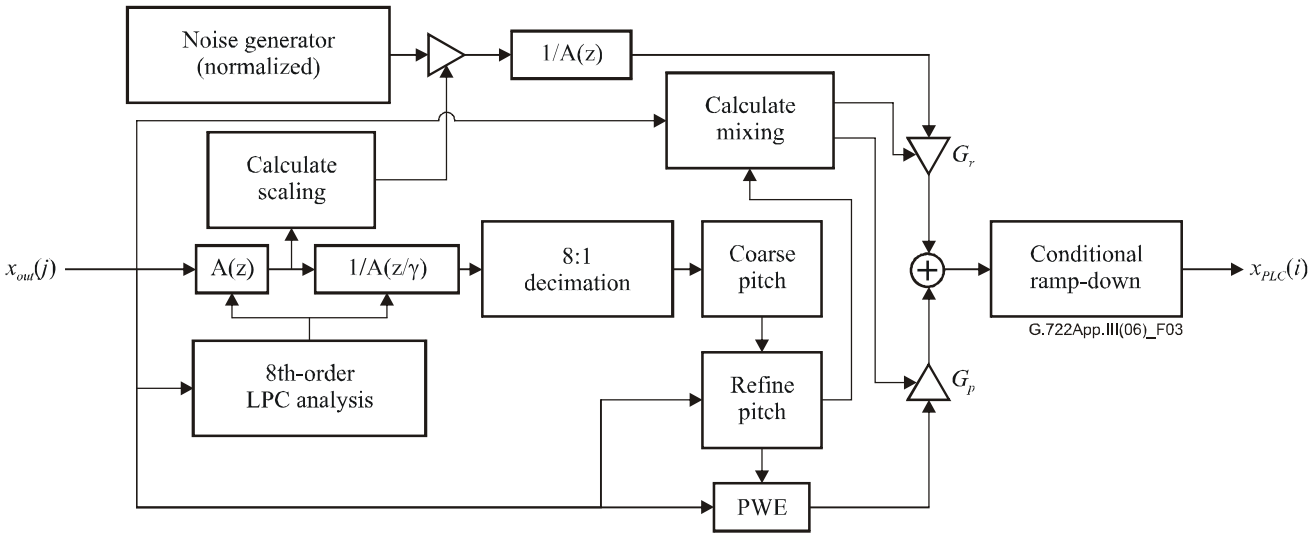


Figure III.3 – Block diagram of WB PCM PLC (for frames of Types 2, 3 and 4)

As shown in Figure III.4, for frames of Types 2, 3 and 4, the output of the WB PCM PLC is passed through the ITU-T G.722 QMF analysis filter bank to obtain corresponding sub-band signals that are subsequently passed to modified low-band and high-band ADPCM encoders, respectively, in order to update the states and memory of the decoder. Only partial simplified sub-band ADPCM encoders are used for this update.

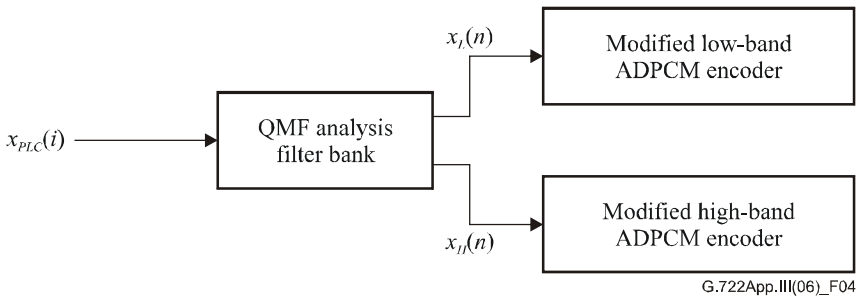


Figure III.4 – Re-encoding of PLC output to update sub-band ADPCM states (for frames of Types 2, 3 and 4)

The processing in Figures III.3 and III.4 takes place during lost frames. The modified low-band and high-band ADPCM encoders of Figure III.4 are simplified to reduce complexity. They are described in detail in clause III.7. One feature that is different from the regular encoders is an adaptive reset of the decoders based on signal property and the duration of packet loss.

Type 5

The most complex processing takes place for frame Type 5. This is the first received frame, where the transition from extrapolated to decoded waveform takes place. Primary techniques are re-phasing and time-warping. A high-level block diagram of re-phasing and time-warping is shown in Figure III.5.

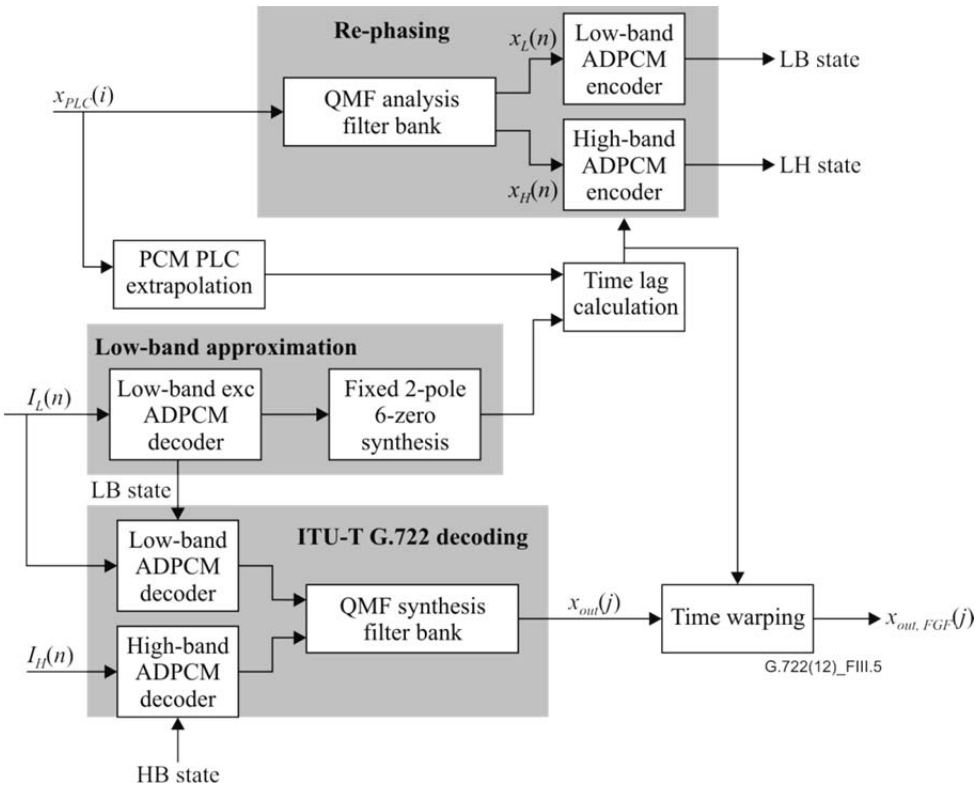


Figure III.5 – Re-phasing and time-warping (Type 5)

Additionally, at the first received frame following packet loss, the memory of the QMF synthesis filter bank at the decoder needs to be updated. See clause III.9 for details. Other important processing takes place in frames of Type 5 as well. This includes techniques such as adaptive setting of low-band and high-band log-scale factors at the beginning of the first received frame (clauses III.8.1 and III.8.2). Furthermore, the state memory update indicated for Type 1 frames applies to Type 5 frames as well. The techniques of clauses III.8.2.3, III.8.3 and III.8.4 apply to Type 5 frames and extend into Type 6 frames.

Types 5 and 6

Frames of Type 6 are decoded with modified and constrained sub-band ADPCM decoders. The block diagram for decoding of Type 6 frames is depicted in Figure III.6.

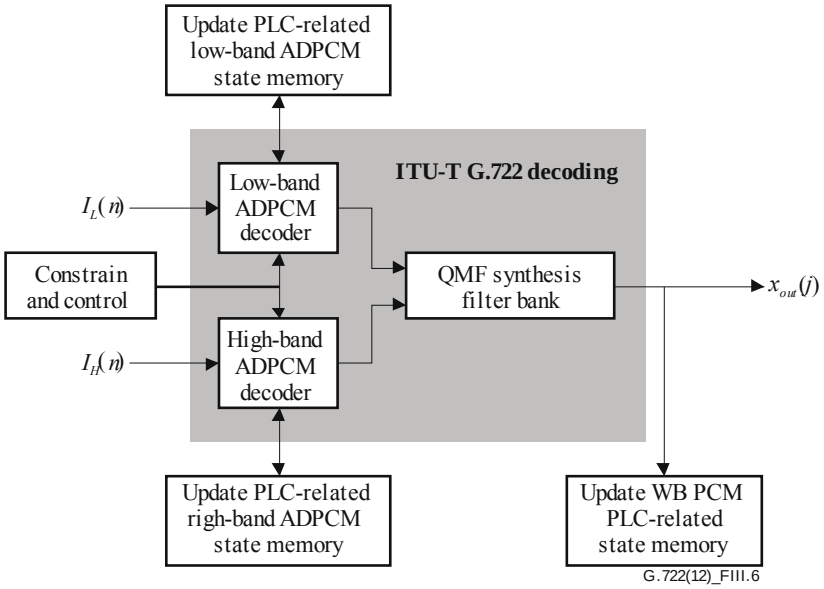


Figure III.6 – Constrained and controlled decoding in first received frames (Types 5 and 6)

The constrain and control of the sub-band ADPCM decoders are imposed for both Types 5 and 6 frames, i.e., for the first 80 ms after packet loss. Some do not extend beyond 40 ms, while others are adaptive in duration and/or degree. See clause III.8 for details.

In error-free channel conditions, the algorithm is bit-exact with ITU-T G.722. Furthermore, in error conditions, the algorithm is identical to ITU-T G.722 beyond the eighth frame after the end of packet loss, and without bit-errors convergence towards the ITU-T G.722 error-free output should be expected.

The PLC algorithm supports any packet size that is a multiple of 10 ms. For packet sizes greater than 10 ms the PLC algorithm is simply called multiple times per packet, at 10-ms intervals. Accordingly, in the following the PLC algorithm is described in this context in terms of the intrinsic frame size of 10 ms.

III.6 WB PCM PLC – Waveform extrapolation of ITU-T G.722 output

For lost frames corresponding to packet loss (Types 2, 3 and 4 frames), the WB PCM PLC scheme depicted in Figure III.3 extrapolates the ITU-T G.722 output waveform $x_{out}(j)$ of the previous frames to fill up the current frame. Such extrapolated wideband signal waveform $x_{PLC}(i)$ is then used as the output waveform of the ITU-T G.722 PLC during Types 2, 3 and 4 frames. For convenience in describing various blocks in Figure III.3, once the signal $x_{PLC}(i)$ has been calculated by the WB PCM PLC for lost frames, this signal $x_{PLC}(i)$ is considered to be written to the buffer for $x_{out}(j)$, which is the final output signal of the entire ITU-T G.722 decoder/PLC system.

Each block of Figure III.3 will now be described in more detail in the following subclauses.

III.6.1 Eighth-order LPC analysis

The eighth-order LPC analysis in Figure III.3 is performed near the end of the frame processing loop, after the current frame of signal $x_{out}(j)$ has been calculated and stored in the buffer. This eighth-order LPC analysis is the common type of autocorrelation LPC analysis, with a 10-ms asymmetric analysis window applied to the $x_{out}(j)$ signal of the current received frame. This asymmetric window is given by:

$$w(j) = \begin{cases} \frac{1}{2}\left[1 - \cos\left(\frac{(j+1)\pi}{121}\right)\right], & \text{for } j = 0, 1, 2, ..., 119 \\ \cos\left(\frac{(j-120)\pi}{80}\right), & \text{for } j = 120, 121, ..., 159 \end{cases} \quad \text{(III-1)}$$

Let $x_{out}(0)$, $x_{out}(1)$, …, $x_{out}(159)$ represent the ITU-T G.722 decoder/PLC output wideband signal samples in the current received frame. The windowing operation is performed as follows.

$$x_w(j) = x_{out}(j)w(j), \quad j = 0, 1, 2, ..., 159 \quad \text{(III-2)}$$

Next, the autocorrelation coefficients are calculated as follows.

$$r(i) = \sum_{j=i}^{159} x_w(j)x_w(j-i), \quad i = 0, 1, 2, ..., 8 \quad \text{(III-3)}$$

The spectral smoothing and white noise correction operations are then applied to the autocorrelation coefficients as follows.

$$\hat{r}(i) = \begin{cases} 1.0001 \times r(0), & i = 0 \\ r(i)e^{\frac{-(2\pi i\sigma/f_s)^2}{2}}, & i = 1, 2, ..., 8 \end{cases} \quad \text{(III-4)}$$

where f_s =16 000 is the sampling rate of the input signal and $\sigma = 40$.

Next, the Levinson-Durbin recursion is used to convert the autocorrelation coefficients $\hat{r}(i)$ to the LPC predictor coefficients $\hat{a}_i$, i = 0, 1, ..., 8. If the Levinson-Durbin recursion exits pre-maturely before the recursion is completed (for example, because the prediction residual energy $E(i)$ is less than zero), then the short-term predictor coefficients of the last frame are also used in the current frame. To do the exception handling this way, there needs to be an initial value of the $\hat{a}_i$ array. The initial value of the $\hat{a}_i$ array is set to $\hat{a}_0 = 1$ and $\hat{a}_i = 0$ for i = 1, 2, ..., 8. The Levinson-Durbin recursion algorithm is specified below.

1) If $\hat{r}(0) \le 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson-Durbin recursion.

2) $E(0) = \hat{r}(0)$

3) $k_1 = -\hat{r}(1)/\hat{r}(0)$

4) $\hat{a}_1^{(1)} = k_1$

5) $E(1) = (1 - k_1^2)E(0)$

6) If $E(1) \le 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson-Durbin recursion.

7) For i = 2, 3, 4, ..., 8, do the following:

$$k_i = \frac{-\hat{r}(i) - \sum_{j=1}^{i-1} \hat{a}_j^{i-1} \hat{r}(i-j)}{E(i-1)}$$

$$\hat{a}_i^{(i)} = k_i$$

$$\hat{a}_j^{(i)} = \hat{a}_j^{i-1} + k_i \, \hat{a}_{i-j}^{i-1}, \quad \text{for } j = 1, 2, ..., i-1$$

$$E(i) = (1 - k_i^2)E(i-1)$$

If $E(i) \leq 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson-Durbin recursion.

If the recursion exited pre-maturely, the $\hat{a}_i$ array of the last frame is used. If the recursion is completed successfully (which is normally the case), the LPC predictor coefficients are taken as:

$$\hat{a}_0 = 1 \tag{III-5}$$

$$\hat{a}_i = \hat{a}_i^{(8)}, \qquad \text{for } i = 1, 2, ..., 8 \tag{III-6}$$

By applying the bandwidth expansion operation to the coefficients derived above, the final set of LPC predictor coefficients is obtained as:

$$a_i = (0.96852)^i \hat{a}_i^{(8)}, \qquad \text{for } i = 0, 1, ..., 8 \tag{III-7}$$

III.6.2 Calculation of short-term prediction residual signal

The block in Figure III.3 with a label of "A(z)" represents a short-term linear prediction error filter, with the filter coefficients of a_i for i = 0, 1, ..., 8 as calculated above. This block is performed after the eighth-order LPC analysis is performed. This block calculates the short-term prediction residual signal $d(j)$ as follows:

$$d(j) = x_{out}(j) + \sum_{i=1}^{8} a_i \cdot x_{out}(j - \mathrm{i}), \qquad \text{for } j = 0, 1, 2, ..., 159 \tag{III-8}$$

As is conventional, the time index n of the current frame continues from the time index of the last frame. In other words, if the time index range of 0, 1, 2, ..., 159 represents the current frame, then the time index range of –160, –159, ..., –1 represents the last frame. Thus, in the equation above, if the index $(j – i)$ is negative, it just points to the signal sample near the end of the last frame.

III.6.3 Calculation of scaling factor

The block in Figure III.3 with a label of "Calculate scaling" calculates the average magnitude of the short-term prediction residual signal in the current frame. This block is performed after the short-term prediction residual signal $d(j)$ is calculated as described in clause III.6.2. This average magnitude *avm* is calculated as follows:

$$avm = \frac{1}{160} \sum_{j=0}^{159} |d(j)| \tag{III-9}$$

If the next frame is a lost frame (i.e., corresponds to a packet loss), this average magnitude *avm* may be used as a scaling factor to scale a white Gaussian noise sequence if the current frame is sufficiently unvoiced.

III.6.4 Calculation of weighted speech signal

The block in Figure III.3 with a label of "1/A(z/γ)" represents a weighted short-term synthesis filter. This block is performed after the short-term prediction residual signal $d(j)$ is calculated for the current frame. The coefficients of this filter, a'_i for i = 0, 1, ..., 8, are calculated as follows with $\gamma_1 = 0.75$.

$$a'_i = \gamma_1^i a_i, \qquad \text{for } i = 1, 2, ..., 8 \tag{III-10}$$

The short-term prediction residual signal $d(j)$ passes through this weighted short-term synthesis filter. The corresponding output weighted speech signal $xw(j)$ is calculated as:

$$xw(j) = d(j) - \sum_{i=1}^{8} a'_i \cdot xw(j-i), \qquad \text{for } j = 0, 1, 2, ..., 159 \tag{III-11}$$

III.6.5 Eight-to-one decimation

Table III.1 – Coefficients for 60th order FIR filter

Lag, i	b_i in Q15	Lag, i	b_i in Q15	Lag, i	b_i in Q15
0	1209	20	–618	40	313
1	728	21	–941	41	143
2	1120	22	–1168	42	–6
3	1460	23	–1289	43	–126
4	1845	24	–1298	44	–211
5	2202	25	–1199	45	–259
6	2533	26	–995	46	–273
7	2809	27	–701	47	–254
8	3030	28	–348	48	–210
9	3169	29	20	49	–152
10	3207	30	165	50	–89
11	3124	31	365	51	–30
12	2927	32	607	52	21
13	2631	33	782	53	58
14	2257	34	885	54	81
15	1814	35	916	55	89
16	1317	36	881	56	84
17	789	37	790	57	66
18	267	38	654	58	41
19	–211	39	490	59	17

The weighted speech signal is passed through a 60th-order minimum-phase FIR low-pass filter, and then 8:1 decimation is performed to down-sample the resulting 16-kHz low-pass filtered weighted speech signal to 2-kHz down-sampled weighted speech signal $xwd(n)$. This decimation operation is performed after the weighted speech signal $xw(j)$ is calculated. To reduce complexity, the FIR low-pass filtering operation is carried out only when a new sample of $xwd(n)$ is needed. Thus, the down-sampled weighted speech signal $xwd(n)$ is calculated as:

$$xwd(n) = \sum_{i=0}^{59} b_i \cdot xw(8n + 7 - i), \quad \text{for } n = 0, 1, 2, ..., 19 \tag{III-12}$$

where b_i, i = 0, 1, 2, ..., 59 are the filter coefficients for the 60th-order FIR low-pass filter as given in Table III.1.

III.6.6 Coarse pitch period extraction

To reduce the computational complexity, in WB PCM PLC the pitch extraction is performed in two stages:

1) determination of the coarse pitch period with a time resolution of the 2-kHz decimated signal;
2) pitch period refinement with a time resolution of the 16-kHz undecimated signal. Such pitch extraction is performed only during the received frames after the down-sampled weighted speech signal $xwd(n)$ is calculated.

This clause describes the first-stage coarse pitch period extraction algorithm, which is represented in Figure III.3 by the block labelled "Coarse Pitch". This algorithm is based on maximizing the normalized cross-correlation with some additional decision logic.

A pitch analysis window of 15 ms is used in the coarse pitch period extraction. The end of the pitch analysis window is lined up with the end of the current frame. At a sampling rate of 2 kHz, 15 ms correspond to 30 samples. Without loss of generality, let the index range of $n = 0$ to $n = 29$ corresponds to the pitch analysis window for $xwd(n)$. The coarse pitch period extraction algorithm starts by calculating the following values:

$$c(k) = \sum_{n=0}^{29} xwd(n)xwd(n-k) \tag{III-13}$$

$$E(k) = \sum_{n=0}^{29} [xwd(n-k)]^2 \tag{III-14}$$

$$c2(k) = \begin{cases} c^2(k), & \text{if } c(k) \geq 0 \\ -c^2(k), & \text{if } c(k) < 0 \end{cases} \tag{III-15}$$

for all integers from $k = MINPPD - 1$ to $k = MAXPPD + 1$, where $MINPPD = 5$ and $MAXPPD = 33$ are the minimum and maximum pitch period in the decimated domain, respectively. The coarse pitch period extraction algorithm then searches through the range of $k = MINPPD$, $MINPPD + 1$, $MINPPD + 2$, ..., $MAXPPD$ to find all local peaks of the array $\{c2(k)/E(k)\}$ for which $c(k) > 0$. (A value is characterized as a local peak if both of its adjacent values are smaller.) Let N_p denote the number of such positive local peaks. Let $k_p(j)$, $j = 1, 2, ..., N_p$ be the indices where $c2(k_p(j))/E(k_p(j))$ is a local peak and $c(k_p(j)) > 0$, and let $k_p(1) < k_p(2) < ... < k_p(N_p)$. For convenience, the term $c2(k)/E(k)$ will be referred to as the "normalized correlation square".

If $N_p = 0$, that is, if there is no positive local peak for the function $c2(k)/E(k)$, then the algorithm searches for the largest negative local peak with the largest magnitude of $|c2(k)/E(k)|$. If such a largest negative local peak is found, the corresponding index k is used as the output coarse pitch period *cpp*, and the processing of this block is terminated. If the normalized correlation square function $c2(k)/E(k)$ has neither a positive nor a negative local peak, then the output coarse pitch period is set to $cpp = MINPPD$, and the processing of this block is terminated. If $N_P = 1$, the output coarse pitch period is set to $cpp = k_p(1)$, and the processing of this block is terminated.

If there are two or more local peaks ($N_p \geq 2$), then this block uses *Algorithms 1, 2, 3* and *4* (to be described below), in that order, to determine the output coarse pitch period *cpp*. Variables calculated in one of the four algorithms will be carried over and used in later algorithms.

Algorithm 1 below is used to identify the largest quadratically interpolated peak around local peaks of the normalized correlation square $c2(k_p)/E(k_p)$. Quadratic interpolation is performed for $c(k_p)$, while linear interpolation is performed for $E(k_p)$. Such interpolation is performed with the time resolution of the undecimated 16-kHz speech signal. In the algorithm below, D denotes the decimation factor used when decimating $xw(n)$ to $xwd(n)$. Thus, $D = 8$ here.

Algorithm 1 – Find the largest quadratically interpolated peak around $c2(k_p)/E(k_p)$:

i) Set $c2max = -1$, $Emax = 1$, and $jmax = 0$.

ii) For $j = 1, 2, ..., N_p$, do the following 12 steps:

1) Set $a = 0.5\,[c(k_p(j) + 1) + c(k_p(j) - 1)] - c(k_p(j))$

2) Set $b = 0.5\,[c(k_p(j) + 1) - c(k_p(j) - 1)]$

3) Set $ji = 0$

4) Set $ei = E(k_p(j))$

5) Set $c2m = c2(k_p(j))$

6) Set $Em = E(k_p(j))$

7) If $c2(k_p(j) + 1)E(k_p(j) - 1) > c2(k_p(j) - 1)E(k_p(j) + 1)$, do the remaining part of step 7:

$\Delta = [E(k_p(j) + 1) - ei]/D$

For $k = 1, 2, ..., D/2$, do the following indented part of step 7:

$ci = a\,(k/D)^2 + b(k/D) + c(k_p(j))$

$ei \leftarrow ei + \Delta$

If $(ci)^2\,Em > (c2m)\,ei$, do the next three indented lines:

$ji = k$

$c2m = (ci)^2$

$Em = ei$

8) If $c2(k_p(j) + 1)E(k_p(j) - 1) \leq c2(k_p(j) - 1)E(k_p(j) + 1)$, do the remaining part of step 8:

$\Delta = [E(k_p(j) - 1) - ei]/D$

For $k = -1, -2, ..., -D/2$, do the following indented part of step 8:

$ci = a\,(k/D)^2 + b(k/D) + c(k_p(j))$

$ei \leftarrow ei + \Delta$

If $(ci)^2\,Em > (c2m)\,ei$, do the next three indented lines:

$ji = k$

$c2m = (ci)^2$

$Em = ei$

9) Set $lag(j) = k_p(j) + ji/D$

10) Set $c2i(j) = c2m$

11) Set $Ei(j) = Em$

12) If $c2m \times Emax > c2max \times Em$, do the following three indented lines:

$jmax = j$

$c2max = c2m$

$Emax = Em$

iii) Set the first candidate for coarse pitch period as $cpp = lag(jmax)$.

The symbol ← indicates that the parameter on the left-hand side is being updated with the value on the right-hand side.

To avoid picking a coarse pitch period that is around an integer multiple of the true coarse pitch period, a search through the time lags corresponding to the local peaks of $c2(k_p)/E(k_p)$ is performed to see if any of such time lags is close enough to the output coarse pitch period of the last frame, denoted as *cpplast*. (For the very first frame, *cpplast* is initialized to 12.) If a time lag is within 25%

of *cpplast*, it is considered close enough. For all such time lags within 25% of *cpplast*, the corresponding quadratically interpolated peak values of the normalized correlation square $c2(k_p)/E(k_p)$ are compared, and the interpolated time lag corresponding to the maximum normalized correlation square is selected for further consideration. *Algorithm 2* below performs the task described above. The interpolated arrays $c2i(j)$ and $Ei(j)$ calculated in *Algorithm 1* above are used in this algorithm.

Algorithm 2 – Find the time lag maximizing interpolated $c2(k_p)/E(k_p)$ among all time lags close to the output coarse pitch period of the last frame:

i) Set index $im = -1$

ii) Set $c2m = -1$

iii) Set $Em = 1$

iv) For $j = 1, 2, ..., N_p$, do the following:

If $| k_p(j) - cpplast | \le 0.25 \times cpplast$, do the following:

If $c2i(j) \times Em > c2m \times Ei(j)$, do the following three lines:

$$im = j$$

$$c2m = c2i(j)$$

$$Em = Ei(j)$$

Note that if there is no time lag $k_p(j)$ within 25% of *cpplast*, then the value of the index *im* will remain at –1 after *Algorithm 2* is performed. If there are one or more time lags within 25% of *cpplast*, the index *im* corresponds to the largest normalized correlation square among such time lags.

Next, *Algorithm 3* determines whether an alternative time lag in the first half of the pitch range should be chosen as the output coarse pitch period. Basically, it searches through all interpolated time lags $lag(j)$ that are less than 16, and checks whether any of them has a large enough local peak of normalized correlation square near every integer multiple of it (including itself) up to 32. If there are one or more such time lags satisfying this condition, the smallest of such qualified time lags is chosen as the output coarse pitch period.

Again, variables calculated in *Algorithms 1* and *2* above carry their final values over to *Algorithm 3* below. In the following, the parameter *MPDTH* is 0.06, and the threshold array $MPTH(k)$ is given as $MPTH(2) = 0.7$, $MPTH(3) = 0.55$, $MPTH(4) = 0.48$, $MPTH(5) = 0.37$, and $MPTH(k) = 0.30$, for $k > 5$.

Algorithm 3 – Check whether an alternative time lag in the first half of the range of the coarse pitch period should be chosen as the output coarse pitch period:

For $j = 1, 2, 3, ..., N_p$, in that order, do the following while $lag(j) < 16$:

i) If $j \ne im$, set $threshold = 0.73$; otherwise, set $threshold = 0.4$.

ii) If $c2i(j) \times Emax \le threshold \times c2max \times Ei(j)$, disqualify this j, skip step iii for this j, increment j by 1 and go back to step i.

iii) If $c2i(j) \times Emax > threshold \times c2max \times Ei(j)$, do the following:

a) For $k = 2, 3, 4, ...,$ do the following while $k \times lag(j) < 32$:

1) $s = k \times lag(j)$

2) $a = (1 - MPDTH)\, s$

3) $b = (1 + MPDTH)\, s$

4) Go through $m = j + 1, j + 2, j + 3, ..., N_p$, in that order, and see if any of the time lags $lag(m)$ is between a and b. If none of them is between a and b, disqualify this j, stop step iii, increment j by 1 and go back to step i. If there is at least one such m that satisfies $a < lag(m) \leq b$ and $c2i(m) \times Emax > MPTH(k) \times c2max \times Ei(m)$, then it is considered that a large-enough peak of the normalized correlation square is found in the neighbourhood of the k-th integer multiple of $lag(j)$; in this case, stop step iii a 4, increment k by 1, and go back to step iii a 1.

b) If step iii a is completed without stopping prematurely, that is, if there is a large enough interpolated peak of the normalized correlation square within $\pm 100 \times MPDTH$ % of every integer multiple of $lag(j)$ that is less than 32, then stop this algorithm, skip *Algorithm 4* and set $cpp = lag(j)$ as the final output coarse pitch period.

If *Algorithm 3* above is completed without finding a qualified output coarse pitch period *cpp*, then *Algorithm 4* examines the largest local peak of the normalized correlation square around the coarse pitch period of the last frame, found in *Algorithm 2* above, and makes a final decision on the output coarse pitch period *cpp*. Again, variables calculated in *Algorithms 1* and *2* above carry their final values over to *Algorithm 4* below. In the following, the parameters are $SMDTH = 0.095$ and $LPTH1 = 0.78$.

Algorithm 4 – Final decision of the output coarse pitch period

i) If $im = -1$, that is, if there is no large enough local peak of the normalized correlation square around the coarse pitch period of the last frame, then use the *cpp* calculated at the end of *Algorithm 1* as the final output coarse pitch period, and exit this algorithm.

ii) If $im = jmax$, that is, if the largest local peak of the normalized correlation square around the coarse pitch period of the last frame is also the global maximum of all interpolated peaks of the normalized correlation square within this frame, then use the *cpp* calculated at the end of *Algorithm 1* as the final output coarse pitch period, and exit this algorithm.

iii) If $im < jmax$, do the following indented part:

If $c2m \times Emax > 0.43 \times c2max \times Em$, do the following indented part of step iii:

a) If $lag(im) > MAXPPD/2$, set output $cpp = lag(im)$ and exit this algorithm.

b) Otherwise, for $k = 2, 3, 4, 5$, do the following indented part:

1) $s = lag(jmax)/k$

2) $a = (1 - SMDTH)\, s$

3) $b = (1 + SMDTH)\, s$

4) If $lag(im) > a$ and $lag(im) < b$, set output $cpp = lag(im)$ and exit this algorithm.

iv) If $im > jmax$, do the following indented part:

If $c2m \times Emax > LPTH1 \times c2max \times Em$, set output $cpp = lag(im)$ and exit this algorithm.

v) If algorithm execution proceeds to here, none of the steps above have selected a final output coarse pitch period. In this case, just accept the *cpp* calculated at the end of *Algorithm 1* as the final output coarse pitch period.

III.6.7 Pitch period refinement

The block in Figure III.3 with a label of "Refine pitch" performs the second-stage processing of the pitch period extraction algorithm by searching in the neighbourhood of the coarse pitch period in full 16-kHz time resolution using the ITU-T G.722 decoded output speech signal. This block first converts the coarse pitch period *cpp* to the undecimated signal domain by multiplying it by the decimation factor D, where D = 8. The pitch refinement analysis window size *WSZ* is chosen as the smaller of $cpp \times D$ samples and 160 samples (corresponding to 10 ms): $WSZ = \min(cpp \times D, 160)$.

Next, the lower bound of the search range is calculated as $lb = \max(MINPP, cpp \times D - 4)$, where $MINPP = 40$ samples is the minimum pitch period. The upper bound of the search range is calculated as $ub = \min(MAXPP, cpp \times D + 4)$, where $MAXPP = 265$ samples is the maximum pitch period.

This block maintains a buffer of 16-kHz ITU-T G.722 decoded speech signal $x_{out}(j)$ with a total of $XQOFF = MAXPP + 1 + FRSZ$ samples, where $FRSZ = 160$ is the frame size. The last $FRSZ$ samples of this buffer contain the ITU-T G.722 decoded speech signal of the current frame. The first $MAXPP + 1$ samples are populated with the ITU-T G.722 decoder/PLC output signal in the previous frames immediately before the current frame. The last sample of the analysis window is aligned with the last sample of the current frame. Let the index range from $j = 0$ to $j = WSZ - 1$, corresponding to the analysis window, which is the last WSZ samples in the $x_{out}(j)$ buffer, and let negative indices denote the samples prior to the analysis window. The following correlation and energy terms in the undecimated signal domain are calculated for time lags k within the search range $[lb, ub]$.

$$\tilde{c}(k) = \sum_{j=0}^{WSZ-1} x_{out}(j) x_{out}(j-k) \tag{III-16}$$

$$\tilde{E}(k) = \sum_{j=0}^{WSZ-1} x_{out}(j-k)^2 \tag{III-17}$$

The time lag $k \in [lb, ub]$ that maximizes the ratio $\tilde{c}^2(k) / \tilde{E}(k)$ is chosen as the final refined pitch period for frame erasure, or *ppfe*. That is:

$$ppfe = \arg\max_{k \in [lb,ub]} \left[\frac{\tilde{c}^2(k)}{\tilde{E}(k)} \right] \tag{III-18}$$

Next, the block labelled "Refine pitch" also calculates two more pitch-related scaling factors. The first is called *ptfe*, or pitch tap for frame erasure. It is the scaling factor used for periodic waveform extrapolation. It is calculated as the ratio of the average magnitude of the $x_{out}(j)$ signal in the analysis window and the average magnitude of the portion of the $x_{out}(j)$ signal that is *ppfe* samples earlier, with the same sign as the correlation between these two signal portions.

$$ptfe = sign(\tilde{c}(ppfe)) \left[\frac{\sum_{j=0}^{WSZ-1} |x_{out}(j)|}{\sum_{j=0}^{WSZ-1} |x_{out}(j - ppfe)|} \right] \tag{III-19}$$

In the degenerate case when $\sum_{j=0}^{WSZ-1} |x_{out}(j - ppfe)| = 0$, *ptfe* is set to 0. After such calculation of *ptfe*, the value of *ptfe* is range-bound to [–1, 1].

The second pitch-related scaling factor is called *ppt*, or pitch predictor tap. It is used for calculating the long-term filter ringing signal (to be described later). It is calculated as $ppt = 0.75 \times ptfe$.

III.6.8 Calculate mixing ratio

The block labelled as "Calculate mixing" in Figure III.3 calculates a figure of merit to determine the mixing ratio between the periodically extrapolated waveform and the filtered noise waveform during lost frames. This calculation is performed only during the very first lost frame in each occurrence of packet loss, and the resulting mixing ratio is used throughout that particular packet

loss. The figure of merit is a weighted sum of three signal features: logarithmic gain, first normalized autocorrelation, and pitch prediction gain. Each of them is calculated as follows.

Using the same indexing convention for $x_{out}(j)$ as in clause III.6.7, the energy of the $x_{out}(j)$ signal in the pitch refinement analysis window is:

$$sige = \sum_{j=0}^{WSZ-1} x_{out}{}^2(j) \tag{III-20}$$

and the base-2 logarithmic gain *lg* is calculated as:

$$lg = \begin{cases} \log_2(sige), & \text{if } sige \neq 0 \\ 0, & \text{if } sige = 0 \end{cases} \tag{III-21}$$

If $\widetilde{E}(ppfe) \neq 0$, the pitch prediction residual energy is calculated as:

$$rese = sige - \widetilde{c}^2(ppfe) / \widetilde{E}(ppfe) \tag{III-22}$$

and the pitch prediction gain *pg* is calculated as:

$$pg = \begin{cases} 10\log_{10}\left(\dfrac{sige}{rese}\right), & \text{if } rese \neq 0 \\ 20, & \text{if } rese = 0 \end{cases} \tag{III-23}$$

If $\widetilde{E}(ppfe) = 0$, set *pg* = 0. If *sige* = 0, also set *pg* = 0.

The first normalized autocorrelation ρ_1 is calculated as:

$$\rho_1 = \begin{cases} \left\lfloor \dfrac{\sum_{j=0}^{WSZ-2} x_{out}(j)x_{out}(j+1)}{sige} \right\rfloor, & \text{if } sige \neq 0 \\ 0, & \text{if } sige = 0 \end{cases} \tag{III-24}$$

After these three signal features are obtained, the figure of merit is calculated as:

$$merit = lg + pg + 12\rho_1 \tag{III-25}$$

The *merit* calculated above determines the two scaling factors G_p and G_r, which effectively determine the mixing ratio between the periodically extrapolated waveform and the filtered noise waveform. There are two thresholds used for *merit*: merit high threshold *MHI* and merit low threshold *MLO*. These thresholds are set as *MHI* = 28 and *MLO* = 20. The scaling factor G_r for the random (filtered noise) component is calculated as:

$$G_r = \frac{MHI - merit}{MHI - MLO} \tag{III-26}$$

and the scaling factor *Gp* for the periodic component is calculated as:

$$G_p = 1 - G_r \tag{III-27}$$

III.6.9 Periodic waveform extrapolation

The block labelled PWE ("Periodic waveform extrapolation") in Figure III.3 periodically extrapolates the previous output speech waveform during the lost frames if *merit* > *MLO*.

At the very first lost frame of each packet loss, the average pitch period increment per frame is calculated. A pitch period history buffer *pph*(*m*), *m* = 1, 2, ..., 5 holds the pitch period *ppfe* for the

previous five frames. The average pitch period increment is obtained as follows. Start with the immediate last frame, and calculate the pitch period increment from its preceding frame to that frame (negative value means pitch period decrement). If the pitch period increment is zero, the algorithm checks the pitch period increment at the preceding frame. This process continues until the first frame with a non-zero pitch period increment or until the fourth previous frame has been examined. If all five previous frames have identical pitch periods, the average pitch period increment is set to zero. Otherwise, if the first non-zero pitch period increment is found at the *m*-th previous frame, and if the magnitude of the pitch period increment is less than 5% of the pitch period at that frame, then the average pitch period increment *ppinc* is obtained as the pitch period increment at that frame divided by *m*, and then the resulting value is limited to the range of [–1, 2].

In the second consecutive lost frame in a packet loss, the average pitch period increment *ppinc* is added to the pitch period *ppfe*, and the resulting number is rounded to the nearest integer and then limited to the range of [*MINPP*, *MAXPP*].

If the current frame is the very first lost frame of a packet loss, a so-called "ringing signal" is calculated for use in overlap-add to ensure smooth waveform transition at the beginning of the frame. The overlap-add length for the ringing signal and the periodically extrapolated waveform is 20 samples for the first lost frame. Let the index range of j = 0, 1, 2, ..., 19 correspond to the first 20 samples of the current first lost frame, which is the overlap-add period, and let the negative indices correspond to previous frames. The long-term ringing signal is obtained as a scaled version of the short-term prediction residual signal that is one pitch period earlier than the overlap-add period.

$$ltring(j) = x_{out}(j - ppfe) + \sum_{i=1}^{8} a_i \cdot x_{out}(j - ppfe - i), \quad \text{for } j = 0, 1, 2, ..., 19 \qquad \text{(III-28)}$$

After these 20 samples of *ltring*(j) are calculated, they are further scaled by the scaling factor *ppt* calculated in clause III.6.7.

$$ltring(j) \leftarrow ppt \cdot ltring(j), \quad \text{for } j = 0, 1, 2, ..., 19 \qquad \text{(III-29)}$$

With the filter memory *ring*(j), j = –8, –7, ..., –1 initialized to the last 8 samples of the $x_{out}(j)$ signal in the last frame, the final ringing signal is obtained as:

$$ring(j) = ltring(j) - \sum_{i=1}^{8} a_i \cdot ring(j - i), \quad \text{for } j = 0, 1, 2, ..., 19 \qquad \text{(III-30)}$$

Let the index range of j = 0, 1, 2, ..., 159 correspond to the current first lost frame, and the index range of j = 160, 161, 162, ..., 209 correspond to the first 50 samples of the next frame. Furthermore, let *wi*(j) and *wo*(j), j = 0, 1, ..., 19, be the triangular fade-in and fade-out windows, respectively, so that *wi*(j) + *wo*(j) = 1. Then, the periodic waveform extrapolation is performed in two steps as follows.

Step 1:

$$x_{out}(j) = wi(j) \cdot ptfe \cdot x_{out}(n - ppfe) + wo(j) \cdot ring(j), \quad \text{for } j = 0, 1, 2, ..., 19 \qquad \text{(III-31)}$$

Step 2:

$$x_{out}(j) = ptfe \cdot x_{out}(j - ppfe), \quad \text{for } j = 20, 21, 22, ..., 209 \qquad \text{(III-32)}$$

III.6.10 Normalized noise generator

If *merit* < *MHI*, the block labelled "Noise generator (normalized)" in Figure III.3 generates a sequence of white Gaussian random noise with an average magnitude of unity. To save computational complexity, the white Gaussian random noise is pre-calculated and stored in a table. To avoid using a very long table, but without repeating the same noise pattern due to a short table, a

special indexing scheme is used. In this scheme, the white Gaussian noise table *wn*(*j*) has 127 entries, and the scaled version of the output of this noise generator block is:

$$wgn(j) = avm \times wn(\mathrm{mod}(cfecount \times j, 127)), \quad \text{for } j = 0, 1, 2, ..., 209 \qquad \text{(III-33)}$$

where *cfecount* is the frame counter with *cfecount* = *k* for the *k*-th consecutive lost frame into the current packet loss, and $\mathrm{mod}(m,127) = m - 127 \times \lfloor m/127 \rfloor$ is the modulo operation.

III.6.11 Filtering of noise sequence

The block in Figure III.3 with a label of "1/A(z)" represents a short-term synthesis filter. If *merit* < *MHI*, it filters the scaled white Gaussian noise to give it the same spectral envelope as that of the $x_{out}(j)$ signal in the last frame. The filtered noise $fn(j)$ is obtained as:

$$fn(j) = wgn(j) - \sum_{i=1}^{8} a_i \cdot fn(j-i), \quad \text{for } j = 0, 1, 2, ..., 209 \qquad \text{(III-34)}$$

III.6.12 Mixing of periodic and random components

If *merit* > *MHI*, only the periodically extrapolated waveform $x_{out}(j)$ calculated in clause III.6.9 is used as the output of the WB PCM PLC. If *merit* < *MLO*, only the filtered noise signal $fn(j)$ is used as the output of the WB PCM PLC. If $MLO \le merit \le MHI$, then the two components are mixed as:

$$x_{out}(j) \leftarrow G_p \cdot x_{out}(j) + G_r \cdot fn(j) \quad \text{for } j = 0, 1, 2, ..., 209 \qquad \text{(III-35)}$$

The first 40 extra samples of extrapolated $x_{out}(j)$ signal for j = 160, 161, 162, ..., 199 will become the ringing signal $ring(j)$, j = 0, 1, 2, ..., 39 of the next frame. If the next frame is again a lost frame, only the first 20 samples of this ringing signal will be used for overlap-add. If the next frame is a received frame, then all 40 samples of this ringing signal will be used for overlap-add.

III.6.13 Conditional ramp down

If the packet loss lasts 20 ms or less, the $x_{out}(j)$ signal calculated in clause III.6.12 is used as the WB PCM PLC output signal. If the packet loss lasts longer than 60 ms, the WB PCM PLC output signal is completely muted. If the packet loss lasts longer than 20 ms but no more than 60 ms, the $x_{out}(j)$ signal calculated in clause III.6.12 is linearly ramped down (attenuate toward zero in a linear fashion). This conditional ramp down is performed as specified in the following algorithm during the lost frames when *cfecount* > 2. The array *gawd*() is given by {–52, –69, –104, –207} in Q15 format. Again, the index range of j = 0, 1, 2, ..., 159 corresponds to the current frame of $x_{out}(j)$.

If *cfecount* ≤ 6, do the next nine indented lines:

delta = *gawd*(*cfecount* – 3)

gaw = 1

For j = 0, 1, 2, ..., 159, do the next two lines:

$x_{out}(j) = gaw \cdot x_{out}(j)$

gaw = *gaw* + *delta*

If *cfecount* < 6, do the next three lines:

For j = 160, 161, 162, ..., 209, do the next two lines:

$x_{out}(j) = gaw \cdot x_{out}(j)$

gaw = *gaw* + *delta*

Otherwise (if *cfecount* > 6), set $x_{out}(j) = 0$ for j = 0, 1, 2, ..., 209.

III.6.14 Overlap-add in the first received frame

In Type 5 frames, the output from the ITU-T G.722 decoder $x_{out}(j)$ is overlap-added with the ringing signal from the last lost frame, $ring(j)$ (see clause III.6.12):

$$x_{out}(j) = w_i(j) \cdot x_{out}(j) + w_o(j) \cdot ring(j) \qquad j = 0 \ldots L_{OLA} - 1 \tag{III-36}$$

where:

$$L_{OLA} = \begin{cases} 8 & \text{if } G_p = 0 \\ 40 & \text{otherwise} \end{cases} \tag{III-37}$$

III.7 Re-encoding of PLC output

In order to update the memory and parameters of the ITU-T G.722 ADPCM decoders during lost frames (frame Types 2, 3 and 4), conceptually, the PLC output is passed through the ITU-T G.722 encoder. This involves:

1) passing the PLC output through the QMF analysis filter bank;

2) encoding the low-band sub-band signal with the low-band ADPCM encoder; and

3) encoding the high-band sub-band signal with the high-band ADPCM encoder.

To save complexity, simplified ADPCM sub-band encoders are designed. Figure III.4 shows the block diagram of steps 1 through 3.

III.7.1 Passing the PLC output through the QMF analysis filter bank

The memory of the QMF analysis filter bank is initialized to provide sub-band signals that are continuous with the decoded sub-band signals. The first 22 samples of the WB PCM PLC output constitutes the filter memory, and the sub-band signals are calculated according to:

$$x_L(n) = \sum_{i=0}^{11} h_{2i} \cdot x_{PLC}(23 + j - 2i) + \sum_{i=0}^{11} h_{2i+1} \cdot x_{PLC}(22 + j - 2i) \tag{III-38}$$

and:

$$x_H(n) = \sum_{i=0}^{11} h_{2i} \cdot x_{PLC}(23 + j - 2i) - \sum_{i=0}^{11} h_{2i+1} \cdot x_{PLC}(22 + j - 2i) \tag{III-39}$$

where $x_{PLC}(0)$ corresponds to the first sample of the 16-kHz WB PCM PLC output of the current frame, $x_L(n = 0)$ and $x_H(n = 0)$ correspond to the first samples of the 8-kHz low-band and high-band sub-band signals, respectively, of the current frame. The filtering is identical to the transmit QMF of the ITU-T G.722 encoder except for the extra 22 samples of offset, and that the WB PCM PLC output (as opposed to the input) is passed to the filter bank. Furthermore, in order to generate a full frame (80 samples ~ 10 ms) of sub-band signals, the WB PCM PLC needs to extend beyond the current frame by 22 samples and generate (182 samples ~ 11.375 ms). Sub-band signals $x_L(n)$, $n = 0, 1, ..., 79$, and $x_H(n)$, $n = 0, 1, ..., 79$ are generated according to Equations III-38 and III-39, respectively.

III.7.2 Re-encoding of low-band signal

The low-band signal, $x_L(n)$, is encoded with a simplified low-band ADPCM encoder shown in Figure III.7. The inverse quantizer has been eliminated, and the unquantized prediction error replaces the quantized prediction error. Furthermore, since the update of the adaptive quantizer is only based on an 8-member subset of the 64-member set represented by the 6-bit low-band encoder index, $I_L(n)$, the prediction error is only quantized to the 8-member set. This provides identical update of the adaptive quantizer, yet simplifies the quantization. Table III.2 lists the decision levels,

output code, and multipliers for the 8-level simplified quantizer based on the absolute value of $e_L(n)$.

Table III.2 – Decision levels, output code, and multipliers for the 8-level simplified quantizer

m_L	Lower threshold	Upper threshold	I_L	Multiplier, W_L
1	0.00000	0.14103	3c	–0.02930
2	0.14103	0.45482	38	–0.01465
3	0.45482	0.82335	34	0.02832
4	0.82335	1.26989	30	0.08398
5	1.26989	1.83683	2c	0.16309
6	1.83683	2.61482	28	0.26270
7	2.61482	3.86796	24	0.58496
8	3.86796	∞	20	1.48535

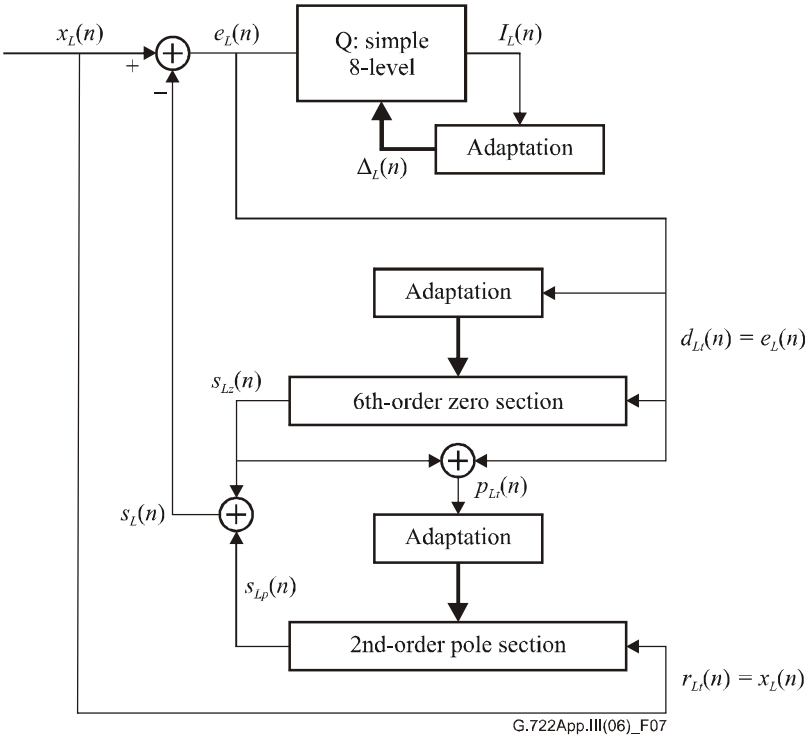


Figure III.7 – Low-band ADPCM sub-band re-encoding

The entities of Figure III.7 are calculated like their equivalents in the ITU-T G.722 low-band ADPCM sub-band encoder:

$$s_{Lz}(n) = \sum_{i=1}^{6} b_{L,i}(n-1) \cdot e_L(n-i) \tag{III-40}$$

$$s_{Lp}(n) = \sum_{i=1}^{2} a_{L,i}(n-1) \cdot x_L(n-i) \tag{III-41}$$

$$s_L(n) = s_{Lp}(n) + s_{Lz}(n) \tag{III-42}$$

$$e_L(n) = x_L(n) - s_L(n) \tag{III-43}$$

$$p_{Lt}(n) = s_{Lz}(n) + e_L(n) \tag{III-44}$$

The adaptive quantizer is updated exactly as in the ITU-T G.722 encoder, see clause 3.5 of [ITU-T G.722]. The adaptation of the zero and pole sections take place as in the ITU-T G.722 encoder, described in clauses 3.6.3 and 3.6.4 of [ITU-T G.722].

The low-band decoder is automatically reset after 60 ms of frame loss, but it may reset adaptively as early as 30 ms into frame loss. During re-encoding of the low-band signal, the properties of the partially reconstructed signal, $p_{Lt}(n)$, are monitored and control the adaptive reset of the low-band ADPCM decoder. The sign of $p_{Lt}(n)$ is monitored over the entire loss, and hence is reset to zero at the first lost frame:

$$\text{sgn}[p_{Lt}(n)] = \begin{cases} \text{sgn}[p_{Lt}(n-1)] + 1 & p_{Lt}(n) > 0 \\ \text{sgn}[p_{Lt}(n-1)] & p_{Lt}(n) = 0 \\ \text{sgn}[p_{Lt}(n-1)] - 1 & p_{Lt}(n) < 0 \end{cases} \tag{III-45}$$

The property of $p_{Lt}(n)$ as a constant signal is monitored on a frame basis for lost frames, and hence is reset at the beginning of every lost frame. It is updated as:

$$\text{cnst}[p_{Lt}(n)] = \begin{cases} \text{cnst}[p_{Lt}(n-1)] + 1 & p_{Lt}(n) = p_{Lt}(n-1) \\ \text{cnst}[p_{Lt}(n-1)] & p_{Lt}(n) \neq p_{Lt}(n-1) \end{cases} \tag{III-46}$$

At the end of lost frames 3 through 5, the low-band decoder is reset if the following condition is satisfied:

$$\left| \frac{\text{sgn}[p_{Lt}(n)]}{N_{lost}} \right| > 36 \quad \text{OR} \quad \text{cnst}[p_{Lt}(n)] > 40 \tag{III-47}$$

where N_{lost} is the number of lost frames, i.e., 3, 4, or 5.

III.7.3 Re-encoding of high-band signal

The high-band signal, $x_H(n)$, is encoded with a simplified high-band ADPCM encoder as shown in Figure III.8. The adaptive quantizer has been eliminated as the algorithm overwrites the log scale factor at the first received frame with a moving average prior to the loss, and hence, does not need the high-band re-encoded log scale factor. The quantized prediction error of the high-band ADPCM encoder is substituted with the unquantized prediction error.

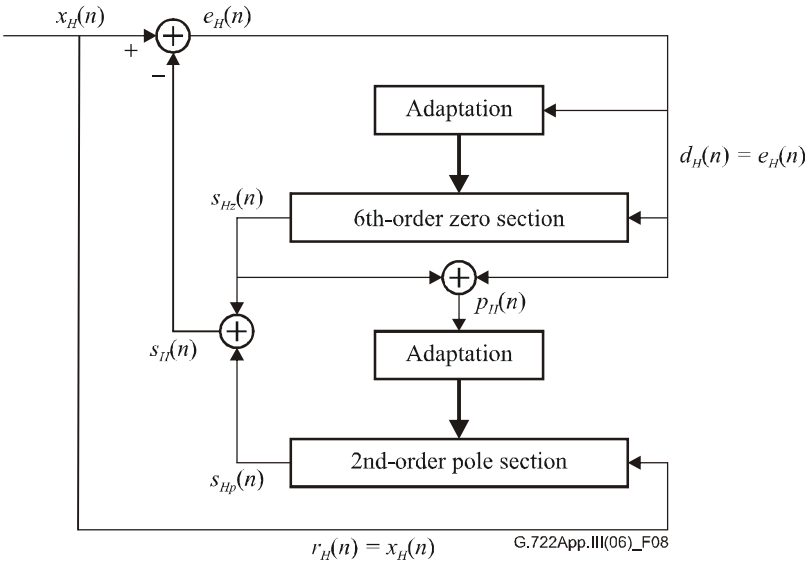


Figure III.8 – High-band ADPCM sub-band re-encoding

The entities of Figure III.8 are calculated like their equivalents in the ITU-T G.722 high-band ADPCM sub-band encoder:

$$s_{Hz}(n) = \sum_{i=1}^{6} b_{H,i}(n-1) \cdot e_H(n-i) \tag{III-48}$$

$$s_{Hp}(n) = \sum_{i=1}^{2} a_{H,i}(n-1) \cdot x_H(n-i) \tag{III-49}$$

$$s_H(n) = s_{Hp}(n) + s_{Hz}(n) \tag{III-50}$$

$$e_H(n) = x_H(n) - s_H(n) \tag{III-51}$$

$$p_H(n) = s_{Hz}(n) + e_H(n) \tag{III-52}$$

Adaptation of the zero and pole sections takes place as in the ITU-T G.722 encoder, described in clauses 3.6.3 and 3.6.4 of [ITU-T G.722].

In the same way as with low-band re-encoding, the high-band decoder is automatically reset after 60 ms of frame loss, but it may reset adaptively as early as 30 ms into frame loss. During re-encoding of the high-band signal, the properties of the partially reconstructed signal, $p_H(n)$, are monitored and control the adaptive reset of the high-band ADPCM decoder. The sign of $p_H(n)$ is monitored over the entire loss, and hence is reset to zero at the first lost frame:

$$\mathrm{sgn}[p_H(n)] = \begin{cases} \mathrm{sgn}[p_H(n-1)] + 1 & p_H(n) > 0 \\ \mathrm{sgn}[p_H(n-1)] & p_H(n) = 0 \\ \mathrm{sgn}[p_H(n-1)] - 1 & p_H(n) < 0 \end{cases} \tag{III-53}$$

The property of $p_H(n)$ as a constant signal is monitored on a frame basis for lost frames, and hence is reset at the beginning of every lost frame. It is updated as:

$$\text{cnst}[p_H(n)] = \begin{cases} \text{cnst}[p_H(n-1)]+1 & p_H(n) = p_H(n-1) \\ \text{cnst}[p_H(n-1)] & p_H(n) \neq p_H(n-1) \end{cases} \quad \text{(III-54)}$$

At the end of lost frames 3 through 5, the high-band decoder is reset if the following condition is satisfied:

$$\left| \frac{\text{sgn}[p_H(n)]}{N_{lost}} \right| > 36 \quad \text{OR} \quad \text{cnst}[p_H(n)] > 40 \quad \text{(III-55)}$$

III.8 Monitoring signal characteristics and their use for PLC

III.8.1 Low-band log scale factor

Characteristics of the low-band log scale factor, $\nabla_L(n)$, are updated during received frames and used at the first received frame after frame loss to adaptively set the state of the adaptive quantizer for the scale factor. A measure of the stationarity of the low-band log scale factor is derived and used to determine proper resetting of the state.

III.8.1.1 Stationarity of low-band log scale factor

The stationarity of the low-band log scale factor, $\nabla_L(n)$, is calculated and updated during received frames. It is based on a first-order moving average, $\nabla_{L,m1}(n)$, of $\nabla_L(n)$ with constant leakage:

$$\nabla_{L,m1}(n) = 7/8 \cdot \nabla_{L,m1}(n-1) + 1/8 \cdot \nabla_L(n) \quad \text{(III-56)}$$

A measure of the tracking, $\nabla_{L,trck}(n)$, of the first-order moving average is calculated as:

$$\nabla_{L,trck}(n) = 127/128 \cdot \nabla_{L,trck}(n-1) + 1/128 \cdot \left| \nabla_{L,m1}(n) - \nabla_{L,m1}(n-1) \right| \quad \text{(III-57)}$$

A second-order moving average, $\nabla_{L,m2}(n)$, with adaptive leakage is calculated according to Equation III-58.

$$\nabla_{L,m2}(n) = \begin{cases} 7/8 \cdot \nabla_{L,m2}(n-1) + 1/8 \cdot \nabla_{L,m1}(n) & \nabla_{L,trck}(n) < 3277 \\ 3/4 \cdot \nabla_{L,m2}(n-1) + 1/4 \cdot \nabla_{L,m1}(n) & 3277 \le \nabla_{L,trck}(n) < 6554 \\ 1/2 \cdot \nabla_{L,m2}(n-1) + 1/2 \cdot \nabla_{L,m1}(n) & 6554 \le \nabla_{L,trck}(n) < 9830 \\ \nabla_{L,m2}(n) = \nabla_{L,m1}(n) & 9830 \le \nabla_{L,trck}(n) \end{cases} \quad \text{(III-58)}$$

The stationarity of the low-band log scale factor is measured as a degree of change according to:

$$\nabla_{L,chng}(n) = 127/128 \cdot \nabla_{L,chng}(n-1) + 1/128 \cdot 256 \cdot \left| \nabla_{L,m2}(n) - \nabla_{L,m2}(n-1) \right| \quad \text{(III-59)}$$

During lost frames, there is no update, i.e.:

$$\begin{aligned} \nabla_{L,m1}(n) &= \nabla_{L,m1}(n-1) \\ \nabla_{L,trck}(n) &= \nabla_{L,trck}(n-1) \\ \nabla_{L,m2}(n) &= \nabla_{L,m2}(n-1) \\ \nabla_{L,chng}(n) &= \nabla_{L,chng}(n-1) \end{aligned} \quad \text{(III-60)}$$

III.8.1.2 Resetting of log scale factor of the low-band adaptive quantizer

At the first received frame after recovery from frame loss, the low-band log scale factor is reset (overwritten) adaptively depending on the stationarity prior to the frame loss:

$$\nabla_L(n-1) \leftarrow \begin{cases} \nabla_{L,m2}(n-1) & \nabla_{L,chng}(n-1) < 6554 \\ \frac{\nabla_L(n-1)}{3276}\left[\nabla_{L,chng}(n-1) - 6554\right] + \frac{\nabla_{L,m2}(n-1)}{3276}\left[9830 - \nabla_{L,chng}(n-1)\right] & 6554 \le \nabla_{L,chng}(n-1) \le 9830 \\ \nabla_L(n-1) & 9830 < \nabla_{L,chng}(n-1) \end{cases} \tag{III-61}$$

III.8.2 High-band log scale factor

Characteristics of the high-band log scale factor, $\nabla_H(n)$, are updated during received frames and used at the first received frame following frame loss to set the state of the adaptive quantization scale factor. Furthermore, the characteristics adaptively control the convergence of the high-band log scale factor after frame loss.

III.8.2.1 Moving average and stationarity of high-band log scale factor

The tracking of $\nabla_H(n)$ is calculated according to:

$$\nabla_{H,trck}(n) = 0.97 \cdot \nabla_{H,trck}(n-1) + 0.03 \cdot \left|\nabla_{H,m}(n-1) - \nabla_H(n)\right| \tag{III-62}$$

Based on the tracking, the moving average is calculated with adaptive leakage as:

$$\nabla_{H,m}(n) = \begin{cases} 255/256 \cdot \nabla_{H,m}(n-1) + 1/256 \cdot \nabla_H(n) & \left|\nabla_{H,trck}(n)\right| < 1638 \\ 127/128 \cdot \nabla_{H,m}(n-1) + 1/128 \cdot \nabla_H(n) & 1638 \le \left|\nabla_{H,trck}(n)\right| < 3277 \\ 63/64 \cdot \nabla_{H,m}(n-1) + 1/64 \cdot \nabla_H(n) & 3277 \le \left|\nabla_{H,trck}(n)\right| < 4915 \\ 31/32 \cdot \nabla_{H,m}(n-1) + 1/32 \cdot \nabla_H(n) & 4915 \le \left|\nabla_{H,trck}(n)\right| \end{cases} \tag{III-63}$$

The moving average is used for resetting the high-band log scale factor at the first received frame as described in clause III.8.2.2.

A measure of the stationarity of the high-band log scale factor is calculated from the moving average above according to:

$$\nabla_{H,chng}(n) = 127/128 \cdot \nabla_{H,chng}(n-1) + 1/128 \cdot 256 \cdot \left|\nabla_{H,m}(n) - \nabla_{H,m}(n-1)\right| \tag{III-64}$$

The measure of stationarity is used to control re-convergence of $\nabla_H(n)$ after frame loss: see clause III.8.2.3.

During lost frames there is no update, i.e.:

$$\begin{aligned} \nabla_{H,trck}(n) &= \nabla_{H,trck}(n-1) \\ \nabla_{H,m}(n) &= \nabla_{H,m}(n-1) \\ \nabla_{H,chng}(n) &= \nabla_{H,chng}(n-1) \end{aligned} \tag{III-65}$$

III.8.2.2 Resetting of log scale factor of the high-band adaptive quantizer

At the first received frame, the high-band log scale factor is reset to the running mean of received frames prior to the loss:

$$\nabla_H(n-1) \leftarrow \nabla_{H,m}(n-1) \tag{III-66}$$

III.8.2.3 Convergence of log scale factor of the high-band adaptive quantizer

The convergence of the high-band log-scale factor after frame loss is controlled by the measure of stationarity, $\nabla_{H,chng}(n)$, prior to the frame loss. For stationary cases, an adaptive low-pass filter is applied to $\nabla_H(n)$ after packet loss. The low-pass filter is applied over either 0 ms, 40 ms, or 80 ms, during which the degree of low-pass filtering is gradually reduced. The duration in samples, N_{LP,∇_H}, is determined according to:

$$N_{LP,\nabla_H} = \begin{cases} 640 & \nabla_{H,chng} < 819 \\ 320 & \nabla_{H,chng} < 1311 \\ 0 & \nabla_{H,chng} \geq 1311 \end{cases} \quad \text{(III-67)}$$

The low-pass filtering is given by:

$$\nabla_{H,LP}(n) = \alpha_{LP}(n)\nabla_{H,LP}(n-1) + (1-\alpha_{LP}(n))\nabla_H(n) \quad \text{(III-68)}$$

where the coefficient is given by:

$$\alpha_{LP}(n) = 1 - \left(\frac{n+1}{N_{LP,\nabla_H}+1}\right)^2 \qquad n = 0, 1 \ldots, N_{LP,\nabla_H} - 1 \quad \text{(III-69)}$$

Hence, the low-pass filtering reduces sample by sample with the time *n*. The low-pass filtered log scale factor simply replaces the regular log scale factor during the N_{LP,∇_H} samples.

III.8.3 Low-band pole section

An entity referred to as the stability margin (of the pole section) is updated during received frames for the low-band ADPCM decoder and used to constrain the pole section following frame loss.

III.8.3.1 Stability margin of low-band pole section

The stability margin of the low-band pole section is defined as:

$$\beta_L(n) = 1 - |a_{L,1}(n)| - a_{L,2}(n) \quad \text{(III-70)}$$

where $a_{L,1}(n)$ and $a_{L,2}(n)$ are the two pole coefficients. A moving average of the stability margin is updated according to:

$$\beta_{L,MA}(n) = 15/16 \cdot \beta_{L,MA}(n-1) + 1/16 \cdot \beta_L(n) \quad \text{(III-71)}$$

during received frames.

During lost frames, the moving average is not updated:

$$\beta_{L,MA}(n) = \beta_{L,MA}(n-1) \quad \text{(III-72)}$$

III.8.3.2 Constraint on low-band pole section

During regular ITU-T G.722 low-band (and high-band) ADPCM encoding and decoding, a minimum stability margin of $\beta_{L,min} = 1/16$ is maintained. During the first 40 ms after a frame loss, an increased minimum stability margin is maintained for the low-band ADPCM decoder. It is a function of both the time since the frame loss and the moving average of the stability margin.

For the first three 10-ms frames, a minimum stability margin of:

$$\beta_{L,\min} = \min\{3/16, \beta_{L,MA}(n-1)\} \tag{III-73}$$

is set at the frame boundary and imposed throughout the frame. At the frame boundary into the fourth 10-ms frame, a minimum stability margin of:

$$\beta_{L,\min} = \min\left\{2/16, \frac{1/16 + \beta_{L,MA}(n-1)}{2}\right\} \tag{III-74}$$

is imposed, while the regular minimum stability margin of $\beta_{L,min}$ = 1/16 is imposed for all other frames.

III.8.4 High-band partially reconstructed and reconstructed signals

During all frames, both lost and received, high-pass filtered versions of the high-band partially reconstructed signal, $p_H(n)$, and reconstructed signal, $r_H(n)$, are maintained:

$$p_{H,HP}(n) = 0.97\left[p_H(n) - p_H(n-1) + p_{H,HP}(n-1)\right] \tag{III-75}$$

$$r_{H,HP}(n) = 0.97\left[r_H(n) - r_H(n-1) + r_{H,HP}(n-1)\right] \tag{III-76}$$

This corresponds to a 3-dB cut-off at about 40 Hz, basically DC removal.

During the first 40 ms after frame loss, the regular partially reconstructed signal and regular constructed signal are substituted with their respective high-pass filtered versions for the purpose of high-band pole section adaptation and high-band reconstructed output, respectively.

III.9 Time lag computation

The re-phasing (clause III.10) and time-warping (clause III.11) techniques require that the number of samples by which the concealment waveform $x_{PLC}(j)$ and the signal in the first received frame are misaligned be known.

III.9.1 Low-complexity estimate of the lower sub-band reconstructed signal

The signal used to calculate the time lag in the first received frame is obtained by filtering the lower sub-band truncated difference signal, $d_{Lt}(n)$ (see Equation 3-11 of [ITU-T G.722]) with the pole-zero filter coefficients ($a_{Lpwe,i}(159)$, $b_{Lpwe,i}(159)$) and other required state information obtained from $STATE_{159}$ (see clause III.10.1):

$$r_{Le}(n) = \sum_{i=1}^{2} a_{Lpwe,i}(159) \cdot r_{Le}(n-i) + \sum_{i=1}^{6} b_{Lpwe,i}(159) \cdot d_{Lt}(n-i) + d_{Lt}(n) \quad , n = 0, 1, ..., 79 \tag{III-77}$$

III.9.2 Determination of re-phasing and time-warping requirement

If the last received frame is unvoiced, as indicated by the value of merit, the time lag T_L is set to zero:

$$IF\ merit \le MLO, T_L = 0 \tag{III-78}$$

Likewise, if the first received frame is unvoiced, as indicated by the normalized first autocorrelation coefficient:

$$r(1) = \frac{\sum_{n=0}^{78} r_{Le}(n) \cdot r_{Le}(n)}{\sum_{n=0}^{78} r_{Le}(n) \cdot r_{Le}(n+1)} \tag{III-79}$$

the time lag is set to zero:

$$IF\ r(1) < 0.125, T_L = 0 \tag{III-80}$$

Otherwise, the time lag is computed as explained in the following clause.

III.9.3 Computation of the time lag

Computation of the time lag involves the following steps:

1) generation of the extrapolated signal;

2) coarse time lag search;

3) refined time lag search.

These steps are described in the following subclauses.

III.9.3.1 Generation of the extrapolated signal

The time lag represents the misalignment between $x_{PLC}(j)$ and $r_{Le}(n)$. To compute the misalignment, $x_{PLC}(j)$ is extended into the first received frame and a normalized cross-correlation function is maximized. This clause describes how $x_{PLC}(j)$ is extrapolated and specifies the length of signal that is needed. As in clause III.6, it is assumed that $x_{PLC}(j)$ is copied into the $x_{out}(j)$ buffer. Since this is a Type-5 frame (first received frame), the assumed correspondence is:

$$x_{out}(j-160) = x_{PLC}(j), \quad j = 0, 1, ..., 159 \tag{III-81}$$

The range over which the correlation is searched is given by:

$$\Delta_{TL} = \min(\lfloor ppfe \cdot 0.5 + 0.5 \rfloor + 3, \Delta_{TLMAX}) \tag{III-82}$$

where Δ_{TLMAX} = 28 and *ppfe* is the pitch period for periodic waveform extrapolation used in the generation of $x_{PLC}(j)$.

The window size (at 16-kHz sampling) for the lag search is given by:

$$LSW_{16k} = \begin{cases} 80 & \lfloor ppfe \cdot 1.5 + 0.5 \rfloor < 80 \\ 160 & \lfloor ppfe \cdot 1.5 + 0.5 \rfloor > 160 \\ \lfloor ppfe \cdot 1.5 + 0.5 \rfloor & otherwise \end{cases} \tag{III-83}$$

It is useful to specify the lag search window, *LSW*, at 8-kHz sampling as:

$$LSW = \lfloor LSW_{16k} \cdot 0.5 \rfloor \tag{III-84}$$

Given the above, the total length of the extrapolated signal that needs to be derived from $x_{PLC}(j)$ is given by:

$$L = 2 \cdot (LSW + \Delta_{TL}) \tag{III-85}$$

The starting position of the extrapolated signal in relation to the first sample in the received frame is:

$$D = 12 - \Delta_{TL} \tag{III-86}$$

The extrapolated signal *es(j)* is constructed according to the following:

If $D < 0$

$es(j) = x_{out}(D + j)$ $\quad j = 0, 1, ..., -D-1$

If $(L + D \le ppfe)$

$es(j) = x_{out}(-ppfe + D + j)$ $\quad j = -D, -D+1, ..., L-1$

Else

$es(j) = x_{out}(-ppfe + D + j)$ $\quad j = -D, -D+1, ..., ppfe - D - 1$

$es(j) = es(j - ppfe)$ $\quad j = ppfe - D, ppfe - D + 1, ..., L - 1$

Else

$ovs = ppfe \cdot \lceil D / ppfe \rceil - D$

If $(ovs \geq L)$

$es(j) = x_{out}(-ovs + j)$ $\quad j = 0, 1, ..., L - 1$

Else

If $(ovs > 0)$

$es(j) = x_{out}(-ovs + j)$ $\quad j = 0, 1, ..., ovs - 1$

If $(L - ovs \leq ppfe)$

$es(j) = x_{out}(-ovs - ppfe + j)$ $\quad j = ovs, ovs + 1, ..., L - 1$

Else

$es(j) = x_{out}(-ovs - ppfe + j)$ $\quad j = ovs, ovs + 1, ..., ovs + ppfe - 1$

$es(j) = es(j - ppfe)$ $\quad j = ovs + ppfe, ovs + ppfe + 1, ..., L - 1$

III.9.3.2 Coarse time lag search

A coarsely estimated time lag, T_{LSUB}, is first computed by searching for the peak of the sub-sampled normalized cross-correlation function $R_{SUB}(k)$:

$$R_{SUB}(k) = \frac{\sum_{i=0}^{LSW/2-1} es(4i - k + \Delta_{TL}) \cdot r_{Le}(2i)}{\sqrt{\sum_{i=0}^{LSW/2-1} es^2(4i - k + \Delta_{TL}) \sum_{i=0}^{LSW-1} r_{Le}{}^2(2i)}}, \quad k = -\Delta_{TL}, -\Delta_{TL} + 4, -\Delta_{TL} + 8, \ldots, \Delta_{TL} \qquad \text{(III-87)}$$

To avoid searching out of bounds during refinement, T_{LSUB} may be adjusted as follows:

$$\text{If } (T_{LSUB} > \Delta_{TLMAX} - 4) \qquad T_{LSUB} = \Delta_{TLMAX} - 4 \qquad \text{(III-88)}$$

$$\text{If } (T_{LSUB} < -\Delta_{TLMAX} + 4) \qquad T_{LSUB} = -\Delta_{TLMAX} + 4 \qquad \text{(III-89)}$$

III.9.3.3 Refined time lag search

The search is then refined to give the time lag, T_L, by searching for the peak of $R(k)$ given by:

$$R(k) = \frac{\sum_{i=0}^{LSW-1} es(2i - k + \Delta_{TL}) \cdot r_{Le}(i)}{\sqrt{\sum_{i=0}^{LSW-1} es^2(2i - k + \Delta_{TL}) \sum_{i=0}^{LSW-1} r_{Le}{}^2(i)}}, \quad k = -4 + T_{LSUB}, -2 + T_{LSUB}, ..., 4 + T_{LSUB} \qquad \text{(III-90)}$$

Finally, the following conditions are checked:

If:

$$\sum_{i=0}^{LSW-1} r_{Le}{}^2(i) = 0 \qquad \text{(III-91)}$$

or:

$$\sum_{i=0}^{LSW-1} es(2i - T_L + \Delta_{TL}) \cdot r_{Le}(i) \leq 0.25 \cdot \sqrt{\sum_{i=0}^{LSW-1} r_{Le}{}^2(i)} \qquad \text{(III-92)}$$

or:

$$(T_L > \Delta_{TLMAX} - 2) \| (T_L < -\Delta_{TLMAX} + 2) \tag{III-93}$$

then:

$$T_L = 0$$

III.10 Re-phasing

Re-phasing is the process of setting the internal states to a point in time where the lost frame concealment waveform $x_{PLC}(j)$ is in phase with the last input signal sample immediately before the first received frame. The re-phasing can be broken down into the following steps:

1) Store intermediate ITU-T G.722 states during re-encoding of lost frames.

2) Adjust re-encoding according to the time lag.

3) Update QMF synthesis filter memory.

The following subclauses describe these steps in more detail.

III.10.1 Storage of intermediate ITU-T G.722 states during re-encoding

As described in clause III.7, the reconstructed signal $x_{PLC}(j)$ is re-encoded during lost frames to update the ITU-T G.722 decoder state memory. Let $STATE_j$ be the ITU-T G.722 state and PLC state after re-encoding the jth sample of $x_{PLC}(j)$. Then, in addition to the ITU-T G.722 state at the frame boundary that would normally be maintained (i.e., $STATE_{159}$), the $STATE_{159-\Delta_{TLMAX}}$ is also stored. To facilitate re-phasing, the sub-band signals $x_L(n)$, $x_H(n)$, $n = 69 - \Delta_{TLMAX}/2 \ldots 79 + \Delta_{TLMAX}/2$ are also stored.

III.10.2 Adjustment of the re-encoding according to the time lag

Depending on the sign of the time lag, the procedure for adjustment of the re-encoding is as follows:

If $\Delta_{TL} > 0$:

1) Restore the ITU-T G.722 state and PLC state to $STATE_{159-\Delta_{TLMAX}}$.

2) Re-encode $x_L(n)$, $x_H(n)$, $n = 80 - \Delta_{TLMAX}/2 \ldots 79 - \Delta_{TL}/2$, according to clauses III.7.2 and III.7.3.

If $\Delta_{TL} < 0$:

1) Restore the ITU-T G.722 state and PLC state to $STATE_{159}$.

2) Re-encode $x_L(n)$, $x_H(n)$, $n = 80 \ldots 79 + |\Delta_{TL}/2|$, according to clauses III.7.2 and III.7.3.

Note that to facilitate re-encoding of $x_L(n)$ and $x_H(n)$ up to $n = 79 + |\Delta_{TL}/2|$, samples of $x_{PLC}(j)$ are required up to $\Delta_{TLMAX} + 182$.

III.10.3 Update of QMF synthesis filter memory

At the first received frame, the QMF synthesis filter memory needs to be calculated since the QMF synthesis filterbank is inactive during lost frames due to the PLC taking place in the 16-kHz output speech domain. Time-wise, the memory would generally correspond to the last samples of the last lost frame. However, the re-phasing needs to be taken into account. According to [ITU-T G.722], the QMF synthesis filter memory is given by:

$$x_d(i) = r_L(n-i) - r_H(n-i), \; i = 1, 2, \ldots, 11 \tag{III-94}$$

$$x_s(i) = r_L(n-i) + r_H(n-i), \; i = 1, 2, \ldots, 11 \tag{III-95}$$

as the first two output samples of the first received frame is calculated as:

$$x_{out}(j) = 2\sum_{i=0}^{11} h_{2i} \cdot x_d(i) \tag{III-96}$$

$$x_{out}(j+1) = 2\sum_{i=0}^{11} h_{2i+1} \cdot x_s(i) \tag{III-97}$$

The filter memory, i.e., $x_d(i)$ and $x_s(i)$, i = 1, 2, ..., 11, is calculated from the last 11 samples of the re-phased input to the simplified sub-band ADPCM encoders during re-encoding, $x_L(n)$ and $x_H(n)$, $n = 69 - \Delta_{TL}/2, 69 - \Delta_{TL}/2 + 1, ..., 79 - \Delta_{TL}/2$, i.e., the last samples up till the re-phasing point:

$$x_d(i) = x_L(80 - \Delta_{TL}/2 - i) - x_H(80 - \Delta_{TL}/2 - i),\ i = 1, 2, ..., 11 \tag{III-98}$$

$$x_s(i) = x_L(80 - \Delta_{TL}/2 - i) + x_H(80 - \Delta_{TL}/2 - i),\ i = 1, 2, ..., 11 \tag{III-99}$$

where $x_L(n)$ and $x_H(n)$ have been stored in state memory during the lost frame.

III.11 Time-warping

Time-warping is the process of stretching or shrinking a signal along the time axis. This clause describes how $x_{out}(j)$ is time-warped to improve alignment with the periodic waveform of the extrapolated signal $x_{PLC}(j)$. The algorithm described in the following subclauses is only executed if $T_L \neq 0$.

III.11.1 Time lag refinement

The time lag, T_L, is refined for time-warping by maximizing the cross-correlation in the overlap-add window. The estimated starting position of the overlap-add window within the first received frame based on T_L is given by:

$$SP_{OLA} = \max(0, MIN\_UNSTBL - T_L) \tag{III-100}$$

where $MIN\_UNSTBL = 16$.

The starting position of the extrapolated signal in relation to SP_{OLA} is given by:

$$D_{ref} = SP_{OLA} - T_L - RSR \tag{III-101}$$

where $RSR = 4$ is the refinement search range.

The required length of the extrapolated signal is given by:

$$L_{ref} = OLALG + RSR \tag{III-102}$$

An extrapolated signal, $es_{tw}(j)$, is obtained using the same procedures as clause III.9.3.1, except $LSW = OLALG$, $L = L_{ref}$, and $D = D_{ref}$.

A refinement lag, T_{ref}, is computed by searching for the peak of the following:

$$R(k) = \frac{\sum_{i=0}^{OLALG-1} es_{tw}(i - k + RSR) \cdot x_{out}(i + SP_{OLA})}{\sqrt{\left(\sum_{i=0}^{OLALG-1} {es_{tw}}^2(i - k + RSR)\right) \cdot \left(\sum_{i=0}^{OLALG-1} {x_{out}}^2(i + SP_{OLA})\right)}},\ k = -RSR, -RSR + 1, \ldots, RSR \tag{III-103}$$

The final time lag used for time-warping is then obtained by:

$$T_{Lwarp} = T_L + T_{ref} \tag{III-104}$$

III.11.2 Computation of time-warped $x_{out}(j)$ signal

The signal $x_{out}(j)$ is time-warped by T_{Lwarp} samples to form the signal $x_{warp}(j)$ which is later overlap-added with the waveform extrapolated signal $es_{ola}(j)$. Three cases, depending on the value of T_{Lwarp}, are illustrated in Figure III.9. In case a, $T_{Lwarp} < 0$ and $x_{out}(j)$ undergoes shrinking or compression. The first *MIN_UNSTBL* samples of $x_{out}(j)$ are not used in the warping to create $x_{warp}(j)$ and *xstart=MIN_UNSTBL*. In case b, $0 \le T_{Lwarp} < MIN\_UNSTBL$, and $x_{out}(j)$ is stretched by T_{Lwarp} samples. Again, the first *MIN_UNSTBL* samples of $x_{out}(j)$ are not used and *xstart=MIN_UNSTBL*. In case c, $T_{Lwarp} \ge MIN\_UNSTBL$, and $x_{out}(j)$ is once more stretched by T_{Lwarp} samples. However, the first T_{Lwarp} samples of $x_{out}(j)$ are not needed in this case since an extra T_{Lwarp} sample will be created during warping; therefore, $xstart = T_{Lwarp}$.

In each case, the number of samples per add/drop is given by:

$$spad = \frac{(160 - xstart)}{|T_{Lwarp}|} \tag{III-105}$$

The warping is implemented via a piece-wise single sample shift and triangular overlap-add, starting from $x_{out}[xstart]$. To perform shrinking, a sample is periodically dropped. From the point of sample drop, the original signal and the signal shifted left (due to the drop) are overlap-added. To perform stretching, a sample is periodically repeated. From the point of sample repeat, the original signal and the signal shifted to the right (due to the sample repeat) are overlap-added. The length of the overlap-add window, $L_{olawarp}$, (note that this is different from the OLA region depicted in Figure III.9) depends on the periodicity of the sample add/drop and is given by:

$$\text{If } T_{Lwarp} < 0,\ L_{olawarp} = \frac{\left(160 - xstart - |T_{Lwarp}|\right)}{|T_{Lwarp}|}$$

$$\text{Else } L_{olawarp} = \lceil spad \rceil \tag{III-106}$$

$$L_{olawarp} = \min(8, L_{olawarp})$$

The length of the warped input signal, x_{warp}, is given by:

$$L_{xwarp} = \min\left(160,\ 160 - MIN\_UNSTBL + T_{Lwarp}\right) \tag{III-107}$$

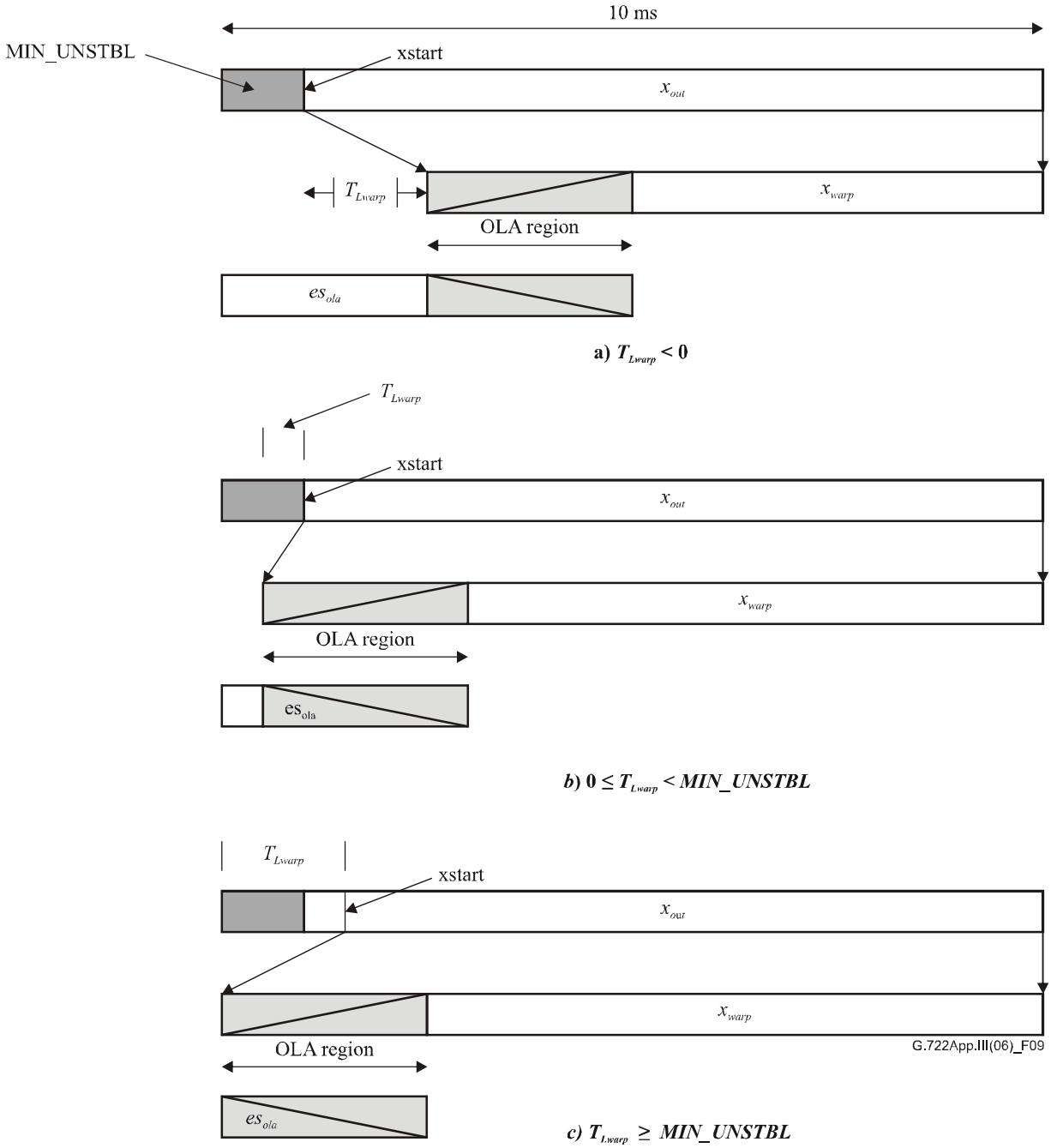


Figure III.9 – Three cases for warping of x_{out}

III.11.3 Computation of the waveform extrapolated signal

The warped signal $x_{warp}(j)$ and the extrapolated signal $es_{ola}(j)$ are overlap-added in the first received frame as shown in Figure III.9. The extrapolated signal $es_{ola}(j)$ is generated directly within the $x_{out}(j)$ signal buffer in a two-step process according to:

Step 1:

$$es_{ola}(j) = x_{out}(j) = ptfe \cdot x_{out}(j - ppfe), \quad j = 0, 1, ..., 160 - L_{xwarp} + 39 \qquad \text{(III-108)}$$

Step 2:

$$x_{out}(j) = x_{out}(j) \cdot w_i(j) + ring(j) \cdot w_o(j), \quad j = 0, 1, ..., 39 \qquad \text{(III-109)}$$

where $w_i(j)$ and $w_o(j)$ (respectively) are triangular upward and downward ramping overlap-add windows of length 40 and *ring(j)* is the ringing signal computed in clause III.6.12.

III.11.4 Overlap-add of time-warped signal with the waveform extrapolated signal

The extrapolated signal computed in clause III.11.3 is overlap-added with the warped signal $x_{warp}(j)$ according to:

$$x_{out}(160 - L_{xwarp} + j) = x_{out}(160 - L_{xwarp} + j) \cdot w_o(j) + x_{warp}(j) \cdot w_i(j), \quad j = 0, 1, \ldots, 39 \qquad \text{(III-110)}$$

The remaining part of $x_{warp}(j)$ is then simply copied into the signal buffer:

$$x_{out}(160 - L_{xwarp} + j) = x_{warp}(j), \quad j = 40, 41, ..., L_{xwarp} - 1 \qquad \text{(III-111)}$$

III.12 Bit-exact description of the ITU-T G.722 PLC algorithm

The ITU-T G.722 PLC is specified in a bit-exact manner by the fixed-point ANSI C-code. The C-code specification takes precedence over the text specification in this Recommendation in case of discrepancy. The fixed-point ANSI C-code is implemented using the ITU-T G.722 C-code of the ITU-T G.191 Software Tool Library (STL2005) and version 2.3 of the 16-bit fixed-point operator library, also of STL2005.

III.12.1 Use of the simulation software

The decg722 executable (including PLC) is called by:

```
decg722 [-fsize N] g192_input_file speech_output_file
```

where N, the frame size, is a multiple of 160, corresponding to multiples of 10 ms.

Note that the Mode and frame size is indirectly embedded in the ITU-T G.192 bit-stream; conflict with a command line-specified frame size will cause the decoder to use a frame size consistent with the ITU-T G.192 bit-stream. For frame sizes of 10 ms and 20 ms, the decoder can uniquely determine both mode and frame size from the ITU-T G.192 bit-stream. However, for frame sizes of 30 ms and longer, the frame size must be specified in a command line so as to allow the decoder to correctly determine both mode and frame size.

III.12.2 Organization of the simulation software

The source code is contained in the directory named "src". A Microsoft Visual C 6.0 workspace file is located in "workspace/VC6.0/". Opening g722_plc_g192.dsw will open the ITU-T G.722 PLC C source code and project.

After compilation a simple test for bit-exact operation can be performed in the directory named "testplc". Executing the Pearl script named "testplc.pl" in the directory will execute the test.

Note that passing this simple test for bit-exactness only provides a simple check and is far from exhaustive in verifying an implementation for proper operation.

Table III.3 lists the new ITU-T G.722 decoder state memory to support the PLC, and Table III.4 provides an overview of new tables.

Table III.3 – New ITU-T G.722 decoder state memory (structure WB_PLC_state)

Member	Words (16-bit)	Description
energymax32	2	Energy
cormax	2	Correlation
wsz	1	Window size
scaled_flag	1	Scaling flag
xq	638	16-kHz speech output buffer
stsyml	8	Short-term synthesis filter memory

Table III.3 – New ITU-T G.722 decoder state memory (structure WB_PLC_state)

Member	Words (16-bit)	Description
al	9	LPC filter coefficients
alast	9	Past LPC filter coefficients
ppt	1	Pitch predictor tap
stwpml	8	Short-term weighted all-pole filter memory
xwd	45	Down-sampled weighted speech buffer
xwd_exp	1	Exponent of down-sampled weighted speech buffer
dfm	60	Down-sampling filter memory
scaler	1	Scaling factor for random component
merit	1	Figure of merit for mixing ratio
ptfe	1	Pitch tap for frame erasure
ppf	1	Pitch period – "floating" point value
ppinc	1	Pitch period increment
pweflag	1	Periodic waveform extrapolation flag
cpplast	1	Coarse pitch period, last
pph	5	Pitch period history
pp	1	Pitch period
cfecount	1	Consecutive frame erasure counter
ngfae	1	Number of good frames after erasure
nfle	1	Number of frames in last erasure
avm	1	Average magnitude
lag	1	Time shift lag
psml_mean	1	Pole section margin, low-band mean
nbpl_mean1	1	nbpl first mean (low-band)
nbpl_mean2	1	nbpl second mean (low-band)
nbpl_trck	1	nbpl tracking (low-band)
nbpl_chng	1	nbpl change (low-band)
pl_postn	1	pl signal positive-negative measure (low-band)
lb_reset	1	Low-band decoder reset flag
nbph_mean	1	nbph mean (high-band)
nbph_trck	1	nbph tracking (high-band)
nbph_chng	1	nbph change (high-band)
nbh_mode	1	nbh mode for convergence (high-band)
hp_flag	1	Flag for hp filter on rh and ph signals (high-band)
nbph_lp	1	Low-pass filtered nbph (high-band)
ph_postn	1	ph signal positive-negative measure (high-band)
hb_reset	1	High-band decoder reset flag
rhhp_m1	1	Past sample of high-pass filtered rh signal (high-band)
rh_m1	1	Past sample of rh signal (high-band)
phhp_m1	1	Past sample of high-pass filtered ph signal (high-band)

Table III.3 – New ITU-T G.722 decoder state memory (structure WB_PLC_state)

Member	Words (16-bit)	Description
ph_m1	1	Past sample of ph signal (high-band)
sb_sample	1	Sub-band sample number
cpl_postn	1	Copy of pl_postn
cph_postn	1	Copy of ph_postn
crhhp_m1	1	Copy of rhhp_m1
crh_m1	1	Copy of rh_m1
cphhp_m1	1	Copy of phhp_m1
cph_m1	1	Copy of ph_m1
ds	104	Copy of regular ITU-T G.722 decoder state
lb	39	Low-band signal
hb	39	High-band signal

Table III.4 – New ITU-T G.722 decoder table ROM

Table	Words (16-bit)	Description
inv_frm_size	3	Inverse of frame size
wlil4rilil	9	Table for low-band scale factor update
q4	8	Table for low-band scale factor update
NGFAEOFFSET_P1	8	Sample offset into 10 ms frames
div_n	20	Table for division
gawd	4	Table for gradual muting
olaup	16	Table for overlap-add
oladown	16	Table for overlap-add
wn	127	Table of normalized noise samples
bdf	60	Filter for 8:1 decimation
x	4	For coarse pitch
x2	4	For coarse pitch
invk	4	For coarse pitch
MPTH	4	For coarse pitch
sstwin_h	8	Upper 16-bit for spectral smoothing
sstwin_l	8	Lower 16-bit for spectral smoothing
bwel	9	Bandwidth expansion
STWAL	9	For short-term weighting filter
win	160	Window for LPC analysis
tablog	33	Table for log2 function
olaug	40	Window for overlap-add
oladg	40	Window for overlap-add
nbphtab	8	Table for nbph

Table III.4 – New ITU-T G.722 decoder table ROM

Table	Words (16-bit)	Description
nbpltab	6	Table for nbpl
ola3	3	Table for overlap-add
ola4	4	Table for overlap-add
ola5	5	Table for overlap-add
ola6	6	Table for overlap-add
ola7	7	Table for overlap-add
ola8	8	Table for overlap-add

Table III.5 lists C-code source files of [ITU-T G.722] from [ITU-T G.191] that remain identical, Table III.6 lists files of [ITU-T G.722] that are modified, and Table III.7 names the new files of the ITU-T G.722 PLC.

Table III.5 – ITU-T G.722 identical source files

File name	Description
decg722.c	ITU-T G.722 main decoder function
softbit.c/h	ITU-T G.722 softbit functions
g722_com.h	Common ITU-T G.722 definitions
ugstdemo.h	Definitions for UGST demo programs

Table III.6 – ITU-T G.722 modified source files

File name	Description
funcg722.c/h	ITU-T G.722 functions
g722.c	ITU-T G.722 frame encoding and decoding functions

Table III.7 – ITU-T G.722 PLC new source files

File name	Description
apfilter.c	All-pole filter functions
autocor.c	Autocorrelation function
azfilter.c	All-zero filter function
coarptch.c	Coarse pitch analysis
decim.c	8:1 decimation function
dspfunc.c	DSP functions
g722plc.c/h	ITU-T G.722 PLC functions
levinson.c	Levinson-Durbin recursion function
memutil.c/h	Memory utility functions allow automatic scratch memory usage
merit.c	Merit calculation function
ppchange.c/h	Re-phasing and time-warping related functions

Table III.7 – ITU-T G.722 PLC new source files

File name	Description
prfn.c	Pitch refinement function
table.c/h	Table ROM
utility.c/h	Utility functions

Appendix IV

A low-complexity algorithm for packet-loss concealment with ITU-T G.722

(This appendix does not form an integral part of this Recommendation.)

IV.1 Scope

This appendix[2] describes a low-complexity packet loss concealment (PLC) algorithm for use by the ITU-T G.722 audio decoder to limit speech quality degradation in the presence of packet loss. The statistical analysis of the ITU-T G.722 PLC selection test results has demonstrated that this appendix meets the same quality requirements as Appendix III. This is achieved with almost no additional complexity compared with ITU-T G.722 normal decoding (0.07 WMOPS), but with a lower quality than Appendix III. Accordingly, this algorithm is suitable for applications that may encounter frame erasures or packet losses, with a special focus on applications having strong complexity constraints and on low-cost devices. As examples, these applications include DECT next generation and VoIP.

The references, abbreviations and notations used throughout this appendix are defined in clauses IV.2, IV.3 and IV.4, respectively. Clause IV.5 gives a general description of the algorithm and includes delay and complexity information. Clause IV.6 contains the detailed functional description of the algorithm. Finally, clause IV.7 provides the information related to the simulation software.

IV.2 References

[ITU-T G.191 Ann.A] Recommendation ITU-T G.191 (2005), *Software tools for speech and audio coding standardization. Annex A: List of software tools available*.

[ITU-T G.192] Recommendation ITU-T G.192 (1996), *A common digital parallel interface for speech standardization activities*.

[ITU-T G.729] Recommendation ITU-T G.729 (2007), *Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)*.

IV.3 Abbreviations and acronyms

This appendix uses the following abbreviations and acronyms:

FIR Finite Impulse Response

HB Higher Band

HPF High Pass Filter

IIR Infinite-Impulse Response

LB Lower Band

LP Linear Prediction

LPC Linear Predictive Coding

LSB Least Significant Bit

LTP Long-Term Prediction

MSB Most Significant Bit

[2] This appendix includes an electronic attachment containing the relevant ANSI-C code.

PLC Packet Loss Concealment

QMF Quadrature Mirror Filter

WB Wideband

WMOPS Weighted Million Operations Per Second

IV.4 Notations and conventions

Throughout this appendix, the terms "*frame*" and "*packet*" are considered to be equivalent.

The notation used in [ITU-T G.722] is followed. The notational conventions are detailed below:

– Each frame comprises $2L$ samples at 16 kHz, where $L = 80$ for 10-ms frames and 160 for 20-ms frames.

– Time-domain signals are denoted by their symbol and a sample index between parentheses [e.g., $y(n)$]. The symbol n is used as sample index.

– By convention, for signals sampled at 8 kHz, the current frame corresponds to $n = 0, \ldots, L - 1$. Samples with an index $n < 0$ are past samples, and samples with index $n \geq L$ are future samples.

Table IV.1 – List of symbols

Type	Name	Description
Filters	$A(z)$	8th-order LPC filter
	$B(z)$	2nd-order LPC filter
	$H_{pre}(z)$	Lower-band high-pass filter
	$H_{post}(z)$	Higher-band high-pass filter
	$H_{dec}(z)$	4:1 decimation filter
Signals	$x_l(n)$	Lower-band ADPCM decoder output
	$y_l(n)$	Lower-band extrapolated signal
	$z_l(n)$	Lower-band reconstruction
	$x_h(n)$	Higher-band ADPCM decoder output
	$y_h(n)$	Higher-band extrapolated signal
	$z_h(n)$	Higher-band reconstruction
	$e(n)$	Lower-band LP residual signal
Parameters	a_i	Lower-band LP coefficients
	T_0	Pitch delay in lower band
	g_mute_lb	Lower-band muting factor
	g_mute_hb	Higher-band muting factor

IV.5 General description of the ITU-T G.722 PLC algorithm

The proposed ITU-T G.722 PLC algorithm is integrated in the standard ITU-T G.722 decoder. Note that the ITU-T G.722 encoder is unchanged (identical to clause 3 of [ITU-T G.722]).

IV.5.1 Modified ITU-T G.722 decoder

The modified ITU-T G.722 decoder is illustrated in Figure IV.1. Decoding is performed in two sub-bands, which are combined using the QMF synthesis filterbank of [ITU-T G.722]. Compared to [ITU-T G.722], this decoder includes a mechanism to conceal frame erasures, shown in the highlighted (grey-shaded) blocks.

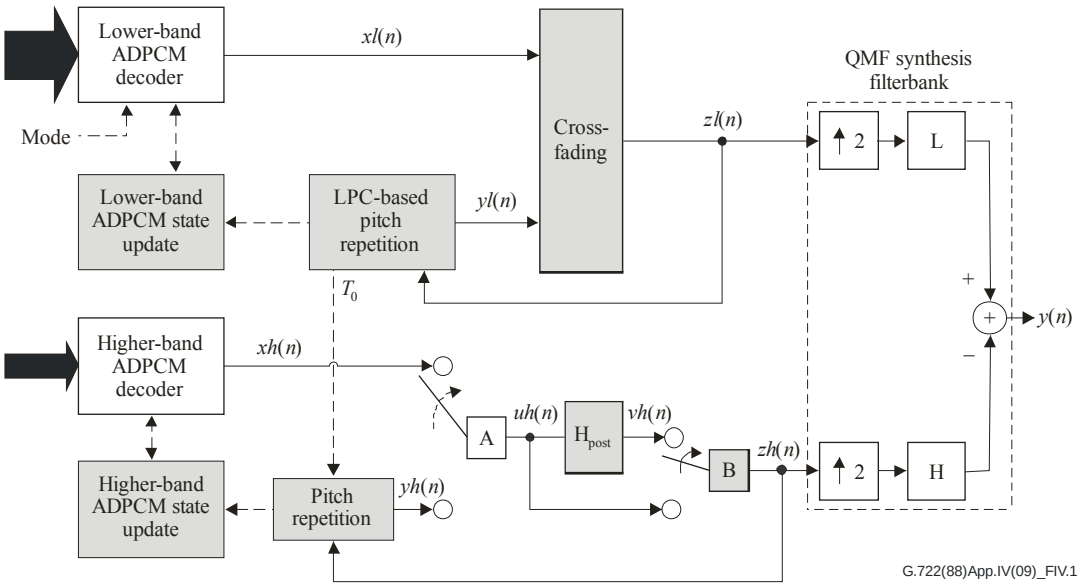


Figure IV.1 – Block diagram of ITU-T G.722 decoder with PLC (the highlighted blocks correspond to the PLC algorithm)

The modified ITU-T G.722 decoder generates an output signal sampled at 16 kHz and divided into frames of 10 or 20 ms. Its behaviour depends on the type of the current and previous frame (either good or bad frame):

– Without frame erasures (i.e., in the presence of good frames only):

  The bit stream of the lower band (LB) is decoded according to the specified mode (1, 2 or 3 indicating 64, 56 or 48 kbit/s, respectively). The cross-fading block does not change the reconstructed signal, i.e., $zl(n) = xl(n)$ similarly, the bit stream of the higher band (HB) is decoded. The A and B switches select $uh(n) = xh(n)$ and $zh(n) = uh(n) = xh(n)$, respectively. The wideband reconstruction $y(n)$ is obtained, as in [ITU-T G.722], from the QMF synthesis filterbank. The decoded signals $zl(n)$ and $zh(n)$ are stored to be used in case of erasure in future frames.

– In case of frame erasure:

  • In the lower band, for the first erased frame, short- and long-term predictors are updated using the past valid signal $zl(n)$, $n < 0$. Class information is also extracted. The signal $yl(n)$ is generated using these predictors and the class information. The signal for the erased frame is reconstructed as $zl(n) = yl(n)$, $n = 0, \ldots, L - 1$. In addition, ADPCM states are updated. The process of erased frame reconstruction and ADPCM states update is repeated until a good frame is received. Note that not only the missing frame is generated, but also an additional 10 ms of signal, $yl(n)$, $n = L, \ldots, L + 79$, to be used for cross-fading. Once a good frame is received, the signal reconstructed by the ADPCM decoder, $xl(n)$, $n = 0, \ldots, L - 1$, and the extrapolated signal, $yl(n)$, $n = 0, \ldots, L - 1$ are cross-faded. This cross-fading is used only during the first 10 ms following the last erasure.

  • In the higher band, the missing frame is extrapolated using the past signal $zh(n)$, $n < 0$, and ADPCM states are updated. The extrapolated signal $yh(n)$ is obtained by repeating pitch-synchronously the previous frame of $zh(n)$. The "A" switch selects $uh(n) = yh(n)$, $n = 0, \ldots, L - 1$. The signal *uh(n)* is high-pass filtered by a remove-DC filter H_{post} to obtain $vh(n)$. The "B" switch selects $zh(n) = vh(n)$, $n = 0, \ldots, L - 1$. This process is repeated until a good frame is received. Once a good frame is received, the "A" switch selects the ADPCM decoder output: $uh(n) = xh(n)$, $n = 0, \ldots, L - 1$. During the first 4 s

following the last erasure, the "B" switch selects: $zh(n) = vh(n)$, $n = 0, \ldots, L - 1$; after 4 s, the B switch selects: $zh(n) = uh(n)$, $n = 0, \ldots, L - 1$.

IV.5.2 Delay and complexity

The proposed ITU-T G.722 PLC operates with no extra delay.

The computational complexity and storage requirements are summarized in Table IV.2.

Table IV.2 – Complexity figures of the ITU-T G.722 PLC algorithm

a) Observed worst-case complexity, in WMOPS, based on STL2005 (figures in brackets represent the additional WMOPS compared with ITU-T G.722 decoding)

Frame length	10 ms	20 ms
ITU-T G.722 decoding (no PLC)	3.11	3.10
ITU-T G.722 with PLC	3.18 [0.07]	3.15 [0.05]

b) Memory requirement (figures in brackets represent the maximum additional complexity compared with ITU-T G.722 decoding)

Frame length	Static RAM (in 16-bit kwords)	Scratch RAM (in 16-bit kwords)	Total RAM	Program ROM (in number of basic ops and function calls)	Data ROM (in 16-bit kwords)
10 ms	967 [863]	692 [452]	1659 [1315]	1061 [615]	882 [109]
20 ms		963 [452]	1899 [1315]		

Compared to the ITU-T G.722 reference decoder, the PLC algorithm brings an additional worst-case complexity of 0.07 WMOPS. Note that the complexity peak occurs at the first valid frame after an erasure.

IV.6 Functional description of the ITU-T G.722 PLC algorithm

IV.6.1 Lower-band decoding

IV.6.1.1 ADPCM decoder in case of a good frame

Same as clauses 4.1, 4.2 and 4.3 of [ITU-T G.722].

In addition, the counter *cnt_mute_lb* and muting factor *g_mute_lb* (used in clause IV.6.1.2.7) for adaptive muting are reset:

$$cnt\_mute\_lb = 0$$

$$g\_mute\_lb = 1$$

and the signal $zl(n)$ is stored to be used in case of erasure in future frames.

IV.6.1.2 Extrapolation of missing frame: Case of bad frame following a good frame

The extrapolation of a missing frame in the lower band is illustrated in Figure IV.2. It comprises analysis of the past valid signal $zl(n)$ n, < 0, followed by synthesis of the signal $yl(n)$, $n = 0, \ldots,$

$L - 1$. The missing frame in the lower band corresponds to $xl(n)$, $n = 0, \ldots, L - 1$ where $L = 80$ for 10-ms frames and $L = 160$ for 20-ms frames.

The past signal $zl(n)$, $n = -297, \ldots, -1$ is buffered using a buffer length of 297 samples, which can be divided as follows:

– the 288 samples corresponding to twice the maximal pitch delay (2×144) used in the PLC algorithm;

– one sample for pitch jitter; and

– eight samples used for LPC memory.

This buffer length makes it possible to store the last two pitch periods of the lower-band reconstruction.

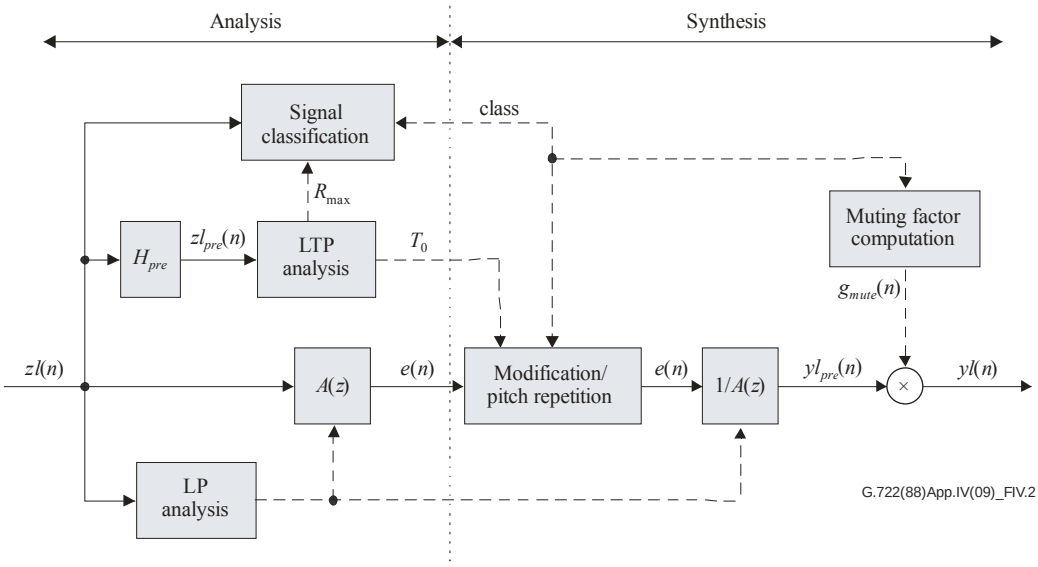


Figure IV.2 – Block diagram of lower-band extrapolation of missing frame

IV.6.1.2.1 LP analysis

The short-term analysis and synthesis filters, $A(z)$ and $1/A(z)$, are based on eighth-order linear prediction (LP) filters. The LP analysis filter is defined as:

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_8 z^{-8} \tag{IV-1}$$

The LP analysis consists of two parts: windowing and autocorrelation computation, and the Levinson-Durbin algorithm. The autocorrelation computation – including 60-Hz bandwidth expansion and 40-dB white-noise correction – is identical to that in clause 3.2.1 of [ITU-T G.729], with only one difference. The LP window here is an asymmetrical Hamming window defined as:

$$w_{lp}(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{(n+80)\pi}{69}\right), & n = -80, \cdots, -11 \\ 0.54 + 0.46\cos\left(\dfrac{(n+11)\pi}{10}\right), & n = -10, \cdots, -1 \end{cases} \tag{IV-2}$$

This window $w_{lp}(n)$, which is limited to 80 samples (10 ms at 8-kHz sampling frequency) to reduce complexity, is applied to the last 10 ms of $zl(n)$, $n = -80, \ldots, -1$. The Levinson-Durbin algorithm is identical to clause 3.2.2 of [ITU-T G.729].

After the LP analysis, the past signal $zl(n)$, $n = -289, \ldots, -1$, is filtered through $A(z)$ to obtain the residual signal $e(n)$, $n = -289, \ldots, -1$

$$e(n) = zl(n) + \sum_{i=1}^{8} a_i \; zl(n-i) \tag{IV-3}$$

IV.6.1.2.2 Pre-processing

A high-pass filter protects against undesired low-frequency components. A first-order pole/zero filter with a cut-off frequency of 50 Hz is used. This filter is given by:

$$H_{pre}(z) = \frac{1 - z^{-1}}{1 - \frac{123}{128} z^{-1}} \tag{IV-4}$$

The past signal $zl(n)$, $n = -288, \ldots, -1$, is filtered through $H_{pre}(z)$ to obtain the pre-processed signal $zl_{pre}(n)$, $n = -288, \ldots, -1$.

IV.6.1.2.3 LTP analysis

The PLC algorithm uses pitch period repetition. The pitch period or pitch delay, T_0, is determined on the past valid pre-processed signal just before erasure, $zl_{pre}(n)$, $n = -288, \ldots, -1$. T_0 is estimated in open loop by a long-term predictive (LTP) analysis.

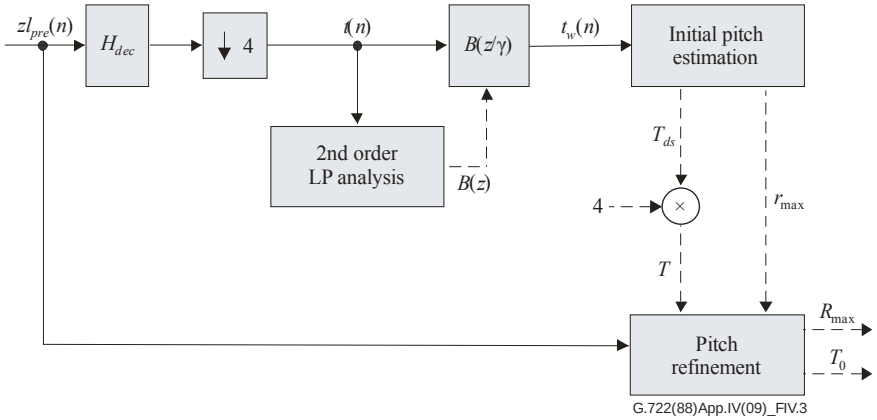


Figure IV.3 – Block diagram of LTP analysis

As illustrated in Figure IV.3, pitch estimation is conducted in the following steps:

– The signal $zl_{pre}(n)$, $n = -288, \ldots, -1$, is low-pass filtered by $H_{dec}(z)$, where:

$$H_{dec}(z) = \frac{3692(1 + z^{-8}) + 6190(z^{-1} + z^{-7}) + 8525(z^{-2} + z^{-6}) + 10186(z^{-3} + z^{-5}) + 10787 z^{-4}}{65536} \tag{IV-5}$$

is an eighth-order FIR filter, and decimated by a factor of 4 to obtain the signal $t(n)$, $n = -72, \ldots, -1$, sampled at 2 kHz. The filter memories are initialized to 0 each time.

– The signal $t(n)$, $n = -72, \ldots, -1$, is weighted by a filter $B(z/\gamma)$, where $B(z) = 1 - b_1 z^{-1} - b_2 z^{-2}$ and $\gamma = 0.94$, to obtain the signal $t_w(n)$, $n = -72, \ldots, -1$. The coefficients of $B(z)$ are obtained by 2nd-order LP analysis of $t(n)$ using the windowing, autocorrelation computation and Levinson-Durbin algorithm described in the previous clause. Note that only the last 72 samples of the window $w_{lp}(n)$, $n = -72, \ldots, -1$, are used, which gives a 36-ms time support at 2-kHz sampling frequency.

– A first estimation T_{ds} of the pitch delay is computed in the weighted decimated signal domain by normalized cross-correlation as follows:

a) Initialization: $T_{ds} = 18$.

b) Computation of the normalized cross-correlation:

$$r(i)=\frac{\sum_{j=-35}^{-1} t_w(j)\,t_w(j-i)}{\max\left(\sum_{j=-35}^{-1} t_{\mathrm{w}}^2(j),\ \sum_{j=-35}^{-1} t_{\mathrm{w}}^2(j-i)\right)},\ i=1,\ldots,35 \qquad \text{(IV-6)}$$

c) Determination of the first delay i_0 in [1,35] for which $r(i) < 0$. Note that if $\min_{\mathrm{i=1,\cdots,35}} r(i)\geq 0$, steps d) and e) are omitted.

d) Determination of the lower bound for the maximum correlation search:

$$i_1 = \max(i_0, 4)$$

e) Search for the maximum correlation in $[i_1, 35]$:

$$T_{ds} = \arg \max_{\mathrm{i=}i_1,\cdots,35} r(i) \qquad \text{(IV-7)}$$

A procedure favouring the smaller pitch values to avoid choosing pitch multiples is also applied.

– The first pitch delay estimation T_{ds} is then refined in the pre-processed signal domain by searching the cross-correlation maximum in the neighbourhood of $T = 4T_{ds}$ to obtain T_0 as follows:

$$T_0 = \arg \max_{i=T-2,\cdots,T+2} R(i) \qquad \text{(IV-8)}$$

with:

$$R(i)=\frac{\sum_{j=-T}^{-1} zl_{pre}(j)\ zl_{pre}(j-i)}{\max\left(\sum_{j=-T}^{-1} zl_{pre}^2(j),\ \sum_{j=-T}^{-1} zl_{pre}^2(j-i)\right)} \qquad \text{(IV-9)}$$

– The value $R_{\max}$ is computed as $R_{\max} = R(T_0)$.

IV.6.1.2.4 Signal classification

The PLC strategy uses signal characteristics to optimize quality. For instance, if the frame preceding an erasure is a non-stationary segment (e.g., plosives), the signal should be rapidly muted; if this frame is a stationary segment (e.g., strongly voiced speech), it can be pitch-synchronously repeated and slowly damped. Classification is used in the proposed PLC algorithm for LP residual extrapolation and muting control.

The signal $zl(n)$, $n = -288, \ldots, -1$ preceding an erasure is classified into one out of five possible classes, which are defined below:

– TRANSIENT for transients with large energy variation (e.g., plosives);

– UNVOICED for unvoiced signals;

– VUV_TRANSITION corresponding to a transition between voiced and unvoiced signals;

– WEAKLY_VOICED for weakly voiced signals (e.g., onset or offset of vowels);

– VOICED for voiced signals (e.g., steady vowels).

The features used for classification are listed below:

– Normalized correlation $R_{\max}$, which is a side product of the LTP analysis.

– Sub-band energy ratio, which is obtained here in the log domain by taking the difference between the lower- and higher-band ADPCM scale factors, NBH – NBL using the ITU-T G.722 notations. NBL and NBH are computed as in clause 3.5 of [ITU-T G.722].

– Zero-crossing rate *zcr* of $zl(n)$, $n = -80, \ldots, -1$, defined as:

$$zcr = \sum_{n=-80}^{-1} \left[(zl(n) \le 0) \; AND \; (zl(n-1) > 0) \right] \tag{IV-10}$$

where the comparisons ≤ and > give a binary result (1 for true, 0 for false) and "AND" is the AND bit operation.

– Number *cnt_peak* of detected large peaks in the last pitch period:

$$cnt\_peak = \sum_{n=-T_0}^{-1} \left[\frac{|e(n)|}{4} > \max_{i=-2,\ldots,2} \left(|e(n - T_0 + i)| \right) \right] \tag{IV-11}$$

where the comparison > gives a binary result (1 for true, 0 for false) and T_0 is the pitch delay estimated by the LTP analysis. The counter, *cnt_peak*, represents the number of detected large peaks in the last pitch period.

Based on these features, the signal category, *class*, is obtained by heuristics according to the flowchart shown in Figure IV.4. Note that if *class* is set to UNVOICED, the pitch delay T_0 may be modified to avoid artefacts due to low-pitch delay values. Note also that if *class* is not VOICED, and T_0 is even, T_0 is increased by 1. The so-called pitch delay T_0 determines the repetition period used in the residual signal generation procedure.

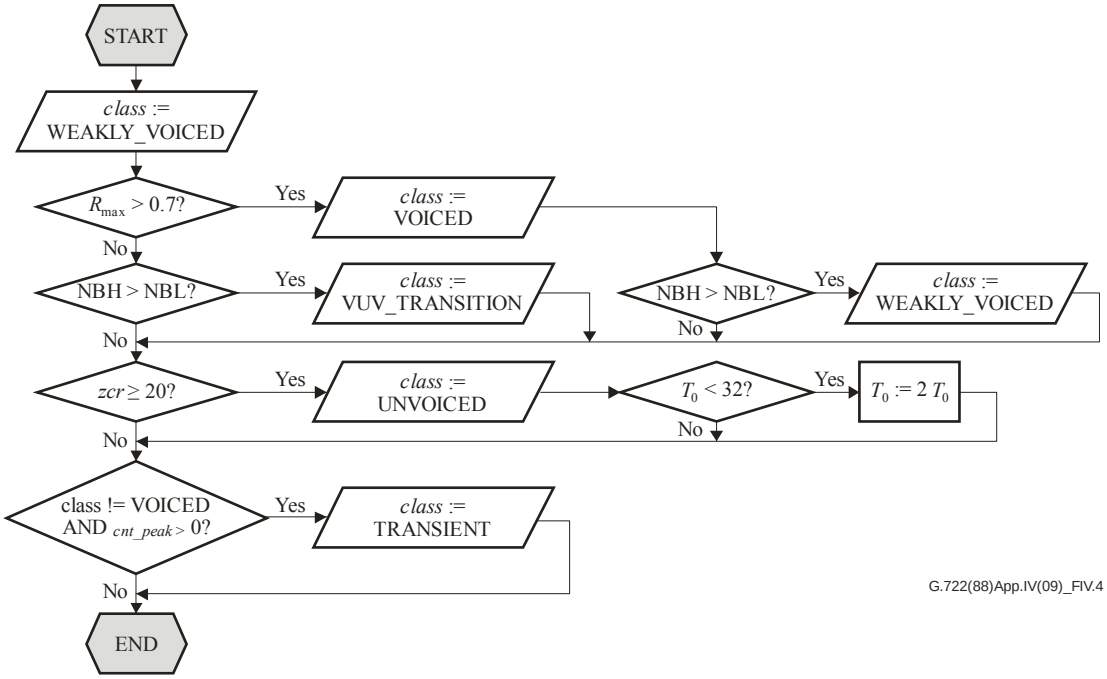


Figure IV.4 – Classification flowchart

IV.6.1.2.5 Modification/pitch repetition of LP residual

A pitch repetition procedure is used to estimate the LP residual $e(n)$, $n = 0, \ldots, L - 1$ in the missing frame from the residual signal of the repetition period. As mentioned above, the repetition period contains the last valid T_0 residual samples $e(n)$, $n = -T_0, \ldots, -1$.

LP residual modification

Before performing the pitch repetition procedure, the repetition period is modified if *class* is not VOICED. The modification consists in limiting the magnitude of each sample of the repetition period as follows:

$$e(n) = \min\left(\max_{i=-2,\cdots,+2} \left(\left|e(n-T_0+i)\right|\right), \left|e(n)\right| \right) \times sign(e(n)), \quad n = -T_0, \cdots, -1 \tag{IV-12}$$

where:

$$sign(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ -1 \text{ if } x < 0 \end{cases} \tag{IV-13}$$

Pitch repetition of LP residual

The LP residual $e(n)$, $n = 0, \ldots, L-1$, in the missing frame is extrapolated based on the classification result:

– If *class* is VOICED, the missing signal, $e(n)$, $n = 0, \ldots, L-1$, is obtained by repeating pitch-synchronously the repetition period:

$$e(n) = e(n - T_0) \tag{IV-14}$$

– If *class* is not VOICED, the pitch-synchronous repetition procedure is modified to avoid over-voicing by introducing, sample by sample, a small jitter using the following procedure. The samples of the repetition period can be viewed as grouped two by two; then, every two samples forming a group are swapped and the swapped groups are concatenated to form the extrapolated residual signal. If $T_0 < L$, the extrapolated residual signal extends the repetition period and the procedure is iterated. With this procedure, the missing signal, $e(n)$, $n = 0, \ldots, L-1$ is obtained as:

$$e(n) = e(n - T_0 + (-1)^n) \tag{IV-15}$$

This procedure is illustrated in Figure IV.5.

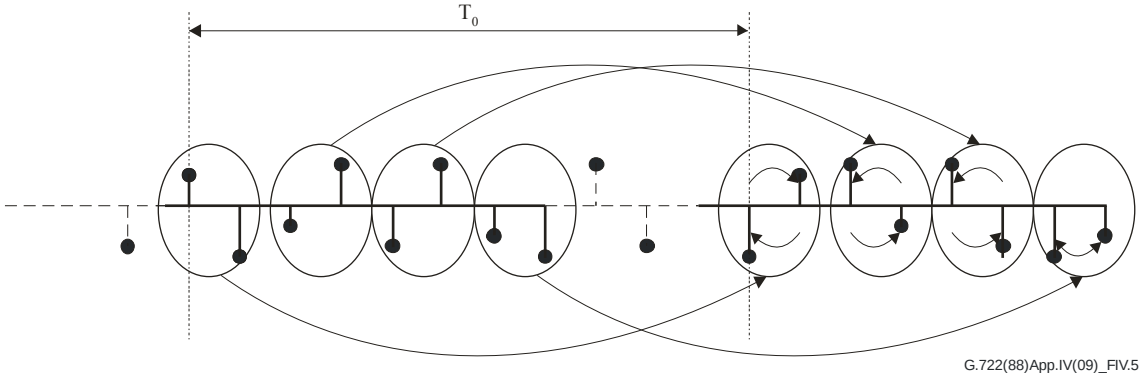


Figure IV.5 – LP residual *e*(*n*) extrapolation with jitter (if *class* is not VOICED)

In addition, 80 extra samples (10 ms), $e(n)$, $n = L, \ldots, L + 79$ are generated using the above equations for the purpose of cross-fading.

IV.6.1.2.6 LP synthesis

After LP synthesis, the reconstructed missing frame is given by:

$$yl_{pre}(n) = e(n) - \sum_{i=1}^{8} a_i \, yl(n-i) \tag{IV-16}$$

where $e(n)$, $n = 0, \ldots, L - 1$, is the extrapolated residual signal and L is the frame length.

In addition, 80 samples (10 ms), $yl_{pre}(n)$, $n = L, \ldots, L + 79$ are generated using the above equation; these samples are used for cross-fading.

IV.6.1.2.7 Adaptive muting

The energy of the reconstructed signal is controlled by applying to each sample a gain factor that is computed and adapted sample by sample. Thus, the synthesized signal $yl_{pre}(n)$ is muted sample by sample with an adaptive muting factor *g_mute_lb* for $n = 0, \ldots, L - 1$ to obtain the reconstructed lower-band signal $yl(n)$:

$$yl(n) = g\_mute\_lb \times yl_{pre}(n) \qquad \text{(IV-17)}$$

The extra synthesis needed for cross-fading, $yl_{pre}(n)$, $n = L, \ldots, L + 79$, is muted in the same way: $yl(n) = g\_mute\_lb \times yl_{pre}(n)$.

The calculation of *g_mute_lb* is performed using several parameters, *inc_mute*, *fac*1, *fac*2*p*, and *fac*3*p*, for 10- and 20-ms frames, as well as *cf*10 for 10-ms frames. These parameters depend on the value of *class* as indicated in Table IV.3.

Table IV.3 – Adaptive muting parameters

Parameter	*class* = **TRANSIENT**	*class* = **UV_TRANSITION**	**Other cases**
inc_mute	4	2	1
*fac*1	409	10	10
*fac*2*p*	409	10	20
*fac*3*p*	409	399	190
*cf*10	0	399	20

Computation of muting factor for the first erased 10-ms frame

In this case, the muting factor is adapted sample by sample with *fac*1 as follows, for $n = 0, \ldots, 79$:

$$g\_mute\_lb = g\_mute\_lb - fac1$$

where *g_mute_lb* has been initialized to 1 (see clause IV.6.1.1).

Then *cnt_mute_lb* is updated at the end of the first erased 10-ms frame:

$$cnt\_mute\_lb = cnt\_mute\_lb + 80 \times inc\_mute$$

where *cnt_mute_lb* has been initialized to 0 (see clause IV.6.1.1).

The muting factor for the extra synthesis is also adapted sample by sample with *cf*10 in the same way for $n = 80, \ldots, 159$:

$$g\_mute\_lb = g\_mute\_lb - cf10$$

Then *cnt_mute_lb* is also updated at the end of the extra frame:

$$cnt\_mute\_lb = cnt\_mute\_lb + 80 \times inc\_mute$$

The muting factor adaptation is illustrated in Figure IV.6.

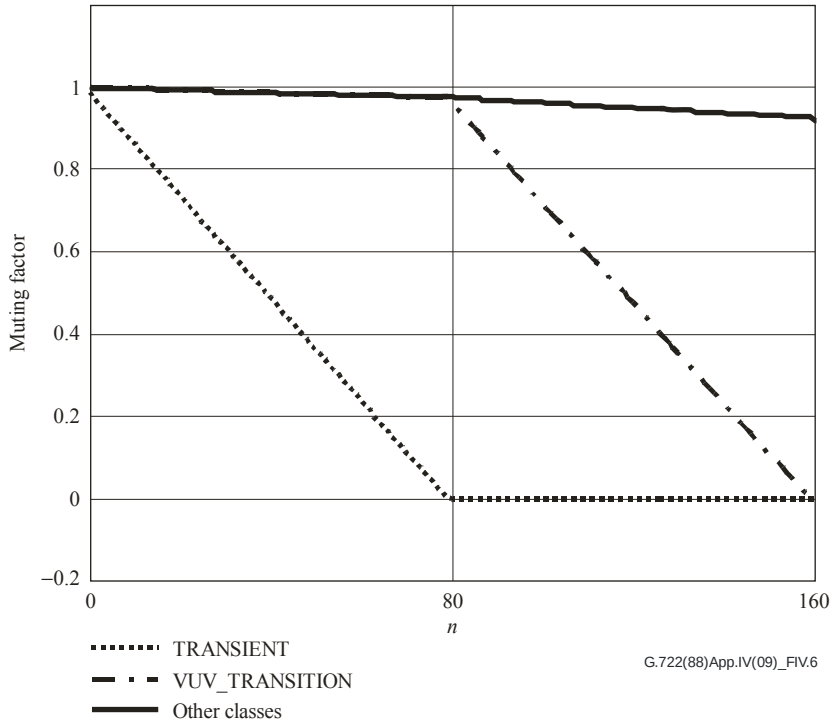


Figure IV.6 – Muting factor as a function of the sample index for 10-ms frames (80 samples for current frame + 80 samples for cross-fading part)

Computation of muting factor for other cases (20-ms frames or consecutive erased 10-ms frames)

In these cases, the muting factor *g_mute_lb* is also adapted sample by sample with the adaptation factor *fac*1, for $n = 0, \ldots, L - 1$. However, *g_mute_lb* may be further decreased depending on the value of *cnt_mute_lb*. The amount of decrease also depends on *class*. The procedure is the following:

$$g\_mute\_lb = g\_mute\_lb - fac1$$

$$\text{if } (cnt\_mute\_lb \geq 80)\ \ g\_mute\_lb = g\_mute\_lb - fac2p$$

$$\text{if } (cnt\_mute\_lb \geq 160)\ g\_mute\_lb = g\_mute\_lb - fac3p$$

$$\text{if } (cnt\_mute\_lb \geq 320)\ g\_mute\_lb = 0$$

$$cnt\_mute\_lb = cnt\_mute\_lb + inc\_mute$$

The muting factor for the extra synthesis needed for cross-fading, $yl_{pre}(n)$, $n = L, \ldots, L + 79$, is muted in the same way for $n = L, \ldots, L + 79$.

NOTE – For the first erased 20-ms frame, *g_mute_lb* and *cnt_mute_lb* have been initialized before the first erasure (see clause IV.6.1.1).

The muting factor adaptation is illustrated in Figure IV.7. The gain values for the first 160 samples are identical to those shown in Figure IV.6.

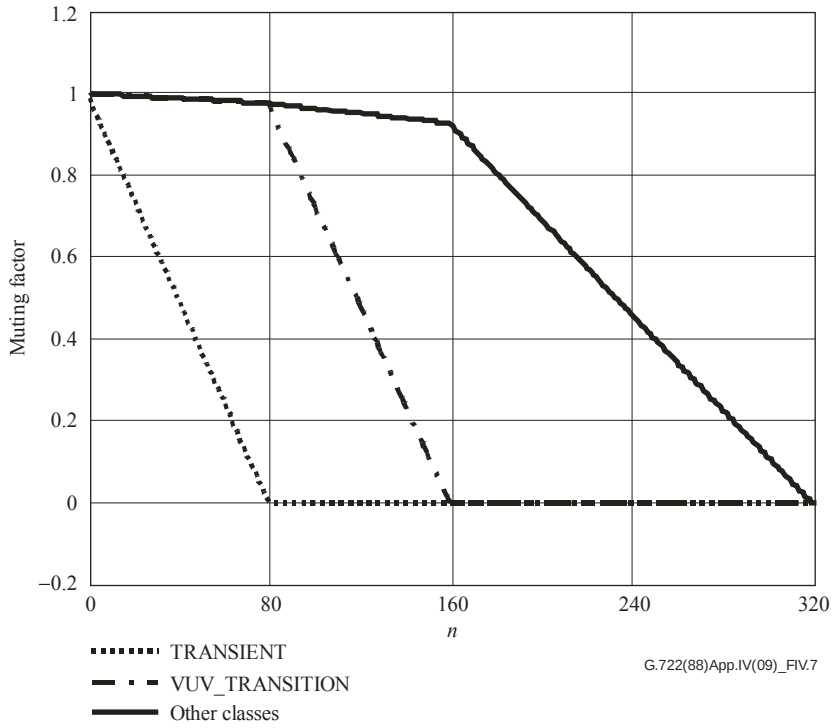


Figure IV.7 – Muting factor as a function of the sample index

IV.6.1.3 Extrapolation of missing frame: Case of bad frame following a bad frame

In the case of a bad frame following a bad frame, the analysis parameters computed for the first erased frame (a_i, i = 1, …, 8, T_0, *class*) are kept. The signal generated in the previous frame for cross-fading is copied to $yl(n)$ n = 0, …, 79. The last L samples, $yl(n)$, n = 80, …, L + 79 including the 10-ms part used for cross-fading with the next frame, are synthesized as described in clause IV.6.1.2.

IV.6.1.4 Update of ADPCM decoder states

The states of the lower-band ADPCM decoder are updated after extrapolating missing frames to help in recovery from frame erasures. This update is more elaborate than a simple ADPCM decoder reset. However, to minimize complexity, the ADPCM states are updated based on available or *a priori* information, without additional processing. The states are modified as follows, using ITU-T G.722 notation:

$$\mathrm{DLT}i = 0,\ i=1,\cdots,6$$

$$\mathrm{PLT}i = \frac{yl(L-i)}{2},\ i=1,2$$

$$\mathrm{RLT1} = yl(L-1)$$

$$\mathrm{SL} = yl(L)$$

$$\mathrm{SZL} = \frac{yl(L)}{2}$$

$$\text{if } cnt\_mute\_hb > 160,$$

$$\mathrm{DETL} = 32$$

$$\mathrm{NBL} = 0$$

IV.6.1.5 Cross-fading

The cross-fading is detailed in Table IV.4. The cross-fading window is based on the Bartlett window (triangular) with a time support of 10 ms.

Table IV.4 – Cross-fading operation

		Current frame	
		Bad	**Good**
Previous frame	**Bad**	$zl(n) = yl(n)$, $n = 0, \ldots, L-1$	$zl(n) = \frac{n}{79} xl(n) + (1 - \frac{n}{79}) yl(n)$, $n = 0, \ldots, 79$ and $zl(n) = xl(n)$, $n = 80, \ldots, L-1$
	Good	$zl(n) = yl(n)$, $n = 0, \ldots, L-1$	$zl(n) = xl(n)$, $n = 0, \ldots, L-1$

IV.6.2 Higher-band decoding

IV.6.2.1 ADPCM decoder in case of a good frame

Same as clauses 4.1, 4.2 and 4.3 of [ITU-T G.722].

In addition, the counter *cnt_mute_hb* and muting factor *g_mute_hb* (used in clause IV.6.2.2.2) for adaptive muting are reset:

$$cnt\_mute\_hb = 0$$

$$g\_mute\_hb = 1$$

and the signal $zh(n)$ is stored to be used in case of erasure in future frames.

IV.6.2.2 Extrapolation of missing frame

The extrapolation of a missing frame in the higher band uses the past signal $zh(n)$, $n = -160, ..., -1$, buffered using a buffer length of 160 samples.

IV.6.2.2.1 Repetition

The extrapolation of a missing frame in the higher band consists of pitch synchronous repeating of the previous signal $zh(n)$ if *class* = VOICED; otherwise, the repetition period is set to 80 samples (10 ms). In other words:

$$yh_{pre}(n) = zh(n - T_h), \; n = 0, \cdots, L-1 \qquad \text{(IV-18)}$$

where $T_h = T_0$ if *class* = VOICED, $T_h = 80$ otherwise.

IV.6.2.2.2 Adaptive muting

As for the lower-band reconstructed signal, the energy of the higher-band reconstructed signal is also controlled by applying a gain factor computed and adapted sample by sample. The synthesized signal $yh_{pre}(n)$ is muted sample by sample with an adaptive muting factor *g_mute_hb* for $n = 0, \ldots, L-1$ to obtain the reconstructed higher-band signal $yh(n)$. The muting of the higher band is identical to the muting of the lower band described in clause IV.6.1.2.7. However, since no cross-fading is used in the higher band, a separate counter *cnt_mute_hb* and muting factor *g_mute_hb* are used as *cnt_mute_hb* is always 80 samples in advance compared to *cnt_mute_lb* after one erasure.

IV.6.2.3 High-pass post-processing

In case of frame erasures, a DC offset of very small magnitude may appear in the higher-band reconstruction $uh(n)$ $n = 0, \ldots, L - 1$ and also affect the first consecutive good frames. After the QMF synthesis, this introduces an 8-kHz component. To avoid this annoying high-frequency noise, a first-order pole/zero filter with a cut-off frequency of 50 Hz is used. This filter is given by:

$$H_{post}(z) = \frac{\frac{7303}{8192}\left(1 - z^{-1}\right)}{1 - \frac{3207}{4096} z^{-1}} \tag{IV-19}$$

The signal $uh(n)$ is filtered through $H_{post}(z)$ to obtain $vh(n)$ where $n = 0, \ldots, L - 1$.

This filter is used during the erased frames and the first 4 s following the erasure.

IV.6.2.4 Update of ADPCM decoder states

Similar to lower-band decoding, the states of the higher-band ADPCM decoder are updated after extrapolating a missing frame. The update is described below using ITU-T G.722 notation:

NBH = NBH/2

DETH = scaleh(NBH)

if $cnt\_mute\_hb > 160$

NBH = 0

DETH = 8

The update is restricted to the higher-band scale factor.

IV.6.3 QMF synthesis filterbank

Same as clause 4.4 of [ITU-T G.722], except that the shift of 22 samples of the delay line after each call of the qmf_rx function (for each two output samples) is replaced by the use of a linear buffer of length L+22 in case of bad frame decoding. The memory part of this buffer is updated at the beginning of each frame, and the filter memory is saved at the end of each frame.

IV.7 Bit-exact description of the ITU-T G.722 PLC algorithm

The current version of the ANSI-C code simulating the candidate ITU-T G.722 PLC algorithm in 16-bit fixed-point is Release 1.2. The following subclauses summarize the use of this simulation code, and how the software is organized. The G722PLC algorithm is implemented using the ITU-T G.722 code from the ITU-T software tool library (STL2005). The mathematical descriptions of the PLC algorithm (clauses IV.5 and IV.6) can be implemented in several other fashions. Therefore, the algorithm description of the ANSI-C code of this clause shall take precedence over the mathematical descriptions of clauses IV.5 and IV.6 whenever discrepancies are found.

IV.7.1 Use of the simulation software

The command line for the ITU-T G.722 decoder with PLC is as follows:

```
decg722 [-fsize N] g192_bst output
```

where N is the frame size at 16 kHz (default: 160).

The output file is a sampled data file containing 16-bit PCM signals.

The mapping table of the encoded bit stream is contained in the simulation software.

Organization of the simulation software

See Tables IV.5 and IV.6.

Table IV.5 – List of tables added by the PLC algorithm

Table name	Size	Description
G722PLC_lpc_win_80	80	LPC window
G722PLC_lag_h	8	Lag window for bandwidth expansion (high part)
G722PLC_lag_l	8	Lag window for bandwidth expansion (low part)
G722PLC_fir_lp	9	Coefficients of low-pass quarter-band decimation filter
G722PLC_b_hp	3	High-pass filter coefficients (numerator)
G722PLC_a_hp	3	High-pass filter coefficients (denominator)

Table IV.6 – List of files (ITU-T G.722 decoder with PLC)

a) Identical files – from ITU-T G.191 STL software

Filename	Description
softbit.c	ITU-T G.722 soft bit handling
g722_com.h	ITU-T G.722 additional header file
softbit.h	ITU-T G.722 soft bit handling
ugstdemo.h	Definitions for UGST demo programs

b) Modified files – from ITU-T G.191 STL software

Filename	Description
decg722.c	ITU-T G.722 decoder interface
g722.c	ITU-T G.722 main decoder routines
g722.h	ITU-T G.722 main header file
funcg722.c	ITU-T G.722 library
funcg722.h	ITU-T G.722 library

c) New files

Filename	Description
g722_plc.c, g722_plc.h	PLC library and headers
g722_plc_tables.c	PLC tables
oper32_b.c, oper32_b.h	Additional basic operators and headers

Appendix V

Mid-side stereo coding

(This appendix does not form an integral part of this Recommendation.)

V.1 Scope

This appendix defines a stereo encoding scheme for ITU-T G.722 Annex B (ITU-T G.722-SWB) mid-side (MS) stereo. By using MS stereo, very low transcoding or down-mix effort between MS-stereo bitstreams and monaural bitstreams is achieved. To ensure interoperability, out-of-band signalling should be separately defined to differentiate MS stereo implementations.

V.2 Description of the mid-side stereo coding

V.2.1 Encoding scheme

The mid-side stereo encoding is realized by left-right (LR) to MS conversion and two ITU-T G.722-SWB encoders, as shown in Figure V.1.

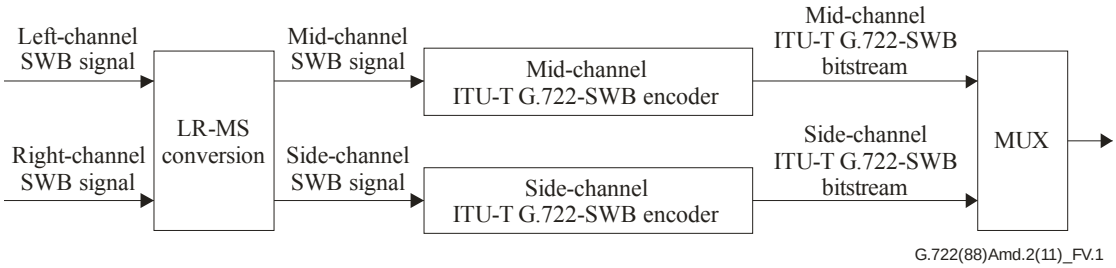


Figure V.1 – Encoder block diagram of MS stereo using ITU-T G.722-SWB

The LR stereo signal is converted into MS stereo using LR-MS conversion and then those two channels are encoded using mid- and side-channel ITU-T G.722-SWB encoders. Those two encoders are identical instances of what is defined in Annex B of [ITU-T G.722].

The LR-MS conversion uses the following two equations.

$$s_{SWB}^{M}(n) = \frac{s_{SWB}^{L}(n) + s_{SWB}^{R}(n)}{2}$$
$$s_{SWB}^{S}(n) = \frac{s_{SWB}^{L}(n) - s_{SWB}^{R}(n)}{2} \qquad n = 0,\ldots,159. \qquad \text{(V-1)}$$

The multiplexer places the mid- and side-channel ITU-T G.722-SWB bitstreams in the order given in Figure V.2.

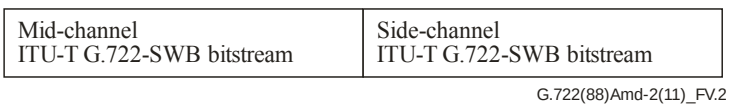


Figure V.2 – Bitstream order of ITU-T G.722-SWB MS stereo

V.2.2 Decoding scheme

The mid-side stereo decoding is realized by MS to LR conversion and two ITU-T G.722-SWB decoders, as shown in Figure V.3.

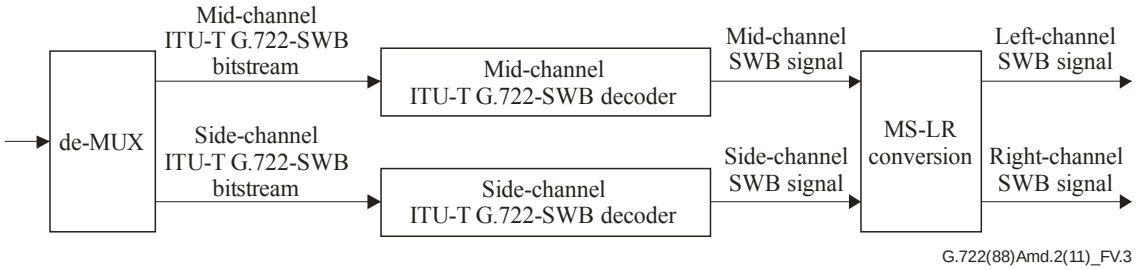


Figure V.3 – Decoder block diagram of MS stereo using ITU-T G.722-SWB

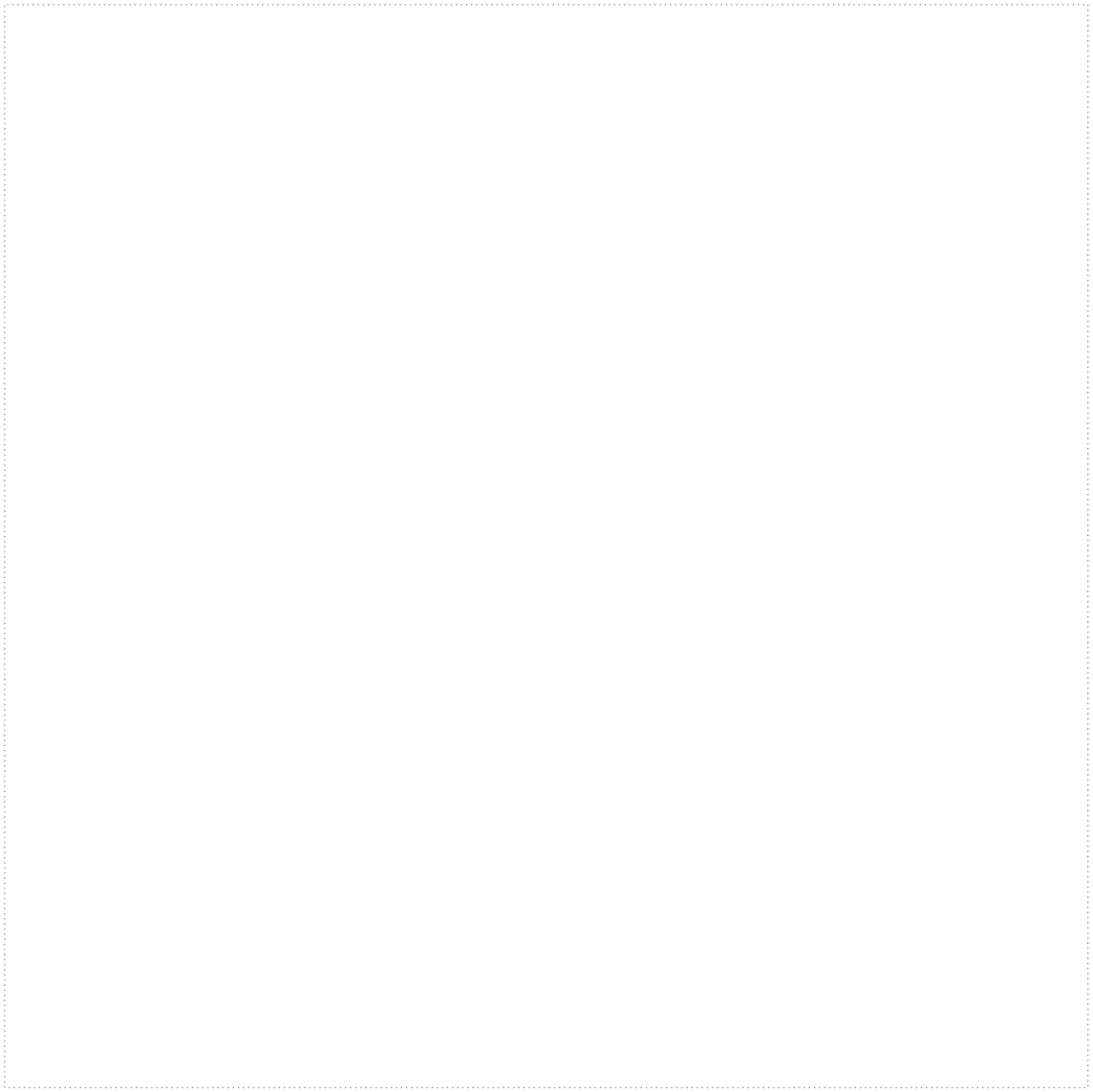
Mid- and side-channel bitstreams are fed to mid- and side-channel ITU-T G.722-SWB decoders and the obtained mid- and side-channel superwideband signals are converted into LR stereo signals using MS-LR conversion. Again, the two decoders are identical instances of what is defined in Annex B of [ITU-T G.722].

The MS-LR conversion uses the following two equations:

$$\begin{aligned} s_{SWB}^{L}(n) &= s_{SWB}^{M}(n) + s_{SWB}^{S}(n) \\ s_{SWB}^{R}(n) &= s_{SWB}^{M}(n) - s_{SWB}^{S}(n) \end{aligned} \qquad n = 0,\ldots,159. \qquad \text{(V-2)}$$

V.3 Computational complexity

The LR-MS conversion in the encoding, described above, requires two arithmetic operations per sample and the MS-LR conversion in the decoding also needs one operation. In an STL2009 [ITU-T G.191] basic operator implementation, the conversion complexity amounts to about 0.2 WMOPS in total.



SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	**Transmission systems and media, digital systems and networks**
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2013