International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.711.1

(03/2008)

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

Digital terminal equipments – Coding of analogue signals by pulse code modulation

---

Wideband embedded extension for G.711 pulse code modulation

Recommendation ITU-T G.711.1

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 analogue signals by pulse code modulation	**G.710–G.719**
Coding of analogue signals by methods other than PCM	G.720–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
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.711.1

Wideband embedded extension for G.711 pulse code modulation

Summary

Recommendation ITU-T G.711.1 describes a G.711 embedded wideband speech and audio coding algorithm operating at 64, 80 and 96 kbit/s.

The encoder input and decoder outputs are sampled at 16 kHz by default, but 8-kHz sampling is also supported. When sampled at 16 kHz, the output of the G.711.1 coder can encode signals with a bandwidth of 50-7000 Hz at 80 and 96 kbit/s, and for 8-kHz sampling the output may produce signals with a bandwidth ranging from 50 up to 4000 Hz, operating at 64 and 80 kbit/s (the bandwidth of the narrow-band signal output from the decoder is characterized by the built-in split-band filterbank which has a frequency cut-offs at 4000 Hz). At 64 kbit/s, G.711.1 is compatible with G.711, hence an efficient deployment in existing G.711-based voice over IP (VoIP) infrastructures is foreseen. The coder operates on 5 ms frames, has a maximum algorithmic delay of 11.875 ms, and has a worst-case computational complexity of 8.70 weighted million operations per second (WMOPS).

The encoder produces an embedded bitstream structured in three layers corresponding to three available bit rates: 64, 80 and 96 kbit/s. The bitstream can be truncated at the decoder side or by any component of the communication system to adjust the bit rate to the desired value, but since it does not contain any information on which layers are contained, an implementation would require outband signalling on which layers are available.

The underlying algorithm has a three-layer coding structure: log companded pulse code modulation (PCM) of the lower band including noise feedback, embedded PCM extension with adaptive bit allocation for enhancing the quality of the base layer in the lower band, and weighted vector quantization coding of the higher band based on modified discrete cosine transformation (MDCT).

Annex A defines an alternative implementation of the G.711.1 algorithm using floating-point arithmetic to facilitate its use on hardware optimized for floating-point operations. The accompanying floating-point C-code is fully interoperable with the fixed-point C-code and provides equivalent quality.

Annex B contains the RTP payload format, capability identifiers and parameters for signalling of G.711.1 capabilities using H.245. The packet format is fully compatible with the corresponding G.711.1 RTP definitions to allow seamless interoperability.

Appendix I describes a supplementary postfilter for use in the decoder. This postfilter enhances the quality of the decoded signal when a legacy G.711 or only the basic log companded PCM part of the G.711.1 bitstream are available. It is intended for end-user terminals and usage in tandem scenarios should be avoided (such as in a signal mixer or bitstream translators).

This Recommendation includes an electronic attachment containing a non-exhaustive set of test signals for use with the ANSI C code.

ANSI C source code is provided for both the main body's fixed-point arithmetic implementation as well as for Annex A's floating-point alternative specification.

This edition incorporates changes needed to correct defects in the pre-published text of this Recommendation. All updates are related to typos and editorial corrections of the text, i.e., the algorithm description, and do not modify the other parts of the Recommendation including Annex A, such as bit-exactness of the supplied C-source code.

Source

Recommendation ITU-T G.711.1 was approved on 15 March 2008 by ITU-T Study Group 16 (2005-2008) under Recommendation ITU-T A.8 procedure.

This edition includes the amendments approved on 13 November 2008 and 16 March 2009 by ITU-T Study Group 16 (2009-2012) under Recommendation ITU-T A.8 procedure.

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/.

CONTENTS

		Page
1	Scope	1
2	References	1
3	Definitions	1
4	Abbreviations and acronyms	1
5	Conventions	2
6	General description of the coder	7
6.1	Encoder	8
6.2	Decoder	8
6.3	Coder modes	9
6.4	Bit allocation	9
6.5	Algorithmic delay	10
6.6	Computational complexity and storage requirements	10
6.7	Coder description	11
6.8	Transcoding between different modes, and legacy G.711	11
7	Functional description of the encoder	12
7.1	Pre-processing high-pass filter	12
7.2	Analysis QMF	12
7.3	Lower-band encoding	14
7.4	MDCT	24
7.5	Higher-band encoding (Layer 2)	26
8	Functional description of the decoder	32
8.1	Embedded lower-band PCM decoder	32
8.2	Higher-band decoder (Layer 2)	35
8.3	Inverse MDCT and overlap-add	36
8.4	Lower-band FERC	37
8.5	Higher-band FERC	51
8.6	Synthesis QMF	54
8.7	Noise gate	55
9	Description of the transmitted parameter indices	56
10	Bit-exact description of the G.711.1 codec	58
10.1	Use of the simulation software	58
10.2	Organization of the simulation software	59
Annex A – Floating-point implementation		62
A.1	Algorithmic description	62
A.2	ANSI C code	62

Page

Annex B – G.711.1 usage in H.245 64
B.1 Scope 64
B.2 Packet structure for G.711.1 frames 64
B.3 G.711.1 capability definitions for H.245 65
B.4 Interoperability with G.711 66
Appendix I – Lower-band postfiltering for R1 mode 67
I.1 Windowing and FFT 68
I.2 PSD of the quantization noise 68
I.3 Postfilter computation 69
I.4 Time domain postfiltering (time OLS with interpolation) 71
I.5 Amplitude modification control 72
I.6 Bit-exact description of the postfilter 72
Electronic attachment – Floating-point reference implementation
– Associated test vectors

Recommendation ITU-T G.711.1

Wideband embedded extension for G.711 pulse code modulation[1]

1 Scope

This Recommendation contains the description of an algorithm extending [ITU-T G.711] for the scalable coding of narrow-band and wideband speech and audio signals from 64 to 96 kbit/s.

This Recommendation is organized as follows. The references, definitions, abbreviations and acronyms, and conventions used throughout this Recommendation are defined in clauses 2, 3, 4, and 5, respectively. Clause 6 gives a general outline of the G.711.1 algorithm. The G.711.1 encoder and decoder principles are discussed in clauses 7 and 8, respectively. The transmitted parameters are presented in clause 9. Clause 10 describes the software that defines this coder in 16-32 bit fixed-point arithmetic. In addition to the algorithm specified in the main body of this Recommendation, Appendix I provides a supplemental functionality enhancing the quality of the decoded signal when a legacy G.711 or only the basic log companded PCM part is available.

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 (2005), *Software tools for speech and audio coding standardization*.

[ITU-T G.711] Recommendation ITU-T G.711 (1988), *Pulse code modulation (PCM) of voice frequencies*.

[ITU-T H.245] Recommendation ITU-T H.245 (2008), *Control protocol for multimedia communication*.

[IETF RFC 3550] IETF RFC 3550 (2003), *RTP: A Transport Protocol for Real-Time Applications*.
<http://www.ietf.org/rfc/rfc3550.txt>

[IETF RFC 5391] IETF RFC 5391 (2008), *RTP Payload format for ITU-T Recommendation G.711.1*.
<http://www.ietf.org/rfc/rfc5391.txt>

3 Definitions

This clause is intentionally blank.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

[1] This Recommendation contains an electronic attachment containing a floating point reference implementation and a non-exhaustive set of test signals for use with the reference implementation.

CSVQ	Conjugate Structured Vector Quantization
DeMUX	Demultiplexer
FER	Frame ERasure
FERC	Frame ERasure Concealment
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
HB	Higher Band
iFFT	Inverse Fast Fourier Transform
iMDCT	Inverse Modified Discrete Cosine Transform
LB	Lower Band
LP	Linear Prediction
LPC	Linear Prediction Coefficient
LSB	Least Significant Bit
LTP	Long-Term Prediction
MDCT	Modified Discrete Cosine Transform
MSB	Most Significant Bit
MUX	Multiplexer
NB	Narrow-Band
NG	Noise Gate
OLA	OverLap Add
OLS	OverLap Save
PCM	Pulse Code Modulation
PSD	Power Spectral Density
QMF	Quadrature-Mirror Filterbank
RMS	Root Mean Square
RSX	Re-synchronized
SNR	Signal to (quantization) Noise Ratio
TDAC	Time Domain Aliasing Cancellation
VQ	Vector Quantization
WB	WideBand
WMOPS	Weighted Million Operations Per Second

5 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., $g^{(m)}(n)$) are used to indicate time-dependency of variables. The variable m refers, depending on the context, to either a frame or subframe index, and the variable n to a sample 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 $\mathrm{sgn}(x)$, where $\mathrm{sgn}(x) = \begin{cases} 1 & if\ x \geq 0 \\ -1 & if\ x < 0 \end{cases}$
- Integer operator $\lfloor x \rfloor$ denotes rounding of x towards minus infinity ($\lfloor x \rfloor = \max\{n \in \{...,-2,-1,0,1,2,...\} \mid x \geq n\}$).
- 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 notated 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., $\lfloor 2^{-N} x \rfloor$.
- In the 16-bit fixed-point ANSI C implementation, the floating-point numbers are rounded to respective 16/32-bit representations.

Table 5-1 lists the most relevant symbols used throughout this Recommendation.

Table 5-1 – Glossary of most relevant symbols

Type	Name	Description
Filters	$F(z)$	Perceptual weighting filter in the lower-band encoder/decoder
	$P(z)$	Lower-band pre-emphasis filter
	$A(z)$	Short term LP analysis filter in lower-band FERC
	$H_{dec}(z)$	Low-pass filter in lower-band FERC
	$B(z/\gamma_{PLC})(z)$	Weighting filter in lower-band FERC
	$H_{pre}(z)$	Pre-processing filter in FERC
	$W_{p2}(k)$	The noise-reduction filter in lower-band postfilter (Appendix I)
	$Wf_{p1}(k)$	The first noise-reduction filter in lower-band postfilter (Appendix I)
	$Wf_{p2}(k)$	The second noise-reduction filter in lower-band postfilter (Appendix I)
Signals	$s_{WB}(n)$	Wideband input signal
	$s_{NB}(n)$	Narrow-band input signal
	$\tilde{s}_{WB}(n)$	Pre-processed wideband input signal
	$\tilde{s}_{NB}(n)$	Pre-processed narrow-band input signal

Table 5-1 – Glossary of most relevant symbols

Type	Name	Description
	$s_{LB}(n)$	Lower-band signal after QMF processing (decimated)
	$s'_{LB}(n)$	Perceptually weighted lower-band target signal
	$d_{L0}(n)$	Lower-band difference signal of $s_{LB}(n)$ and $s'_{LB}(n)$
	$\hat{s}_{L0}(n)$	Decoded signal of Layer 0 bitstream
	$s'_{L0}(n)$	Decoded signal of Layer 0 bitstream, without offset c_{Loff}
	$S''_{L0}(n)$	Pre-emphasized decoded-signal of Layer 0 for LP analysis of lower-band perceptual filter
	$\hat{s}_{L0w}(n)$	Windowed pre-emphasized decoded-signal in LP analysis of perceptual filter
	$r_{L0}(k)$	Autocorrelation function used in the lower-band perceptual filter
	$r'_{L0}(k)$	Lag-windowed autocorrelation function used in the lower-band perceptual filter
	$s'_{LB\exp}(n)$	Lower-band target signal exponent (3-bit precision)
	$s'_{LBref}(n)$	Lower-band target signal's refinement signal (3-bit precision)
	$s_{HB}(n)$	Higher-band signal after QMF processing (decimated)
	$s_{HB}^{\eta}(n)$	Pre-scaled higher-band signal of $s_{HB}(n)$
	$S_{HB}^{TDAC}(k)$	Higher-band MDCT coefficients before post-scaling
	$S_{HB}(k)$	Higher-band MDCT coefficients
	$S_{HBw}(k)$	Weighted higher-band MDCT coefficients
	$\overline{S}_{HBw}(k)$	Weighted and normalized higher-band MDCT coefficients
	$S'_{HB}(v)$	Decimated higher-band MDCT coefficients as CSVQ targets
	$\hat{s}'_{LB\exp}(n)$	Lower-band exponent signal in decoder
	$\hat{s}'_{LB\mathrm{sgn}}(n)$	Lower-band signal sign in decoder
	$\hat{e}_{L1}(n)$	Lower-band enhancement signal in decoder
	$\hat{s}_{L1}(n)$	Layer 1 decoded signal in decoder
	$\hat{s}_{LB}(n)$	Lower-band signal after decoding
	$\hat{\overline{S}}_{HBw}(k)$	Decoded higher-band MDCT coefficients with weighting and normalization
	$\hat{S}_{HBw}(k)$	Decoded higher-band MDCT coefficients with weighting
	$\hat{S}_{HBm}(k)$	Decoded higher-band MDCT coefficients
	$\hat{s}_{HB\_OLA}(n)$	Higher-band signal before OLA in MDCT
	$\hat{s}_{LB}^{pre}(n)$	Pre-processed decoded signal in lower-band FERC

Table 5-1 – Glossary of most relevant symbols

Type	Name	Description
	$r_{LB}(n)$	LP residual signal in lower-band FERC
	$\hat{s}_{LB}^{extr}(n)$	Extrapolated signal in lower-band FERC
	$t(n)$	Decimated signal for pitch estimation in lower-band FERC
	$t_w(n)$	Weighted and decimated signal for pitch estimation in lower-band FERC
	$\hat{s}_{LB}^{extrg}(n)$	Extrapolated signal with gain adjustment in lower-band FERC
	$\hat{s}_{LB}^{RSX}(n)$	Re-synchronized signal in lower-band FERC
	$\hat{s}_{LB}^{SYN}(n)$	Scaled re-synchronized signal in lower-band FERC
	$\hat{s}_{HB}(n)$	Higher-band signal after decoding
	$\hat{s}_{HB,PLCbuff}(n)$	A higher-band signal buffer in higher-band FERC
	$\hat{S}_{HB}(k)$	Higher-band MDCT coefficients after decoding
	$\hat{s}_{WB}(n)$	Decoded wideband signal
	$\hat{S}_{L0}(k)$	Layer 0 decoded signal in frequency domain (Appendix I)
	$\hat{s}'_{L0}(k)$	Output signal of the postfilter (Appendix I)
Parameters	$h_0^{qmf}(i)$	QMF coefficient set 1
	$h_1^{qmf}(i)$	QMF coefficient set 2
	β_e	Lower-band pre-emphasis filter coefficient
	c_{zc1}	Zero-crossing rate calculated in lower-band encoder
	$w_{LP1}(i)$	LP analysis window for lower-band perceptual filter
	$w_{lag}(i)$	LP analysis lag window of lower-band perceptual filter
	c_{Loff}	Core encoder offset value
	γ_{P1}	Perceptual weighting coefficient of lower band
	a_i	LP coefficient of the lower-band perceptual filter
	k_i	Reflection coefficient obtained in lower-band perceptual filter
	η_{LB}	Lower-band signal normalization factor
	α_s	Attenuation factor of the perceptual filter in the lower-band
	$M_{\exp}$	Exponent map of the lower-band signal used for Layer 1 calculation
	$B_A(n)$	Bit-allocation table of Layer 1
	$w_{TDAC}(i)$	Higher-band MDCT overlap window
	η_{HB}^{TDAC1}	Higher-band MDCT normalization factor for pre-scaling
	η_{HB}^{TDAC2}	Higher-band MDCT normalization factor for post-scaling

Table 5-1 – Glossary of most relevant symbols

Type	Name	Description
	η_{HB}	Higher-band MDCT normalization factor
	w_{HB}	Higher-band MDCT coefficient weighting factor
	g_{HB}	RMS value of weighted MDCT coefficients $S_{HBw}(k)$
	ε_g	A value added to avoid zero-divide in gain index calculation of MDCT coefficients
	$I_{Hs0pre}(v,l)$, $I_{Hs1pre}(v,l)$	Pre-selected Channel-0 and -1 candidate indices of *v*-th decimated vector in Layer 2
	$P_{Hs0pre}(v,l)$, $P_{Hs1pre}(v,l)$	Pre-selected Channel-0 and -1 candidate signs of *v*-th decimated vector in Layer 2
	$D_{Hs0pre}(v,l)$, $D_{Hs1pre}(v,l)$	Pre-selected Channel-0 and -1 candidate distortion measure of *v*-th decimated vector in Layer 2
	$g_{H\log}$	RMS value of weighted MDCT coefficients $S_{HBw}(k)$ in u-law domain
	$\hat{g}_{HB}$	Gain value of weighted and normalized MDCT coefficients $\hat{S}_{HBw}(k)$ in decoder
	$\hat{\eta}_{HB}$	Higher-band MDCT normalization factor in decoder
	$w_{LP2}(i)$	LP analysis window for lower-band FERC
	T_{LB}	Lower-band pitch lag obtained in lower-band FERC
	$R_{\max}$	Maximum of the auto-correlation in lower-band FERC
	g_{mute}	Adaptive gain factor in lower-band FERC
	I_{L0}	G.711-compatible core bitstream (Layer 0)
	I_{L1}	Lower-band enhancement bitstream (Layer 1)
	I_{L2}	Higher-band enhancement bitstream (Layer 2)
	γ_{PLC}	Weighting factor in lower-band FERC
	R'_{norm}	Normalized cross-correlation in lower-band FERC
	T_{ds}	Preliminary estimation of the pitch delay in lower-band FERC
	n_0	First zero crossing position of weighted decimated signal in lower-band FERC
	i_0	Temporary index for maximum correlation search in lower-band FERC
	i_1	The first delay in $[1, 35]$ for maximum correlation search in lower-band FERC
	i_2	The lower bound for the maximum correlation search in lower-band FERC
	c_{zc2}	Zero-crossing rate of pre-processed decoded signal in lower-band FERC

Table 5-1 – Glossary of most relevant symbols

Type	Name	Description
	c_{peak}	Number of large peaks detected in last pitch period in lower-band FERC
	$n_{\max r}$	The position of the maximum amplitude sample in the repetition period in lower-band FERC
	$A_{\max r}$	The amplitude of the maximum amplitude sample in the repetition period in lower-band FERC
	A_{meanr}	The mean amplitude of the repetition period in lower-band FERC
	$n_{\max r2}$	A second maximum amplitude position at the other bound of the repetition period in lower-band FERC
	Δ_e	A threshold used for modifications of sample amplitudes in the repetition period in lower-band FERC
	α_{es}	Dynamic energy scaling factor in lower-band FERC
	w_{cf}	A triangular window for cross-fading in lower-band FERC
	T_{HB}	Estimated higher-band pitch lag in higher-band FERC
	R_{HB}	Normalized correlation in higher-band FERC
	R_{HB_MAX}	Maximum correlation in higher-band FERC
	ζ_{HB}	High correlation flag in higher-band FERC
	T_{HBmax}	Maximum pitch lag in higher-band FERC
	T_{HBmin}	Minimal pitch lag in higher-band FERC
	$\alpha^{(n)}_{HBPLC1}$	Sample-by-sample attenuation weighting factor in higher-band FERC
	$\alpha^{(n)}_{HBPLC2}$	An attenuation factor in higher-band FERC
	$\lvert N_{L0}(k)\rvert^2$	A frequency-dependent noise power spectral density (PSD) estimation in lower-band postfiltering
	$w_a(n)$	A Hanning asymmetrical window in lower-band postfiltering
	Γ_{L0}	A load factor in lower-band postfiltering
	$\mathrm{SNR}_{post}(k)$	*A posteriori* SNR for each frequency bin in lower-band postfiltering
	$\mathrm{SNR}^{(m)}_{prio1}(k)$	*A priori* SNR on the current frame in lower-band postfiltering
	$\gamma_w(n)$	First half of a 16-length Hanning window in lower-band postfiltering
	$d'_{\max}$	Allowed maximum distortion in lower-band postfiltering

6 General description of the coder

The G.711 wideband extension codec is implemented in fixed point using basic operators version 2.2 defined in [ITU-T G.191], software tool library. This Recommendation provides the detailed algorithm description.

6.1 Encoder

Figure 6-1 shows the high-level block diagram of the encoder. A pre-processing high-pass filter is applied to the 16-kHz-sampled input signal $s_{WB}(n)$ to remove 0-50 Hz components. The pre-processed signal $\tilde{s}_{WB}(n)$ is divided into 8-kHz-sampled lower-band and higher-band signals, $s_{LB}(n)$ and $s_{HB}(n)$, using a 32-tap quadrature mirror filterbank (QMF). The lower-band signal is encoded with an embedded lower-band encoder which generates G.711-compatible core bitstream (Layer 0) I_{L0} at 64 kbit/s, and lower-band enhancement (Layer 1) bitstream I_{L1} at 16 kbit/s. The higher-band signal is transformed into modified discrete cosine transform (MDCT) domain and the frequency domain coefficients $S_{HB}(k)$ are encoded together with and its normalization factor η_{HB} by the higher-band encoder which generates higher-band enhancement (Layer 2) bitstream I_{L2} at 16 kbit/s. All bitstreams are multiplexed as a scalable bitstream.

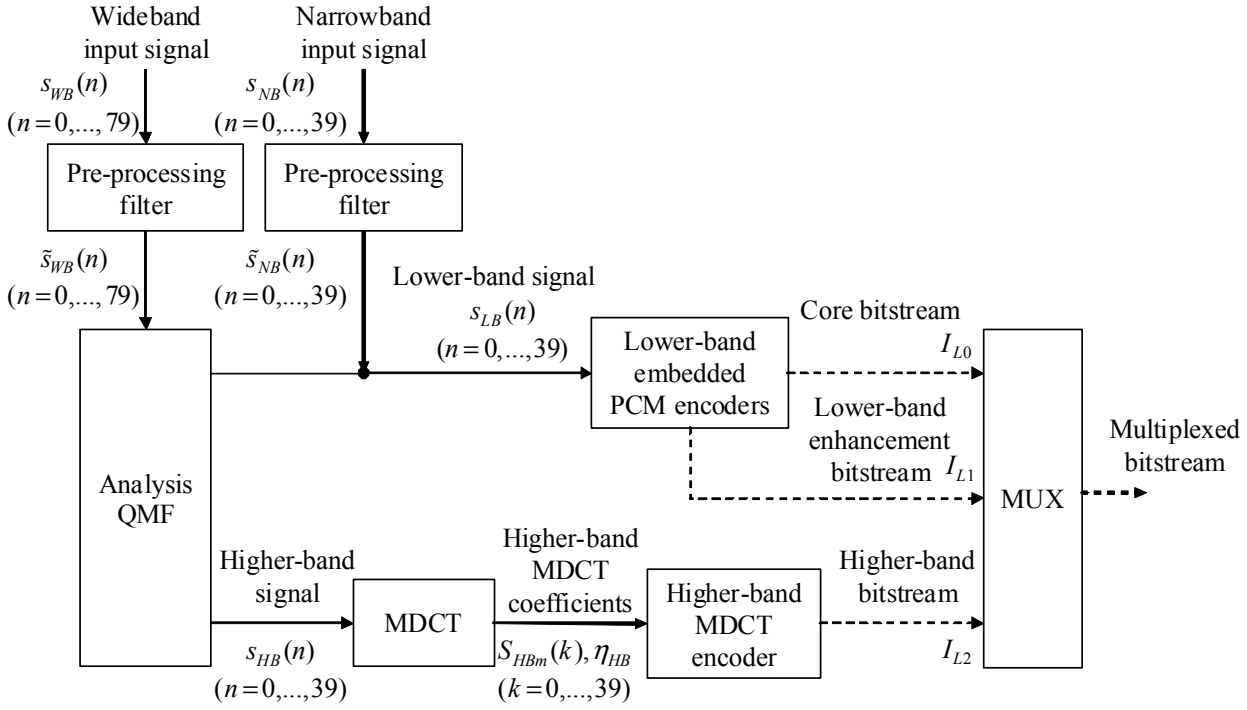


Figure 6-1 – High-level encoder block diagram

6.2 Decoder

Figure 6-2 shows the high-level block diagram of the decoder. The whole bitstream is de-multiplexed to G.711-compatible core (Layer 0) bitstream I_{L0}, lower-band enhancement (Layer 1) bitstream I_{L1}, and higher-band enhancement (Layer 2) bitstream I_{L2}. Both the Layer 0 and 1 bitstreams are handed to the lower-band decoder. The Layer 2 bitstream is given to the higher-band decoder, and consequently decoded signal in the frequency domain $\hat{S}_{HB}(k)$ is fed to inverse MDCT (iMDCT), together with the normalization factor η_{HB}. By this iMDCT process, the higher-band signal in time domain $\hat{s}_{HB}(n)$ is obtained. To improve the quality under frame erasures due to channel errors such as packet-losses, frame erasure concealment (FERC) algorithms are applied to the lower-band and higher-band signals independently. Although the concealment process is performed independently, the pitch lag T_{LB} estimated in the lower-band FERC is given to higher-band FERC as auxiliary information. The lower- and higher-band signals, $\hat{s}_{LB}(n)$ and

$\hat{s}_{HB}(n)$, are combined using a synthesis QMF filterbank to generate a wideband signal $\hat{s}_{QMF}(n)$. Noise gate processing is applied to the QMF output to reduce low-level background noise. At the decoder output, 16-kHz-sampled speech $\hat{s}_{WB}(n)$ (or 8-kHz-sampled speech $\hat{s}_{NB}(n)$) is synthesized.

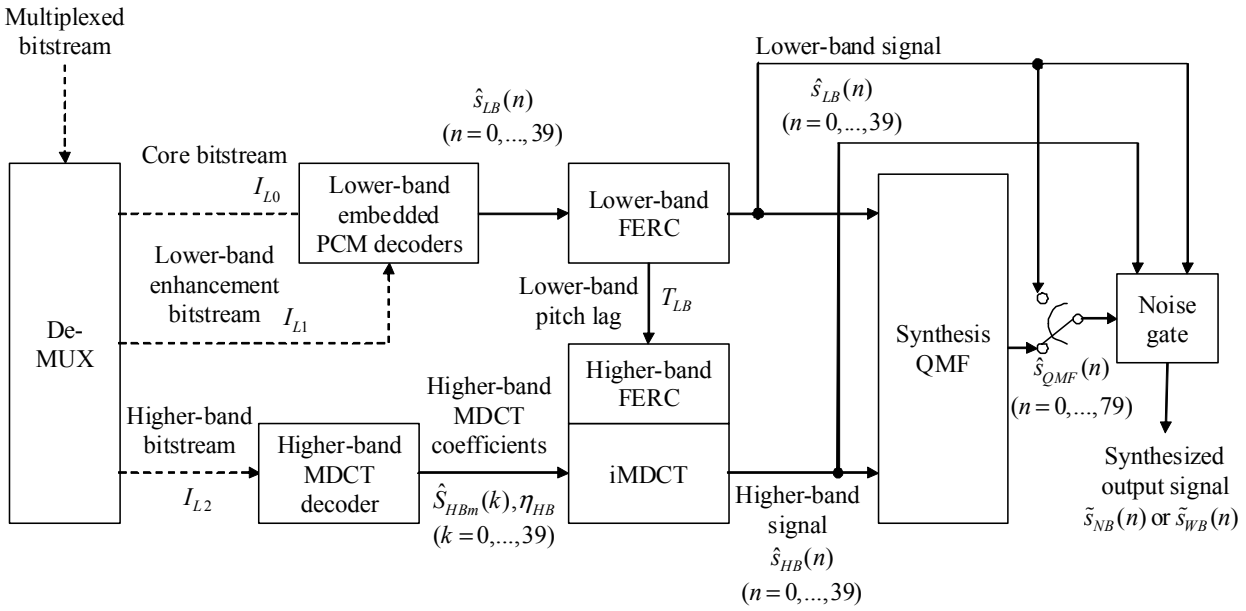


Figure 6-2 – High-level decoder block diagram

6.3 Coder modes

The possible bitstream combinations are shown in Table 6-1. In total, four modes are defined: R1, R2a, R2b and R3, with combinations of bitstreams. It should be noted that since the bitstream itself cannot discriminate between R2a and R2b, an explicit outband signalling of the mode is required.

Table 6-1 – Sub-bitstream combination for each mode

Mode	Sampling rate (kHz)	Core layer (Layer 0, I_{L0})	Lower-band enhancement layer (Layer 1, I_{L1})	Higher-band enhancement layer (Layer 2, I_{L2})	Overall bit rate (kbit/s)
		64 kbit/s	16 kbit/s	16 kbit/s	
R1	8	x	–	–	64
R2a	8	x	x	–	80
R2b	16	x	–	x	80
R3	16	x	x	x	96

6.4 Bit allocation

The bit allocation of the coder is presented in Table 6-2. This table is structured according to the different bitstream layers. For a given bit rate, the bitstream is obtained by concatenation of the contributing layers as described in clause 6.3.

Table 6-2 – Bit allocation per 5 ms frame

Parameter	Symbol	Bits per frame
Layer 0 (G.711-compliant)		
G.711 code	$I_{L0}(0),...,I_{L0}(39)$	8 bits x 40 samples
Layer 0 subtotal		**320 bits**
Layer 1		
G.711 refinement code	$I_{L1}(0),...,I_{L1}(39)$	$\sum_{n=0}^{39} B_A(n),\ 0 \le B_A(i) \le 3$
Layer 1 subtotal		**80 bits**
Layer 2		
MDCT gain	I_{Hg}	8 bits
MDCT VQ index	$I_{Hs0}(0),...,I_{Hs0}(5)$	5 bits x 6 sub-vectors
MDCT VQ index	$I_{Hs1}(0),...,I_{Hs1}(5)$	5 bits x 6 sub-vectors
MDCT vector sign	$I_{Hp0}(0),...,I_{Hp0}(5)$	1 bit x 6 sub-vectors
MDCT vector sign	$I_{Hp1}(0),...,I_{Hp1}(5)$	1 bit x 6 sub-vectors
Layer 2 subtotal		**80 bits**
TOTAL		**480 bits**

6.5 Algorithmic delay

The G.711.1 coder has an algorithmic delay of 11.875 ms (190 samples at 16000 Hz). The delay contributions are listed below:

- 5 ms for input frame.
- 5 ms for the MDCT analysis (lookahead).
- 1.875 ms for the QMF analysis-synthesis filterbank.

6.6 Computational complexity and storage requirements

The observed worst-case complexity of the G.711.1 coder (encoder plus decoder) is 8.70 WMOPS based on the basic operators of ITU-T software tool library STL2005 v2.2 in [ITU-T G.191]. The worst computational complexity is detailed in Table 6-3, and all the figures show the observed worst complexity either in u-law or A-law. The G.711.1 storage requirements in 16-bit words for the cases of with and without the postfilter are given in Tables 6-4 and 6-5, respectively. Note that the RAM figures are based on the arrays which form the dominant part, but not on singular variables. It was found that the number of singular variables were insignificant when compared with the size required by arrays.

Table 6-3 – Worst computational complexity of the G.711.1 coder (WMOPS)

Encoder	Decoder		
5.396	No FER	R1	0.700
		R1 with postfilter	2.679
		R2a	1.181
		R2b	1.852
		R3	2.333
	FER	R1	2.476
		R1 with postfilter	3.049
		R2a	2.476
		R2b	3.304
		R3	3.304

Table 6-4 – Storage requirements of the G.711.1 coder

	Encoder	Decoder
Static RAM (kWords)	0.18	1.50
Scratch RAM (kWords)	0.66	0.70
Data ROM (kWords)	2.21	
Program ROM (number of basic ops)	1943	

Table 6-5 – Increase in the storage requirements of the G.711.1 decoder with R1 postfilter

	Decoder
Static RAM (kWords)	+0.33
Scratch RAM (kWords)	+0.20
Data ROM (kWords)	+0.19
Program ROM (number of basic ops)	+527

6.7 Coder description

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 10, 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 code of clause 10 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.

6.8 Transcoding between different modes, and legacy G.711

In case the bitstream must be transcoded in translators, truncating layer bitstreams (downward transcoding) or adding layers (upward transcoding) can be performed (note R1 mode is compatible with legacy G.711). While downward transcoding can be trivial, upward transcoding can be performed as setting all bits in an added layer to "0". When a respective layer decoder receives a

bitstream with all bits set to zero, the layer output becomes transparent, i.e., the respective enhancement signal becomes 0.

- For Layer 1, if I_{L1} is all 0, the enhancement signal $e_{L1}(n)$ becomes zero and, as a consequence, $\hat{s}_{L1}(n)$ becomes zero (see clause 8.1 for the signal).
- Similarly, if I_{L2} is all 0, the all higher-band MDCT coefficients $\hat{S}_{HBm}(k)$ becomes zero. Note that this does mean that the decoded wideband signal $\hat{s}_{WB}(n)$ results in no power above 4000 Hz because the QMF is not an ideal band-split filterbank.

7 Functional description of the encoder

7.1 Pre-processing high-pass filter

The pre-processing filter applied to the 16-kHz sampled input signal $s_{WB}(n)$ is defined as:

$$\tilde{s}_{WB}(n) = 0.984375 \cdot \tilde{s}_{WB}(n-1) + s_{WB}(n) - s_{WB}(n-1)\ , \qquad n = 0,\cdots,79 \qquad (7\text{-}1)$$

where $\hat{s}_{WB}(n)$ is the filter output. When the input signal is a narrow-band 8-kHz sampled signal, the filtering operation is:

$$\tilde{s}_{NB}(n) = 0.96875 \cdot \tilde{s}_{NB}(n-1) + s_{NB}(n) - s_{NB}(n-1)\,, \qquad n = 0,\cdots,39 \qquad (7\text{-}2)$$

where $\hat{s}_{NB}(n)$ is similarly the filter output. The pre-processing high-pass filters in both cases are designed with a cut-off frequency of 50 Hz.

7.2 Analysis QMF

An analysis QMF, h^{qmfA}, is applied to the high-pass filtered input signal $\hat{s}_{WB}(n)$ in order to split it into two 8-kHz-sampled signals; lower-band signal $s_{LB}(n)$ and higher-band signal $s_{HB}(n)$. The 8-kHz-sampled lower band signal $s_{LB}(n)$ is obtained by filtering the 16-kHz-sampled inputs through a symmetric FIR low-pass filter with 32 coefficients given by:

$$s_{WL}(n) = \sum_{i=0}^{31} h_L^{qmfA}(i)\tilde{s}_{WB}(n-i), \qquad n = 0,\cdots,79 \qquad (7\text{-}3)$$

where $s_{WL}(n)$ is the 16-kHz-sampled lower band signal, and $h_L^{qmfA}(i)$ is the filter coefficient, and then by decimating $s_{WL}(n)$ by a factor of 2, given by:

$$s_{LB}(n) = s_{WL}(2n+1) \qquad n = 0,\cdots,39 \qquad (7\text{-}4)$$

Similarly, the 8-kHz-sampled higher-band signal $s_{HB}(n)$ is obtained by:

$$s_{WH}(n) = \sum_{i=0}^{31} h_H^{qmfA}(i)\tilde{s}_{WB}(n-i), \qquad n = 0,\cdots,79 \qquad (7\text{-}5)$$

$$s_{HB}(n) = s_{WH}(2n+1), \qquad n = 0,\cdots,39 \qquad (7\text{-}6)$$

Since the high-pass and low-pass filter coefficients, $h_H^{qmfA}(i)$ and $h_L^{qmfA}(i)$, have a relationship of:

$$h_H^{qmfA}(i) = (-1)^i h_L^{qmfA}(i)\,, \qquad i = 0,\cdots,31 \qquad (7\text{-}7)$$

In fact, $s_{LB}(n)$ and $s_{HB}(n)$ can be directly computed as follows:

$$s_{LB}(n) = \sum_{i=0}^{15} h_0^{qmf}(i)\tilde{s}_{WB}(2(n-i)) + \sum_{i=0}^{15} h_1^{qmf}(i)\tilde{s}_{WB}(2(n-i)+1)\,, \qquad n = 0,\cdots,39 \qquad (7\text{-}8)$$

$$s_{HB}(n) = -\sum_{i=0}^{15} h_0^{qmf}(i)\tilde{s}_{WB}(2(n-i)) + \sum_{i=0}^{15} h_1^{qmf}(i)\tilde{s}_{WB}(2(n-i)+1) \qquad n = 0,\cdots,39 \qquad (7\text{-}9)$$

where

$$h_0^{qmf}(i) = h_H^{qmfA}(2i)$$
$$h_1^{qmf}(i) = h_L^{qmfA}(2i+1) \qquad i = 0,\cdots,15 \qquad (7\text{-}10)$$

Table 7-1 gives the values of the coefficients h_0^{qmf} and h_1^{qmf} and the frequency response of QMF is given in Figure 7-1.

Table 7-1 – QMF coefficients

i	$h_0^{qmf}(i)$	$h_1^{qmf}(i)$
0	$-6.5064660 \times 10^{-4}$	$-1.3508480 \times 10^{-3}$
1	$-1.2601150 \times 10^{-3}$	4.1581240×10^{-3}
2	1.4272050×10^{-3}	$-9.3636330 \times 10^{-3}$
3	$-1.7219030 \times 10^{-4}$	1.7881950×10^{-2}
4	$-4.1094160 \times 10^{-3}$	$-3.1155320 \times 10^{-2}$
5	1.4468810×10^{-2}	5.2909300×10^{-2}
6	$-3.9244910 \times 10^{-2}$	$-9.9800110 \times 10^{-2}$
7	1.2846510×10^{-1}	4.6645830×10^{-1}
8	4.6645830×10^{-1}	1.2846510×10^{-1}
9	$-9.9800110 \times 10^{-2}$	$-3.9244910 \times 10^{-2}$
10	5.2909300×10^{-2}	1.4468810×10^{-2}
11	$-3.1155320 \times 10^{-2}$	$-4.1094160 \times 10^{-3}$
12	1.7881950×10^{-2}	$-1.7219030 \times 10^{-4}$
13	$-9.3636330 \times 10^{-3}$	1.4272050×10^{-3}
14	4.1581240×10^{-3}	$-1.2601150 \times 10^{-3}$
15	$-1.3508480 \times 10^{-3}$	$-6.5064660 \times 10^{-4}$

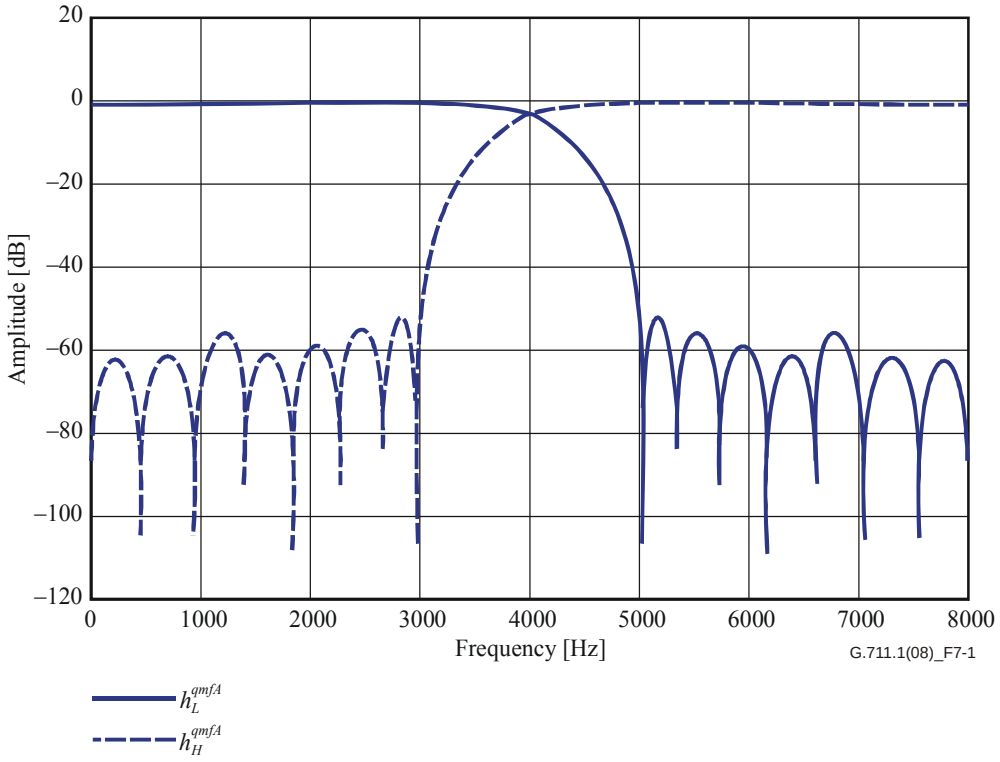


Figure 7-1 – QMF frequency response

When the input is a narrow-band 8-kHz sampling input, this QMF operation is bypassed and the pre-processed input signal is copied to the lower-band signal as:

$$s_{LB}(n) = \tilde{s}_{NB}(n), \qquad n = 0,\cdots,39 \qquad (7\text{-}11)$$

7.3 Lower-band encoding

The lower-band signal $s_{LB}(n)$ is encoded using embedded μ- or A-law pulse coded modulation (PCM) with noise feedback to perceptually shape the coding noise of the PCM encoder. The lower-band encoder with weighted noise feedback loop is shown in Figure 7-2. In the figure, Q_{L0} is the Layer 0 quantizer which is a modified version of the G.711 quantizer and Q_{L1} is the Layer 1 quantizer (which adds a variable number of additional bits of resolution to each sample). Also, Q_{L0}^{-1} is the Layer 0 decoder which is a G.711-compliant decoder. Here, the lower-band input signal $s_{LB}(n)$ is added with a noise feedback signal and an offset value c_{Loff}, and the resulting signal $s'_{LB}(n)$ is fed to the Layer 0 quantizer Q_{L0}. Based on the obtained Layer 0 bitstream $I_{L0}(n)$, Q_{L0}^{-1} locally decodes signal $\hat{s}'_{L0}(n)$. In order to refine the quantization result of G.711, the refinement signal $s'_{LBref}(n)$ and its exponent $s'_{LB\exp}(n)$ are fed to Layer 1 quantizer Q_{L1}, resulting in the adaptively multiplexed bitstream I_{L1}(n). Meanwhile, the perceptual filter $F(z)$ is calculated using $\hat{s}'_{L0}(n)$, and then the quantization noise $d_{L0}(n)$ filtered by $F(z)$ is fed back to be added with the input signal $s_{LB}(n)$. It should be noted that in the Layer 0 encoding, ordinary log-PCM encoding does not apply for input signals with very low energy, and a different encoding scheme called "dead-zone quantizer" is employed as described later in clause 7.3.3. It should also be noted that the noise shaping is applied both in the encoder and the decoder. In the encoder, it is applied to the Layer 0 signal while keeping interoperability with legacy G.711 decoders whereas, in the decoder, it is applied to the Layer 1 signal before addition to the Layer 0 signal. This way, the proper noise shaping is performed not only for Layer 0 but also for Layer 1.

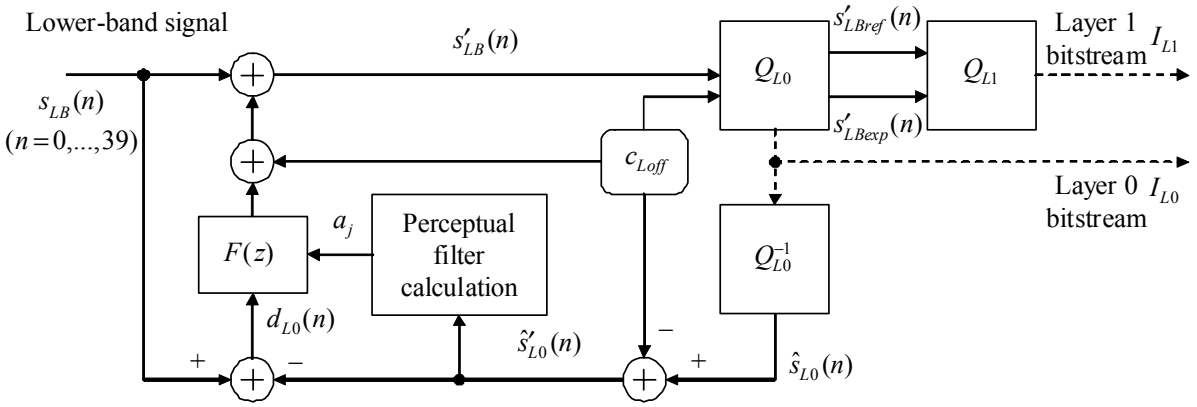


Figure 7-2 – Lower-band encoder with weighted noise feedback loop

7.3.1 Core PCM encoder based on G.711 (Layer 0)

The lower-band encoder processes input signals in two stages: a core PCM encoder, which forms the Layer 0 I_{L0} of the bitstream, followed by an enhancement stage, where *extension bits* are added to form the Layer 1 I_{L1} of the bitstream.

The core PCM encoder is a modified implementation of the legacy G.711 encoder, but such that interoperability with the legacy codec is preserved. Each 16-bit sample $s'_{LB}(n)$ is coded with 8 bits, where 1 bit represents the *sign* (S), three *exponent bits* (E_2, E_1, E_0) indicate a given compander segment, and four *mantissa bits* (M_3, M_2, M_1, M_0) indicate a given position within the compander segment. However, there are some modifications with respect to G.711, which are described in detail in the following two subclauses.

Quantization is performed with either μ-law or A-law. Each frame contains 40 samples which are quantized on a sample-by-sample basis to produce Layer 0 bitstream $I_{L0}(n)$ and Layer 1 bitstream $I_{L1}(n)$, both for n = 0,1,…,39. During the course of encoding, the exponent $s'_{LB\exp}(n)$, extension (or refinement) code $s'_{LBref}(n)$, and locally-decoded signal $\hat{s}_{L0}(n)$ are calculated and stored. The first two values are later used for the Layer-1 adaptive multiplexer (see clause 7.3.4), whereas the third one is used for noise feedback (see clause 7.3.2). Further, the behaviour of the encoder depends on signal conditions which will be explained in clause 7.3.3.

The following equation defines the offset value c_{Loff} depending on the encoding law used:

$$c_{Loff} = \begin{cases} 0 & \text{for } \mu-law \\ 8 & \text{for } A-law \end{cases} \tag{7-12}$$

Since c_{Loff} is 0 for μ-law, denoting that variable may sometimes be omitted in the following descriptions.

7.3.1.1 μ-law encoding process

Given $x = \min\left(\left|s'_{LB}(n)+c_{Loff}\right|, 32635\right)+132$ and the sign given by $s = \begin{cases} 0\text{x}80 & \text{if } s'_{LB}(n) \geq 0 \\ 0 & \text{if } s'_{LB}(n) < 0 \end{cases}$

$$e = \lfloor \log_2(x) \rfloor - 7$$

$$r = \lfloor 2^{-e} \cdot x \rfloor \otimes 0\text{x}07$$

$$m = \lfloor 2^{-(e+3)} \cdot x \rfloor - 16$$

$$\hat{s}_{L0}(n) = \begin{cases} 2^e \cdot \left(2^3(m+16)+4\right) - 132 & \text{if } s = 0\text{x}80 \\ -\left(2^e \cdot \left(2^3(m+16)+4\right) - 132\right) & \text{if } s = 0 \end{cases}$$

$$I_{L0}(n) = \left(s + 2^4 e + m\right) \oplus 0\text{x}7\text{F}$$

Note that s, e, r, and m are temporary variables calculated for each sample, meaning "sign", "exponent", "quantization residual" and "mantissa" of x, respectively. During the course of Layer 0 encoding, the intermediate variables e and r are stored as:

$$s'_{LB\exp}(n) = e$$

$$s'_{LBref}(n) = r$$

and are fed to the adaptive multiplexer described in clause 7.3.4 to generate Layer 1.

7.3.1.2 A-law encoding process

Given $x = \min\left(\left|s'_{LB}(n) + c_{Loff}\right|, 32767\right)$, and the sign $s = \begin{cases} 0\text{x}80 & \text{if } s'_{LB}(n) + c_{Loff} \geq 0 \\ 0 & \text{if } s'_{LB}(n) + c_{Loff} < 0 \end{cases}$

if $|x| > 255$

$$e = \lfloor \log_2(x) \rfloor - 7$$

$$r = \lfloor 2^{-e} \cdot x \rfloor \otimes 0\text{x}07$$

$$m = \lfloor 2^{e+3} \cdot x \rfloor - 16$$

$$\hat{s}_{L0}(n) = \begin{cases} 2^{e-1} \cdot \left((2^4 m + 8) + 256\right) & \text{if } s = 0\text{x}80 \\ -2^{e-1} \cdot \left((2^4 m + 8) + 256\right) & \text{if } s = 0 \end{cases}$$

else

$$e = 0$$

$$r = \left\lfloor \frac{x}{2} \right\rfloor \otimes 0\text{x}07$$

$$m = \lfloor 2^{-4} x \rfloor$$

$$\hat{s}_{L0}(n) = \begin{cases} 2^4 m + 8 & \text{if } s = 0\text{x}80 \\ -\left(2^4 m + 8\right) & \text{if } s = 0 \end{cases}$$

In both conditions, the indices in Layer 0 are calculated with: $I_{L0}(n) = \left(s + 2^4 e + m\right) \oplus 0\text{x}55$.

Note again that *s*, *e*, *r*, and m are temporary variables calculated for each sample, meaning "sign", "exponent", "quantization residual" and "mantissa" of x, respectively. During the course of core layer encoding, the intermediate variables e and r are stored as:

$$s'_{LB\exp}(n) = e$$

$$s'_{LBref}(n) = r$$

and are fed to the adaptive multiplexer described in clause 7.3.4, to generate the Layer 1 bitstream.

7.3.2 Perceptual filtering

7.3.2.1 Calculation of the perceptual filter

The transfer function of the perceptual noise shaping filter is given by:

$$F(z) = A_0(z/\gamma_{P1}) - 1 \tag{7-13}$$

where $A_0(z)$ is the transfer function of a linear prediction (LP) filter and γ_{P1} is a perceptual weighting factor. It can be shown that by using the above filter, the spectrum of the PCM quantization noise is shaped with the spectrum of $1/A_0(z/\gamma_{P1})$. To control the tilt in noise shaping, the filter $A_0(z)$ is computed based on a pre-emphasized signal, with an adaptive pre-emphasis factor. This noise shaping improves the perceptual quality of lower-band encodings, and is applied to Layer 0 at the encoder side, and to Layer 1 at the decoder side (see clause 8.1). In order to avoid filter mismatch between the encoder and the decoder, the filter $A_0(z)$ is computed based on the past decoded signal $\hat{s}_{L0}(n)$, which is also available at the decoder.

The LP filter is computed based on the past decoded signal $\hat{s}_{L0}(n)$ in the range –80,..,–1. Note that the negative indices refer to the past signal (the range –80,..,–1 represents two past frames). It should be recalled that, as indicated in clause 7.3.1, the mantissa value is calculated using the input signal $s'_{LB}(n)$ with the offset c_{Loff}. First, this offset is subtracted from the locally-decoded signal $\hat{s}_{L0}(n)$ as follows:

$$\hat{s}'_{L0}(n) = \hat{s}_{L0}(n) - c_{Loff} \tag{7-14}$$

Then, the samples in the decoded signal $\hat{s}'_{L0}(n)$ are pre-emphasized. 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:

$$\beta_e = 0.38275 + 0.007813 c_{zc1} \tag{7-15}$$

Where c_{zc1} is a zero-crossing rate. The zero-crossing rate is calculated as:

$$c_{zc1} = \frac{1}{2} \sum_{n=-79}^{-1} \left| \mathrm{sgn}[\hat{s}'_{L0}(n-1)] + \mathrm{sgn}[\hat{s}'_{L0}(n)] \right| \tag{7-16}$$

This results in $0.38 < \beta_e < 1.0$. The pre-emphasis signal $\hat{s}''_{L0}(n)$ is obtained as:

$$\hat{s}''_{L0}(n) = \hat{s}'_{L0}(n) - \beta_e \hat{s}'_{L0}(n-1) \qquad n = -80, \cdots, -1 \tag{7-17}$$

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:

$$w_{LP1}(n)=\begin{cases} 0 & n=0 \\ \frac{1}{2}\cos\left(\frac{\pi}{2L_1}\left(n+\frac{1}{2}\right)-\frac{\pi}{2}\right)+\frac{1}{2}\cos^2\left(\frac{\pi}{2L_1}\left(n+\frac{1}{2}\right)-\frac{\pi}{2}\right) & n=1,...,L_1-1 \\ \frac{1}{2}\cos\left(\frac{\pi}{2L_2}\left(n-L_1+\frac{1}{2}\right)\right)+\frac{1}{2}\cos^2\left(\frac{\pi}{2L_2}\left(n-L_1+\frac{1}{2}\right)\right) & n=L_1,...,L_1+L_2-1 \end{cases} \quad (7\text{-}18)$$

where the values L_1=60 and L_2=20 are used. The past pre-emphasized signal $\hat{s}''_{L0}(n)$ is multiplied with this window to obtain the signal $\hat{s}_{L0w}(n)$:

$$\hat{s}_{L0w}(n) = w_{LP1}(n)\hat{s}''_{L0}(n-80) \qquad n=0,\cdots,79 \quad (7\text{-}19)$$

The autocorrelation function of the windowed signal $\hat{s}_{L0w}(n)$ is computed by:

$$r_{L0}(k)=\varepsilon_{L0}(k)+\sum_{n=k}^{79}\hat{s}_{L0w}(n)\hat{s}_{L0w}(n-k) \qquad k=0,\cdots,4 \quad (7\text{-}20)$$

where $\varepsilon_{L0}(k)$ is an initialization value added to each correlation coefficients to ensure a proper shape of the noise shaping filter for signals with low energy, given by $\varepsilon_{L0}(k)=0.95^k\cdot 100^2$. 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:

$$r'_{L0}(k)=\begin{cases} r_{L0}(k) & k=0 \\ w_{lag}(k)r_{L0}(k) & k=1,...,4 \end{cases} \quad (7\text{-}21)$$

The windowing function is defined as:

$$w_{lag}(i)=\frac{1}{1.0001}\exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right] \qquad i=1,\cdots,4 \quad (7\text{-}22)$$

where f_0=120 Hz is the bandwidth expansion and f_s=8000 Hz is the sampling frequency. The multiplication factor 1/1.0001 is for white noise correction for stabilizing 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 $r'_{L0}(k)$ are used to obtain the LP filter coefficients a_k, with the Levinson-Durbin algorithm [b-Levinson]. This algorithm is performed in the following recursive steps:

1) Set iteration number $i=1$, $a_0^{[0]}=1.0$, and $E^{[0]}=r'_{L0}(0)$

2) Compute $k_i=-\frac{1}{E^{[i-1]}}\left(r'_{L0}(i)+\sum_{j=1}^{i-1}a_k^{[i-1]}r'_{L0}(i-j)\right)$

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

4) Compute $a_j^{[i]}=a_j^{[i-1]}+k_i a_{i-j}^{[i-1]}$ 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 $a_j = a_j^{[4]}$, j=1,…,4.

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

$$A_0(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} \tag{7-23}$$

where a_i, i=1,..,4, are the coefficients of the LP filter with order of 4. This function is then weighted using a weighting factor γ_{P1}, as follows:

$$A_0(z/\gamma_{P1}) = 1 + \sum_{i=1}^{4} \gamma_{P1}^i a_i z^{-i} \tag{7-24}$$

Substituting equation 7-23 in equation 7-13, the noise-feedback transfer function becomes:

$$F(z) = A_0(z/\gamma_{P1}) - 1 = \sum_{i=1}^{4} \gamma_{P1}^i a_i z^{-i} \tag{7-25}$$

The weighting factor γ_{P1} is usually set to 0.92, but is changed in some special cases: low level input signal or signal with its spectra tilted towards high-frequencies. The decision to modify the perceptual filter is based on two values: the normalization factor η_{LB} (equation 7-26 below) and the first reflection coefficient k_1 as:

If $\eta_{LB} \geq 16$

Attenuate perceptual filter for low level signals (described in clause 7.3.2.2).

else if $k_1 > 0.9844$

Attenuate perceptual filter for spectrally tilted input signals (described in clause 7.3.2.3).

else

Use the perceptual filter of equation 7-25 with $\gamma_{P1} = 0.92$ (no attenuation).

end.

In order to make the judgement, the normalization factor η_{LB} is calculated with:

$$\eta_{LB} = 30 - \lfloor \log_2(r_{L0}(0)) \rfloor \tag{7-26}$$

where $r_{L0}(0)$ is the first autocorrelation coefficient calculated in equation 7-20. The first reflection coefficient k_1 is obtained in the above Levinson-Durbin algorithm.

7.3.2.2 Attenuation of the perceptual filter for signals with very low level

When the input signal has a very low energy, the noise-shaping may fail to properly mask the quantization noise of G.711. This leads to a power mismatch of the decoded signal due to the noise, and the noise feedback loop may be saturated. To prevent this saturation, the perceptual noise-shaping filter is attenuated for very-low level signals using the following procedure.

When the normalization factor η_{LB} fulfils the condition:

$$\eta_{LB} \geq 16 \tag{7-27}$$

the attenuation for very low level signal is performed. In this case, the attenuated perceptual weighting filter of equation 7-25 becomes:

$$F(z) = \sum_{i=1}^{4} 2^{-(i+\eta_{LB}-16)} a_i z^{-i} \tag{7-28}$$

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. It also helps reducing the effects of filter mismatch between encoder and decoder.

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

In some limited cases, the energy of a signal may be concentrated in a single frequency peak near 4000 Hz (the Nyquist frequency in the lower band). In this specific case, the noise-shaping feedback may become unstable since the filter is highly resonant. This can cause an audible artefact lasting for several consecutive frames until the noise-shaping loop converges to a stable state. To prevent this problem, the noise-shaping feedback 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:

$$k_1 > 0.9844 \tag{7-29}$$

In this case, the perceptual weighting filter of equation 7-25 is used with the weighting factor γ_{P1} defined by:

$$\gamma_{P1} = 0.92\alpha_s \tag{7-30}$$

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

$$\alpha_s = 16 \cdot (1.047 - k_1) \tag{7-31}$$

7.3.2.4 Computation of modified input signal

From the input signal $s_{LB}(n)$ (lower-band signal at the output of the QMF), a modified signal $s'_{LB}(n)$ is obtained as an input to the embedded PCM encoder. The modified signal $s'_{LB}(n)$ is computed by adding to the input signal $s_{LB}(n)$ a weighted function of the quantization noise $d_{L0}(n)$, i.e.:

$$s'_{LB}(n) = s_{LB}(n) + \sum_{i=1}^{4} \gamma_{P1}^{i} a_i \cdot d_{L0}(n-i) + c_{Loff} \tag{7-32}$$

The constant c_{Loff} is again the offset value depending on the companding law of the core encoder as was defined before. It should be noted that the quantization error $d_{L0}(n)$ is computed using only the Layer 0, i.e., does not take into account the contribution from Layer 1. Thus, in the *z*-transform domain, the feedback loop in Figure 7-2 gives a relationship between the input signal $s_{LB}(n)$ and the Layer 0 synthesis $\hat{s}_{L0}(n)$ as:

$$\hat{S}_{L0}(z) = S_{LB}(z) + \left(\frac{1}{A_0(z/\gamma_{P1})} \right) D_{L0}(z) \tag{7-33}$$

where $D_{L0}(z)$ is the z-transform representation of the quantization noise from the Layer 0 quantizer Q_{L0}. It should be noted that using the form $A_0(z/\gamma_{P1})-1$ in the feedback loop filter introduces a one-sample delay, which means that the error sample $d_{L0}(n)$ is delayed and filtered before adding it to $s_{LB}(n)$.

7.3.3 Dead-zone quantizer

To further increase the quality of signal which has very low energy, a dead-zone quantizer is used instead of the embedded low-band encoder and decoder. The dead-zone quantizer is activated only when the following condition is satisfied:

$$\eta_{LB} \geq 16 \text{ and } \begin{cases} s'_{LB}(n) \in [-7,7] & \text{for } \mu-law \\ s'_{LB}(n) - c_{Loff} \in [-11,11] & \text{for } A-law (c_{Loff}=8) \end{cases} \tag{7-34}$$

where η_{LB} is the normalization factor calculated in the low-level signal detection in clause 7.3.2.1. When the condition above is satisfied, the embedded low-band quantizers Q_{L0} and Q_{L1}, and dequantizer Q_{L0}^{-1} are not used, and the following quantization scheme is employed instead.

As seen in condition of equation 7-34, the dead-zone quantizer is activated only for extremely low-level input signal $s'_{LB}(n)$. The interval is called a dead zone and, within this interval, the locally decoded Layer 0 signal $\hat{s}_{L0}(n)$ is suppressed to zero. In this dead-zone quantizer, the samples $s'_{LB}(n)$ are quantized according to the set of equations described below.

7.3.3.1 μ-law encoding process in dead zone

$$s'_{LB\exp}(n) = 0$$

$$s'_{LBref}(n) = \begin{cases} 0 & s'_{LB}(n) \in [-7,-2] \\ 2 & s'_{LB}(n) = -1 \\ 4 & s'_{LB}(n) \in [0,1] \\ 8 & s'_{LB}(n) \in [2,7] \end{cases}$$

$$\hat{s}_{L0}(n) = 0$$

$$I_{L0}(n) = 0\text{xFF}$$

This gives the quantization results as shown in Figure 7-3. The *x*-axis represents the input value to the dead-zone quantizer and the *y*-axis represents the respective decoded output value (note that the results shown in the figure do not take into account the perceptual weighting).

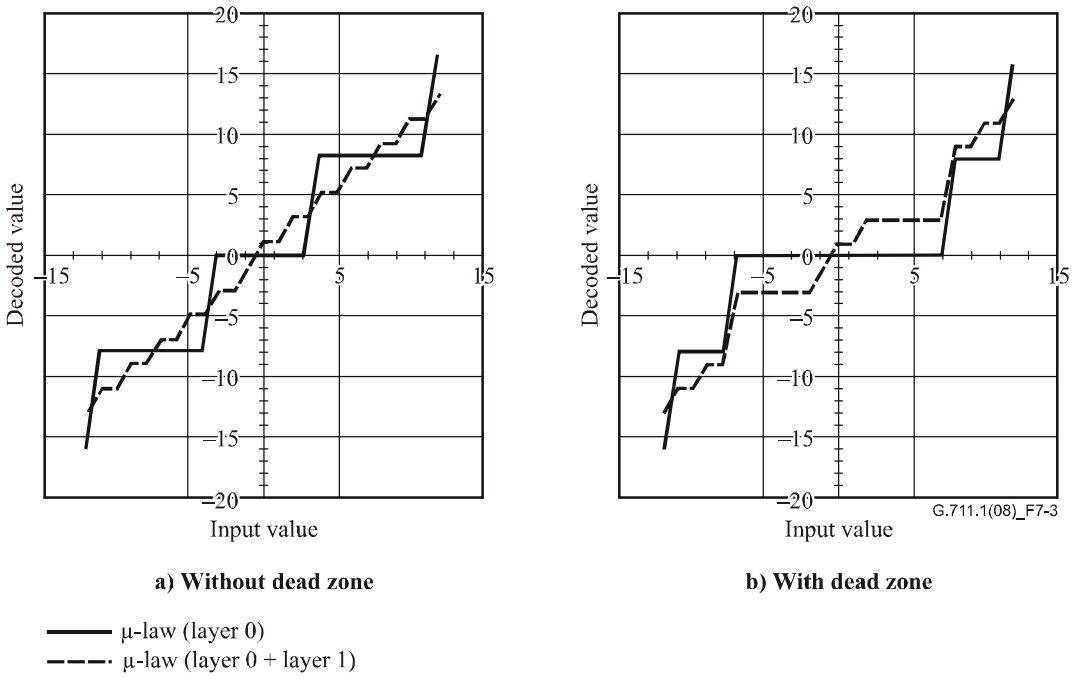


Figure 7-3 – Dead-zone quantizer characteristics (μ-law case)

7.3.3.2 A-law encoding process in dead zone

$$s'_{LB\exp}(n) = 0$$

$$s'_{LBref}(n) = \begin{cases} 0 & s'_{LB}(n) - c_{Loff} \in [-11,-7] \\ s'_{LB}(n)/2 & s'_{LB}(n) - c_{Loff} \in [-6,6] \\ 7 & s'_{LB}(n) - c_{Loff} \in [7,11] \end{cases}$$

$$\hat{s}_{L0}(n) = 0$$

$$I_{L0}(n) = 0\text{xD5}$$

This gives the quantization results as shown in Figure 7-4. Similar to Figure 7-3, the *x*-axis represents the input value to the dead-zone quantizer and the *y*-axis represents the respective decoded output value (the results shown in the figure do not take into account the perceptual weighting).

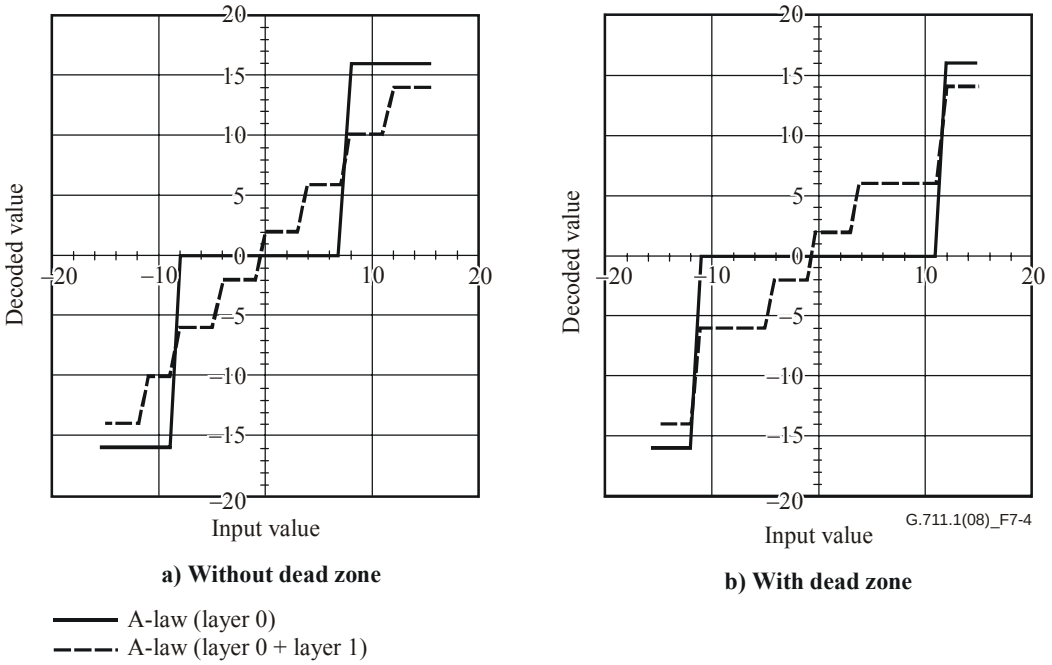


Figure 7-4 – Dead-zone quantizer characteristics (A-law case)

7.3.4 Adaptive multiplexing of refinement signal (Layer 1)

The lower-band enhancement layer (Layer 1) quantizer encodes the refinement signal $s'_{LBref}(n)$ calculated in the core PCM encoder (previously described in clause 7.3.1). The refinement signal $s'_{LBref}(n)$ has a 3-bit resolution per sample. However, since the bit-budget of the Layer 1 bitstream is 16 kbit/s, i.e., 2 bits per sample, an adaptive multiplexing is performed in order to reduce the bits. This adaptive multiplexing dynamically allocates more bits for samples with larger exponent magnitude $s'_{LB\exp}(n)$, which results in allocated bits varying from 0 to 3 bits for each sample. The encoding in this adaptive multiplexing is done in two stages: bit-allocation table generation and refinement signal multiplexing. The procedures are common to both companding laws.

7.3.4.1 Generation of bit allocation table

The bit allocation table is generated using the exponent value $s'_{LB\exp}(n)$. Firstly, an exponent signal is expanded into an exponent map $M_{\exp}(j,n)$, j = 0,…,9 and n = 0,…,39. The exponent map is a (10×40) matrix, which stores the indices of samples that use a specific exponent index j to express the refinement signal $s'_{LBref}(n)$. For counting the number of samples with the same exponent index, an array $N_{\exp}(j)$ (j = 0,…,9), is initialized to 0. For each sample $s'_{LB\exp}(n)$, the following steps are performed for i = 0, 1, 2:

1) Calculate the exponent index of a sample by: $j = s'_{LB\exp}(n) + i$.

2) Update the exponent map as $M_{\exp}(j, N_{\exp}(j)) = n$.

3) Increment the number of samples that fit in the exponent index $N_{\exp}(j) = N_{\exp}(j) + 1$.

Then the bit allocation table $B_A(n)$ (n = 0,…,39) is then calculated using the following steps:

1) Initialize all elements of $B_A(n)$ to 0 (n = 0,…,39), remaining bit budget $b^{[0]} = 80$, set exponent index j = 9, and the iteration number i = 0.

2) Compare remaining bit budget $b^{[i]}$ and number of samples that exist at the current exponent index j, $N_{\exp}(j)$ and the smaller one is set to available bits q in the exponent index, by $q = \min\left[b^{[i]}, N_{\exp}(j)\right]$.

3) For all sample indices in the exponent index j, increment allocated bits of samples at the current exponent index as: $B_A(n) = B_A(n) + 1$, for $n = M_{\exp}(j,k), k = 0,...,q-1$.

4) Update remaining bit budget as: $b^{[i+1]} = b^{[i]} - q$.

5) Check whether remaining bit budget is exhausted, $b^{[i+1]} = 0$. If not, increment i by one, decrement exponent index $j = j - 1$, and go to step 2.

7.3.4.2 Multiplexing

The obtained bit-allocation table $B_A(n)$ (n = 0,…,39) gives the number of most significant bits of $s'_{LBref}(n)$ to be multiplexed. The refinement code $I_{L1}(n)$ is calculated from the $B_A(n)$ MSB of the refinement signal $s'_{LBref}(n)$:

$$I_{L1}(n) = \left\lfloor \frac{s'_{LBref}(n)}{2^{(3-B_A(n))}} \right\rfloor \qquad n = 0,...,39 \qquad (7\text{-}35)$$

The 40 refinement codes $I_{L1}(n)$ (n = 0,…,39) are sequentially multiplexed in the Layer 1 bitstream I_{L1}.

7.4 MDCT

The 8-kHz-sampled higher-band signal $s_{HB}(n)$ is transformed into MDCT coefficients $S_{HB}(k)$. The transform operation is done with a frame length of 5 ms and an analysis window length of 10 ms. The higher-band spectrum $S_{HB}(k)$ in MDCT representation is defined as:

$$s_{HB}(k) = \frac{1}{80}\sum_{n=0}^{79} w_{TDAC}(n)\cos\left(\frac{\pi}{40}(n+20.5)(k+0.5)\right)s_{HB}(n) \qquad k = 0,\cdots,39 \qquad (7\text{-}36)$$

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

$$w_{TDAC}(n) = \sqrt{2}\sin\left(\frac{\pi}{80}(n+0.5)\right) \qquad n = 0,\cdots,79 \qquad (7\text{-}37)$$

In calculating the MDCT coefficients, a variable which holds the fixed-point Q-format is used to deal with variable dynamic range. The normalization parameter is calculated by:

$$\eta_{HB}^{TDAC1} = 14 - \left\lfloor \log_2\left(\max_{n=0,...,79}|s_{HB}(n)|\right)\right\rfloor \qquad (7\text{-}38)$$

The actual computation of the 40 MDCT coefficients $S_{HB}(k)$ is done as follows:

1) Pre-scaling.

First, the scaled signal $s_{HB}^{\eta}(n)$ is computed:

$$s_{HB}^{\eta}(n) = 2^{\eta_{HB}^{TDAC1}} s_{HB}(n) \qquad n = 0,\cdots,79 \qquad (7\text{-}39)$$

2) Windowing and folding.

The real and imaginary parts of the complex numbers $z(n) = z_R(n) + jz_I(n)$, where $j = \sqrt{-1}$, are calculated with:

$$\begin{aligned} z_R(n) &= w_{TDAC}(2n)s^{\eta}_{HB}(2n) - w_{TDAC}(39-2n)s^{\eta}_{HB}(39-2n) \\ z_I(n) &= w_{TDAC}(79-2n)s^{\eta}_{HB}(79-2n) + w_{TDAC}(40+2n)s^{\eta}_{HB}(40+2n) \end{aligned} \tag{7-40}$$

3) Complex pre-multiplication.

The complex signal $z(n)$ is modified to $z'(n)$:

$$z'(n) = W_{80}^{n} \cdot z(n) \qquad n = 0,\cdots,19 \tag{7-41}$$

with $W_N = e^{j\frac{2\pi}{N}}$ is the Nth root of unity, which gives $W_{80}^{n} = \cos\left(\frac{2\pi n}{80}\right) + j\sin\left(\frac{2\pi n}{80}\right)$. This complex pre-multiplication is expanded as:

$$\begin{aligned} z'_R(n) &= \cos\left(\frac{2\pi n}{80}\right) z_R(n) - \sin\left(\frac{2\pi n}{80}\right) z_I(n) \\ z'_I(n) &= \sin\left(\frac{2\pi n}{80}\right) z_R(n) + \cos\left(\frac{2\pi n}{80}\right) z_I(n) \end{aligned} \tag{7-42}$$

where $z'_R(k)$ and $z'_I(k)$ are the real and imaginary parts of $z'(k) = z'_R(k) + jz'_I(k)$.

4) Inverse complex FFT.

The *scaled* inverse complex Fourier transform of $z'(n)$ is computed to obtain the coefficients $Z'(k)$:

$$Z'(k) = \sum_{n=0}^{19} z'(n).W_{20}^{nk} = \sum_{n=0}^{19} z'(n).e^{j\frac{2\pi nk}{20}} \qquad k = 0,\cdots,19 \tag{7-43}$$

This transform of size 20 is implemented using the Good-Thomas FFT algorithm [b-Blahut].

5) Complex post-multiplication.

The resulting coefficients $Z'(k)$ are then modified into:

$$Z(k) = \frac{1}{80}\left((-1)^{k+1} W_8^{-1} W_{320}^{4k+1} \cdot Z'(k)\right) \qquad k = 0,\cdots,19 \tag{7-44}$$

with $W_8^{-1} = \frac{1-j}{\sqrt{2}}$ and $W_{320}^{4k+1} = \cos\left(\frac{2\pi(4k+1)}{320}\right) + j\sin\left(\frac{2\pi(4k+1)}{320}\right)$. This complex post-multiplication can be expanded as:

$$\begin{aligned} Z_R(k) &= \frac{(-1)^{k+1}}{80\sqrt{2}}\left[\cos(\theta)+\sin(\theta)\right]Z'_R(k) - \frac{(-1)^{k+1}}{80\sqrt{2}}\left[\sin(\theta)-\cos(\theta)\right]Z'_I(k) \\ Z_I(k) &= \frac{(-1)^{k+1}}{80\sqrt{2}}\left[\sin(\theta)-\cos(\theta)\right]Z'_R(k) + \frac{(-1)^{k+1}}{80\sqrt{2}}\left[\cos(\theta)+\sin(\theta)\right]Z'_I(k) \end{aligned} \tag{7-45}$$

where $Z_R(k)$ and $Z_I(k)$ are the real and imaginary parts of $Z(k) = Z_R(k) + jZ_I(k)$, with $\theta = \frac{2\pi(4k+1)}{320}$.

6) Reordering.

The MDCT coefficients $S_{HB}(k)$ are initially given by:

$$\begin{cases} S_{HB}^{TDAC}(2k) = Z_I(k) \\ S_{HB}^{TDAC}(39-2k) = -Z_R(k) \end{cases} \qquad k = 0,\cdots,19 \qquad (7\text{-}46)$$

7) Post-scaling.

To improve the accuracy of the subsequent higher-band quantization, another scaling parameter is computed:

$$\eta_{HB}^{TDAC2} = 14 - \left\lfloor \log_2\left(\max_{k=0,\ldots,39} \left| S_{HB}^{TDAC}(k) \right| \right) \right\rfloor \qquad (7\text{-}47)$$

and the MDCT spectrum is finally scaled:

$$S_{HB}(k) = 2^{\eta_{HB}^{TDAC2}} S_{HB}^{TDAC}(k) \qquad k = 0,\cdots,39 \qquad (7\text{-}48)$$

The overall normalization parameter $\eta_{HB} = \eta_{HB}^{TDAC1} + \eta_{HB}^{TDAC2}$ is handed to gain quantization and later used in clause 7.5.2. Note that the value of η_{HB} can be expanded as:

$$\eta_{HB} = 28 - \left(\left\lfloor \log_2\left(\max_{n=0,\ldots,79} \left| s_{HB}(n) \right| \right) \right\rfloor + \left\lfloor \log_2\left(\max_{k=0,\ldots,39} \left| S_{HB}^{TDAC}(k) \right| \right) \right\rfloor \right) \qquad (7\text{-}49)$$

7.5 Higher-band encoding (Layer 2)

For the higher-band codec, an MDCT-based transform coding with interleave conjugate-structure vector quantization (CSVQ) is used. The details of the higher-band encoder are shown in Figure 7-5. The higher-band signal $s_{HB}(n)$ which has been transformed into MDCT coefficients $S_{HB}(k)$ and a respective normalization parameter η_{HB} is applied with a combination of frequency window and weighting $w_{HB}(k)$. The weighted MDCT coefficients $S_{HBw}(k)$ are normalized using a RMS factor g_{HB}. The normalized coefficients $\bar{S}_{HBw}(k)$ are then decimated into 6 sets of 6-sample sub-vectors and those sub-vectors are then independently quantized using two-channel vector quantization (VQ), and this gives the VQ shape indices $I_{Hs0}(v)$, $I_{Hs1}(v)$ and their respective sign indices $I_{Hp0}(v)$ and $I_{Hp1}(v)$. The RMS factor g_{HB} is uniformly scalar quantized in μ-law domain, along with the normalization factor η_{HB}, yielding a gain index I_{Hg}. All indices are then multiplexed and sent as the higher-band bitstream I_{L2}. The bit-allocation of the higher-band coder is shown in Table 7-2.

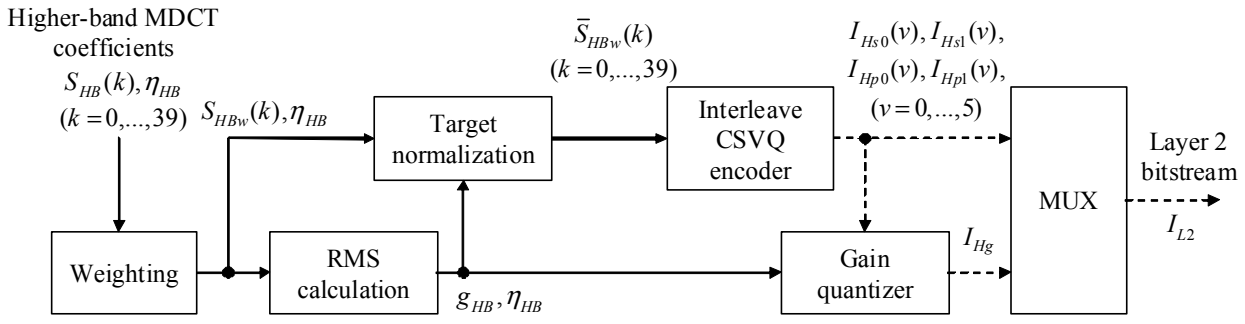


Figure 7-5 – Block diagram of higher-band encoder

Table 7-2 – Bit-allocation of Layer 2 sub-codec

Parameter	Bits per sub-vectors	Bits per 5-ms frame
MDCT coefficients (VQ)	5 + 5	60
Sign	1 + 1	12
Gain	-	8
Total	12	80
Bit-rate	16 kbit/s	

7.5.1 Frequency weighting

The MDCT coefficients $S_{HB}(k)$ are weighted in the frequency domain as follows:

$$S_{HBw}(k) = \begin{cases} 0, & (0 \le k \le 3) \\ w_{HB}(5)\sin\left(\frac{\pi}{24}(2(k-4)+1)\right)S_{HB}(k) & (4 \le k \le 9) \\ w_{HB}(4)S_{HB}(k) & (10 \le k \le 15) \\ w_{HB}(3)S_{HB}(k) & (16 \le k \le 21) \\ w_{HB}(2)S_{HB}(k) & (22 \le k \le 27) \\ w_{HB}(1)S_{HB}(k) & (28 \le k \le 33) \\ w_{HB}(0)S_{HB}(k) & (34 \le k \le 39) \end{cases} \tag{7-50}$$

where $w_{HB}(j)$ are frequency-domain weighting factors. The factors are given in Table 7-3.

Table 7-3 – Frequency-domain weighting factors

j	0	1	2	3	4	5
$w_{HB}(j)$	1	1	4/3	5/3	2	1

7.5.2 RMS calculation and target normalization

In the gain calculation part, the RMS value g_{HB} of the weighted MDCT coefficients $S_{HBw}(k)$ is calculated by:

$$g_{HB} = \sqrt{\frac{1}{36}\sum_{k=0}^{35}(S_{HBw}(k+4))^2} \tag{7-51}$$

The MDCT coefficients $S_{HBw}(k)$ are further normalized with g_{HB} to obtain $\overline{S}_{HBw}(k)$, as follows:

$$\overline{S}_{HBw}(k) = \frac{S_{HBw}(k)}{g_{HB} + \varepsilon_g} \tag{7-52}$$

where ε_g is a value added to avoid zero-divide:

$$\varepsilon_g = 32768 \cdot 2^{-(\eta_{HB}+15)} \tag{7-53}$$

7.5.3 Interleave conjugate-structure vector quantization

Figure 7-6 shows the block diagram of the interleave CSVQ. Firstly, the normalized MDCT coefficients $\overline{S}_{HBw}(k)$ are decimated to construct six sub-vectors $S'_{HB}(v)$, $v = 0,...,5$ of dimension 6. Each sub-vector is then quantized by a two-channel CSVQ, where a codevector is expressed by the

average of two sub-codevectors from two different codebooks. To quantize the v-th sub-vector $S'_{HB}(v)$ by using CSVQ, the code indices $I_{Hs0}(v)$ and $I_{Hs1}(v)$, and the associated polarities (1 or −1) $P_{H0}(v)$ and $P_{H1}(v)$, are selected so as to minimize the distortion $d_{HB}(I_{Hs0}(v), I_{Hs1}(v), P_{H0}(v), P_{H1}(v))$ given below:

$$d_{HB}\left(I_{Hs0}(v), I_{Hs1}(v), p_{H0}(v), p_{H1}(v)\right) = \sum_{j=0}^{5}\left(S'_{HB}(v,j) - \frac{p_{H0}(v)C_{H0w}\left(I_{Hs0}(v),j\right) + p_{H1}(v)C_{H1w}\left(I_{Hs1}(v),j\right)}{2}\right)^2 \qquad \begin{Bmatrix} k=0,...,5, \\ j=0,...,5 \end{Bmatrix} \qquad (7\text{-}54)$$

where $C_{H0w}(I_{Hs0}(v))$ is the codevector with the index $I_{Hs0}(v)$ in the sub-codebook 0, $C_{H1w}(I_{Hs1}(v))$ is the codevector with the index $I_{Hs1}(v)$ in the sub-codebook 1. As shown in Figure 7-6, a pre-selection method for each sub-codebook is introduced before selecting the best combination of the codevectors according to the distortion mentioned above. The details of the pre-selection and the main selection are described in clauses 7.5.3.2 and 7.5.3.3, respectively.

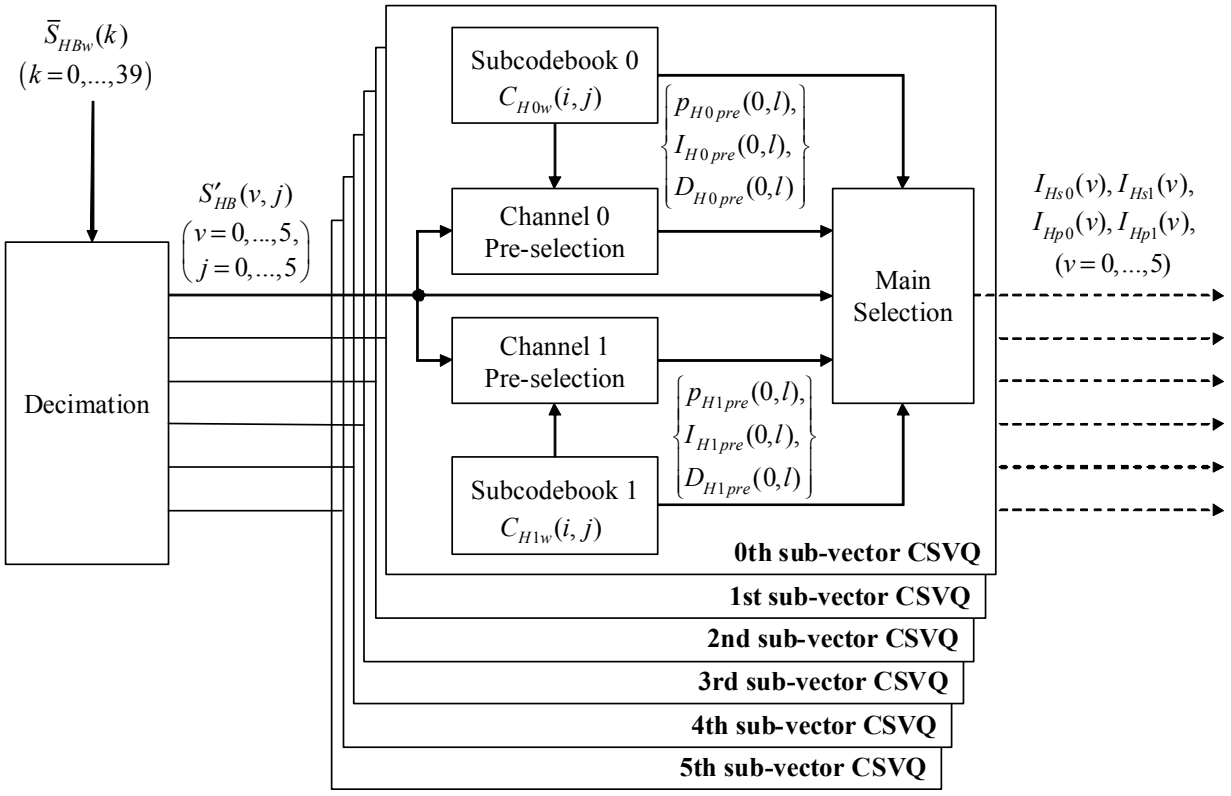


Figure 7-6 – Block diagram of interleave CSVQ

7.5.3.1 Decimation

The weighted and normalized MDCT coefficients $\bar{S}_{HBw}(k)$ are decimated into 6 sub-vectors $S'_{HB}(v)$ of dimension 6 as:

$$S'_{HB}(v,j) = \bar{S}_{HBw}(39 - v - 6j) \qquad V=0,...,5, j=0,...,5 \qquad (7\text{-}55)$$

7.5.3.2 Pre-selection

To reduce the complexity in the codebook search, complexity-reduced pre-selection is performed before costly main code selection. In the pre-selection, for each v-th sub-vector $S'_{HB}(v)$, eight

candidates ($l = 0,...,7$) are selected among each codebook, both of which contain 32 codevectors. Here, codevector candidates with indices $I_{Hs0pre}(v,l)$ and $I_{Hs1pre}(v,l)$, with their respective polarities, $p_{H0pre}(v,l)$ and $p_{H1pre}(v,l)$ are selected.

In this pre-selection stage, the distance $d_{0pre}(v,i_0)$, respectively $d_{1pre}(v,i_1)$, between the target sub-vector $S'_{HB}(v)$ and the codevector $C_{H0w}(i_0)$ in codebook 0, respectively $C_{H1w}(i_1)$ in codebook 1, is calculated for the 32 codevector indices:

$$d_{0pre}(v,i_0) = \sum_{j=0}^{5}\left(S'_{HB}(v,j) - p(v,i_0)\frac{C_{H0w}(i_0,j)}{2}\right)^2$$
$$d_{1pre}(v,i_1) = \sum_{j=0}^{5}\left(S'_{HB}(v,j) - p(v,i_1)\frac{C_{H1w}(i_1,j)}{2}\right)^2 \qquad \begin{Bmatrix} v = 0,...,5 \\ i_0, i_1 = 0,...,31 \end{Bmatrix} \qquad (7\text{-}56)$$

where the polarities $p(v,i_0)$ and $p(v,i_1)$ are set equal to the signs (1 or –1) of $\sum_{j=0}^{5}\left(S'_{HB}(v,j)\cdot C_{H0w}(i_0,j)\right)$ and $\sum_{j=0}^{5}\left(S'_{HB}(v,j)\cdot C_{H1w}(i_1,j)\right)$, respectively.

Here, each codebook is separated into eight ($l = 0,...,7$) segments, and among the four ($q = 0,...,3$) codevectors in a segment, one that minimizes the distance between the *v*-th target sub-vector $S'_{HB}(v)$ is selected as an *l*-th pre-selected candidate. This means that the codebook indices i $(0 \le i \le 31)$ and the candidate indices *l* have the following relationship:

$$i = 4l + q \qquad \begin{Bmatrix} l = 0,...,7, \\ q = 0,...,3 \end{Bmatrix} \qquad (7\text{-}57)$$

In order to reduce the computation, the pre-selection is done to find candidates that *maximize* the following alternative distance criteria d'_{0pre} and d'_{1pre}:

$$d'_{0pre}(v,l,q) = 4\left|\varphi_0(v,l,q)\right| - \sigma^2_{H0w}(4l+q)$$
$$d'_{1pre}(v,l,q) = 4\left|\varphi_1(v,l,q)\right| - \sigma^2_{H1w}(4l+q) \qquad \begin{Bmatrix} v = 0,...,5, \\ l = 0,...,7, \\ q = 0,...,3 \end{Bmatrix} \qquad (7\text{-}58)$$

where

$$\varphi_0(v,l,q) = \sum_{j=0}^{5} S'_{HBw}(v,j)\cdot C_{H0w}(4l+q,j)$$
$$\varphi_1(v,l,q) = \sum_{j=0}^{5} S'_{HBw}(v,j)\cdot C_{H1w}(4l+q,j) \qquad \begin{Bmatrix} v = 0,...,5, \\ l = 0,...,7, \\ q = 0,...,3 \end{Bmatrix} \qquad (7\text{-}59)$$

and

$$\sigma^2_{H0w}(4l+q) = \sum_{j=0}^{5}\left(C_{H0w}(4l+q,j)\right)^2$$
$$\sigma^2_{H1w}(4l+q) = \sum_{j=0}^{5}\left(C_{H1w}(4l+q,j)\right)^2 \qquad \begin{Bmatrix} l = 0,...,7, \\ q = 0,...,3 \end{Bmatrix} \qquad (7\text{-}60)$$

Then the candidate parameter sets $\{P_{H0pre}(v,l), I_{Hs0pre}(v,l), D_{H0pre}(v,l)\}$ and $\{P_{H1pre}(v,l), I_{Hs1pre}(v,l), D_{H1pre}(v,l)\}$ are selected as:

$$\begin{aligned}
&q_{hs0pre}(v,l) = \arg\max\left[d'_{0pre}(v,l,q)\right] \\
&I_{Hs0pre}(v,l) = 4l + q_{Hs0pre}(v,l) \\
&P_{H0pre}(v,l) = \mathrm{sgn}\left[\varphi_0\left(v,l,q_{Hs0pre}(v,l)\right)\right] \\
&D_{H0pre}(v,l) = d'_{0pre}(v,l,q_{Hs0pre}(v,l)) \\
&q_{hs1pre}(v,l) = \arg\max\left[d'_{1pre}(v,l,q)\right] \\
&I_{Hs1pre}(v,l) = 4l + q_{Hs1pre}(v,l) \\
&P_{H1pre}(v,l) = \mathrm{sgn}\left[\varphi_1\left(v,l,q_{Hs1pre}(v,l)\right)\right] \\
&D_{H1pre}(v,l) = d'_{1pre}(v,l,q_{Hs1pre}(v,l))
\end{aligned} \quad \begin{Bmatrix} v = 0,...,5 \\ l = 0,...,7 \end{Bmatrix}. \tag{7-61}$$

It should be noted that codevectors $C_{H0w}(i)$ and $C_{H1w}(i)$ are in fact weighted version of original codevectors: $C_{H0w}(i,j) = w_{HB}(j) \cdot C_{H0}(i,j)$ and $C_{H1w}(i,j) = w_{HB}(j) \cdot C_{H1}(i,j)$, $j = 0,...,5$. Since the weighting is constant, the codevectors are stored in weighted form for complexity reduction. The power of the *i*-th weighted codevectors $\sigma^2_{H0w}(i)$ and $\sigma^2_{H1w}(i)$ are also stored as supplementary codebooks.

7.5.3.3 Main selection

For each subvector, an exhaustive search is performed among all combinations of the two pre-selected parameter sets $\{P_{H0pre}(v,l), I_{Hs0pre}(v,l), D_{H0pre}(v,l)\}$ and $\{P_{H1pre}(v,l), I_{Hs1pre}(v,l), D_{H1pre}(v,l)\}$ to select a combination that minimizes the distance function d_{HB} defined in equation 7-54. By disregarding constant terms, the main selection procedure is simplified so as to *minimize* the following distortion measure:

$$\begin{aligned}
&d'_{HB}(v,l_0,l_1) \\
&= 2p_{H0pre}(v,l_0)p_{H1pre}(v,l_1) \cdot R_{H0H1}\left(I_{Hs0pre}(v,l_0), I_{Hs1pre}(v,l_1)\right) - D_{Hs0pre}(v,l_0) - D_{Hs1pre}(v,l_1)
\end{aligned} \quad \begin{Bmatrix} l_0 = 0,...,7, \\ l_1 = 0,...,7 \end{Bmatrix} \tag{7-62}$$

where $R_{H0H1}(i_0,i_1)$ is given by:

$$R_{H0H1}(i_0,i_1) = \sum_{j=0}^{5} C_{H0w}(i_0,j) C_{H1w}(i_1,j) \tag{7-63}$$

Like previously described $\sigma^2_{H0w}(i)$ and $\sigma^2_{H1w}(i)$, the cross term $R_{H0H1}(i_0,i_1)$ is similarly given as a supplementary codebook. A combination (I_{Hs0}, I_{Hs1}), which gives the minimum distance d'_{HB} is selected as:

$$I_{Hs0}(v) = \underset{I_{Hs0\,pre}(v,l_0)}{\arg\max}\ [d'_{HB}(v,l_0,l_1)]$$
$$I_{Hs1}(v) = \underset{I_{Hs1\,pre}(v,l_1)}{\arg\max}\ [d'_{HB}(v,l_0,l_1)] \qquad \begin{Bmatrix} v=0,...,5, \\ l_0=0,...,7, \\ l_1=0,...,7 \end{Bmatrix} \qquad (7\text{-}64)$$

and the sign indices are obtained as:

$$I_{Hp0}(v) = \begin{cases} 0 & \text{if } p_{H0\,pre}(v, \underset{l_0}{\arg\max}[d'_{HB}(v,l_0,l_1)]) = 1 \\ 1 & \text{otherwise} \end{cases}$$
$$I_{Hp1}(v) = \begin{cases} 0 & \text{if } p_{H1\,pre}(v, \underset{l_1}{\arg\max}[d'_{HB}(v,l_0,l_1)]) = 1 \\ 1 & \text{otherwise} \end{cases} \qquad \begin{Bmatrix} v=0,...,5 \\ l_0=0,...,7 \\ l_1=0,...,7 \end{Bmatrix} \qquad (7\text{-}65)$$

Before the gain scalar quantization (described in clause 7.5.4), the sub-vectors are locally decoded as:

$$\hat{S}'_{HBw}(v,j) = \frac{1}{2}\left(p_{H0}(v)\cdot C_{H0w}(I_{Hs0}(v),j) + p_{H1}(v)\cdot C_{H1w}(I_{Hs1}(v),j)\right) \qquad \begin{Bmatrix} v=0,...,5 \\ j=0,...,5 \end{Bmatrix} \qquad (7\text{-}66)$$

where the polarities are obtained by:

$$\begin{aligned} p_{H0}(v) &= 1 && \text{if } I_{Hp0}(v) = 0 \\ &= -1 && \text{otherwise} \\ p_{H1}(v) &= 1 && \text{if } I_{Hp1}(v) = 0 \\ &= -1 && \text{otherwise} \end{aligned} \qquad v=0,...,5 \qquad (7\text{-}67)$$

7.5.4 Gain scalar quantizer part

In the gain scalar quantizer, an adjusted gain parameter $\tilde{g}_{HB}$ is calculated using locally-decoded sub-vectors $\hat{S}'_{HBw}(v,j)$ as:

$$\tilde{g}_{HB} = \frac{\sum_{v=0}^{5}\sum_{j=0}^{5} S'_{HBw}(v,j)\hat{S}'_{HBw}(v,j)}{\sum_{v=0}^{5}\sum_{j=0}^{5}\left(\hat{S}'_{HBw}(v,j)\right)^2} g_{HB} \qquad (7\text{-}68)$$

where g_{HB} is the RMS value of the input MDCT coefficient previously calculated in clause 7.5.2. The adjusted gain $\tilde{g}_{HB}$ is then scaled as $g_{Hscaled} = 4\sqrt{10}\tilde{g}_{HB}$. The factor $4\sqrt{10}$ is used to scale the value between 0 and 32767. Before the quantization, the scaled gain is converted to the value in μ-law domain, with an exception in the low range as follows, and then scalar quantized:

$$g_{H\log} = \begin{cases} g_{Hscaled} & \text{if } 0 \le g_{Hscaled} < 4 \\ \dfrac{32768\log_{10}\left(1+\dfrac{255 g_{Hscaled}}{32768}\right)}{\log_{10} 256}, & \text{if } 4 \le g_{Hscaled} \le 32767 \end{cases} \qquad (7\text{-}69)$$

Where $g_{H\log}$ is the RMS value in the μ-law domain. This μ-law quantization is performed as:

$$I_{Hg} = \begin{cases} \lfloor g_{Hscaled} \rfloor & \text{if } 0 \le g_{Hscaled} < 4 \\ 2^5 \cdot e_g + \left\lfloor \frac{1}{4}\left(\left\lfloor 2^{-e_g}(g_{Hscaled} + 130) \right\rfloor - 128 \right) \right\rfloor + 3 & \text{if } 4 \le g_{Hscaled} < 30590 \\ 255 & \text{if } 30590 \le g_{Hscaled} \end{cases} \quad (7\text{-}70)$$

where $e_g = \lfloor \log_2(g_{Hscaled} + 130) \rfloor - 7$ is the exponent value of g_{scaled} and I_{Hg} is the gain index of the MDCT signal.

8 Functional description of the decoder

8.1 Embedded lower-band PCM decoder

The embedded PCM decoder is equipped with Layer 0 and Layer 1 decoders, and a perceptual filter calculator and a perceptual noise shaping filter as shown in Figure 8-1.

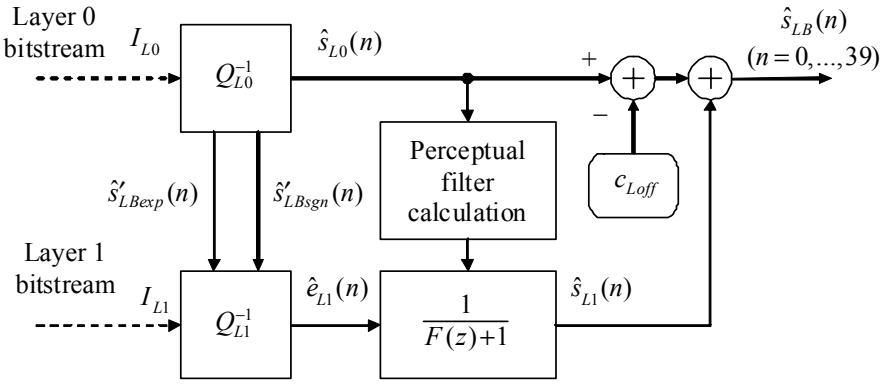


Figure 8-1 – Lower-band embedded PCM decoder

From the Layer 0 bitstream I_{L0}, the Layer 0 decoder (Q_{L0}^{-1}) produces $\hat{s}_{L0}(n)$, in the same manner as the 8-bit PCM G.711 decoder. If Layer 1 bitstream is not available, this signal becomes the output of lower-band decoder, i.e.,

$$\hat{s}_{LB}(n) = \hat{s}_{L0}(n) - c_{Loff} \quad (8\text{-}1)$$

On the other hand, if Layer 1 bitstream I_{L1} is available, the output signal of Q_{L1}^{-1}, $\hat{e}_{L1}(n)$, is shaped with a perceptual weighting filter to obtain the Layer 1 decoded signal $\hat{s}_{L1}(n)$. The transfer function of this perceptual weighting filter is $1/(F(z)+1) = 1/A_0(z/\gamma_{P1})$, and this achieves the same noise shaping as performed in Layer 0 encoding. The shaping operation is:

$$\hat{s}_{L1}(n) = \hat{e}_{L1}(n) - \sum_{i=1}^{4} a_i \gamma_{P1}^i \hat{s}_{L1}(n-i) \quad (8\text{-}2)$$

It should be noted that the perceptual filter $1/A_0(z/\gamma_{P1})$ is identical to the one used in the encoder and is calculated using the decoded signal of the Layer 0 (see clause 7.3.2). Also note that the same attenuation rules apply as in the encoder (see clauses 7.3.2.2 and 7.3.2.3). When Layer 1 bitstream is decoded, the output signal is basically calculated as the summation of the two decoded signals $\hat{s}_{L0}(n)$ and $\hat{s}_{L1}(n)$:

$$\hat{s}_{LB}(n) = \hat{s}_{L0}(n) - c_{Loff} + \hat{s}_{L1}(n) \quad (8\text{-}3)$$

where c_{Loff} is the offset value dependent on the encoding law, as given in equation 7-12. When I_{L0} is missing, this means that the frame was erased, and the whole lower-band decoding, i.e., both Layer 0 and 1, will not be performed and the packet loss concealment routine described in clause 8.4 will generate the lower-band signal instead.

8.1.1 Core PCM decoder based on G.711 (Layer 0)

The core layer (Layer 0) contains G.711-compatible indices $I_{L0}(n)$ for each sample of the frame. If only the Layer 0 is available at the decoder, the reconstruction operates as follows, according to the selected G.711 companding law. As was in the encoder (clause 7.3.1), the temporary variables *s*, *e*, and *m* again mean "sign", "exponent" and "mantissa", respectively. Also, the exponent of the signal is stored as $\hat{s}'_{LB\exp}(n) = e$, together with the sign as $\hat{s}'_{LB\,\mathrm{sgn}}(n) = s$. Both are used to calculate the bit allocation table to obtain the refinement signal $\hat{e}_{L1}(n)$. This is described in clause 8.1.2.

8.1.1.1 μ-law decoding process

Given $s = I_{L0}(n) \otimes 0\text{x}80$ and $y = \left(I_{L0}(n) \oplus 0\text{x}7\text{F}\right) \otimes 0\text{x}7\text{F}$,

$$e = \left\lfloor \frac{y}{2^4} \right\rfloor$$

$$m = y \otimes 0\text{x}0\text{F}$$

$$\hat{s}_{L0}(n) = \begin{cases} 2^e \cdot \left(2^3 m + 128 + 4\right) - 132 & \text{if } s = 0\text{x}80 \\ -\left(2^e \cdot \left(2^3 m + 128 + 4\right) - 132\right) & \text{if } s = 0 \end{cases}$$

$$\hat{s}'_{LB\exp}(n) = e$$

$$\hat{s}'_{LB\,\mathrm{sgn}}(n) = s$$

8.1.1.2 A-law decoding process

Given $s = I_{L0}(n) \otimes 0\text{x}80$ and $y = \left(I_{L0}(n) \oplus 0\text{x}55\right) \otimes 0\text{x}7\text{F}$,

$$e = \left\lfloor \frac{y}{2^4} \right\rfloor$$

$$m = y \otimes 0\text{x}0\text{F}$$

if $e > 0$

$$\hat{s}_{L0}(n) = \begin{cases} 2^{e-1} \cdot \left(2^4 m + 8 + 256\right) & \text{if } s = 0\text{x}80 \\ -\left(2^{e-1} \cdot \left(2^4 m + 8 + 256\right)\right) & \text{if } s = 0 \end{cases}$$

else

$$\hat{s}_{L0}(n) = \begin{cases} 2^4 m + 8 & \text{if } s = 0\text{x}80 \\ -\left(2^4 m + 8\right) & \text{if } s = 0 \end{cases}$$

$$\hat{s}'_{LB\exp}(n) = e$$

$$\hat{s}'_{LB\,\mathrm{sgn}}(n) = s$$

8.1.2 Lower-band enhancement decoder (Layer 1)

In order to generate the refinement signal, the bit allocation table $B_A(n)$ is firstly reconstructed by the same procedure as described in clause 7.3.4.1 using the exponent values $\hat{s}'_{LB\exp}(n)$. This bit allocation table is identical to the one calculated by the encoder. The code $\hat{s}_{LBref}(n)$ for the refinement signal is initialized to 0 and then demultiplexed from the Layer 1 bitstream I_{L1} from the most significant bits, as obtained in $B_A(n)$.

$$\hat{s}'_{LBref}(n) = 2^{4-B_A(n)} I_{L1}(n) \qquad n = 0,...,39 \qquad (8\text{-}4)$$

Then, using the reconstructed $B_A(n)$, $\hat{s}'_{LBref}(n)$ and Layer 0 decoder outputs $\hat{s}'_{LB\,\mathrm{sgn}}(n)$ and $\hat{s}'_{LB\exp}(n)$, the enhancement signal $\hat{e}_{L1}(n)$ is calculated according to the selected G.711 companding law, as follows.

8.1.2.1 μ-law case

Given $s = \hat{s}'_{LB\,\mathrm{sgn}}(n)$, $e = \hat{s}'_{LB\exp}(n)$, $r = \hat{s}'_{LBref}(n)$, and $b = B_A(n)$,

$$m = r - 8 + \left\lfloor \frac{8}{2^b} \right\rfloor$$

if $e > 0$

$$\hat{e}_{L1}(n) = \begin{cases} 2^{e-1} m & \text{if } s = 0\text{x}80 \\ -2^{e-1} m & \text{if } s = 0 \end{cases}$$

else

$$\hat{e}_{L1}(n) = \begin{cases} \left\lfloor \frac{m}{2} \right\rfloor & \text{if } s = 0\text{x}80 \\ -\left\lfloor \frac{m}{2} \right\rfloor & \text{if } s = 0 \end{cases}$$

8.1.2.2 A-law case

Given $s = \hat{s}'_{LB\,\mathrm{sgn}}(n)$, $e = \hat{s}'_{LB\exp}(n)$, $r = \hat{s}'_{LBref}(n)$, and $b = B_A(n)$,

$$m = r - 8 + \left\lfloor \frac{8}{2^b} \right\rfloor$$

if $e > 0$

$$\hat{e}_{L1}(n) = \begin{cases} 2^{e-1} m & \text{if } s = 0\text{x}80 \\ -2^{e-1} m & \text{if } s = 0 \end{cases}$$

else

$$\hat{e}_{L1}(n) = \begin{cases} m & \text{if } s = 0\text{x}80 \\ -m & \text{if } s = 0 \end{cases}$$

It should be noted that there is a special case for decoding of the Layer 1 bitstream I_{L1}. When all bits in the bitstream I_{L1} are 0, this means that the algorithm described above is not performed, and the enhancement signal $\hat{e}_{L1}(n)$ is set to zero as:

$$\hat{e}_{L1}(n) = 0 \qquad n = 0,...,39 \qquad (8\text{-}5)$$

8.2 Higher-band decoder (Layer 2)

The gain-normalized MDCT coefficients are dequantized from the received indices by vector dequantizer. The output MDCT coefficients are reconstructed by multiplying decoded gain and inverse weighting. A block diagram of the higher-band decoder is shown in Figure 8-2.

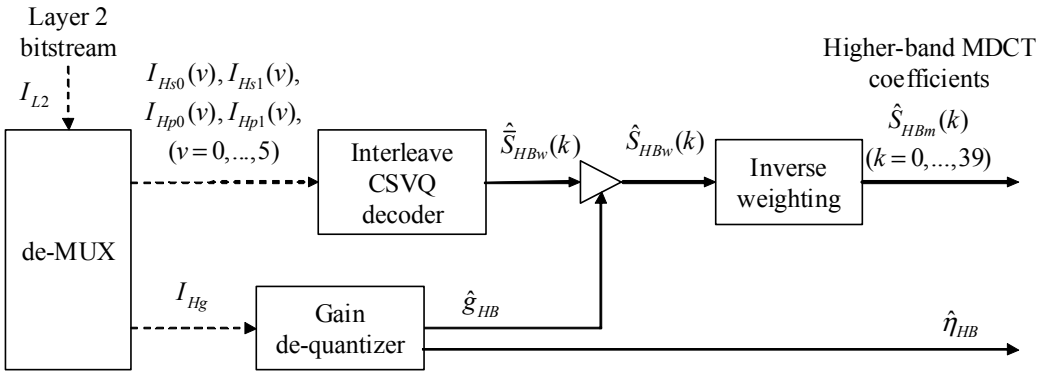


Figure 8-2 – Higher-band MDCT decoder

8.2.1 Interleave vector dequantizer part

8.2.1.1 Vector de-quantizer

The de-quantization of the sub-vectors $\hat{S}'_{HBw}(v,j)$ is identical to what is described in clause 7.5.3.3. Here, the polarities $p_{H0}(v)$ and $p_{H1}(v)$ of the codebook vectors are obtained using equation 7-67. Then the decoded sub-vectors $\hat{S}'_{HBw}(v,j)$ are calculated with equation 7-66.

8.2.1.2 Interleaving

The decoded sub-vectors $\hat{S}'_{HBw}(v,j)$ are interleaved to form a set of weighted MDCT coefficients $\hat{\tilde{S}}_{HBw}(k)$ as follows:

$$\hat{\tilde{S}}_{HBw}(k) = \begin{cases} \hat{S}'_{HBw}\left(39-k-6\left\lfloor\frac{39-k}{6}\right\rfloor, \left\lfloor\frac{39-k}{6}\right\rfloor\right) & \text{for } 4 \le k \le 39 \\ 0 & \text{for } 0 \le k \le 3 \end{cases} \qquad (8\text{-}6)$$

8.2.2 Gain scalar de-quantizer

In the gain scalar dequantizer, decoded gain $\hat{g}_{HB}$ is dequantized from the gain index I_{Hg} as follows:

$$\hat{g}_{HB} = \begin{cases} I_{Hg} & (0 \le I_{Hg} < 4) \\ 2^{\lfloor 2^{-5}(I_{Hg}-3)\rfloor}\left(\left\lfloor 2^{-2}(I_{Hg}-3)\right\rfloor \otimes 0\text{x1F} - 130\right) - 130 & (4 \le I_{Hg} \le 255) \end{cases} \qquad (8\text{-}7)$$

This is equivalent to an 8-bit μ-law decoding. Then the dequantized weighted MDCT coefficients $\hat{\tilde{S}}_{HBw}(k)$ are then multiplied with de-scaled $\hat{g}_{HB}$ (see clause 7.5.4) to obtain $\hat{S}_{HBw}(k)$ as follows:

$$\hat{S}_{HBw}(k) = \frac{\hat{g}_{HB}}{4\sqrt{10}}\hat{\tilde{S}}_{HBw}(k) \qquad k = 0,...,39 \qquad (8\text{-}8)$$

8.2.3 Inverse weighting

Since the reproduced MDCT coefficients $\hat{S}_{HBw}(k)$ are weighted, the unweighted MDCT coefficients $\hat{S}_{HBm}(k)$ are calculated by multiplying $\hat{S}_{HBw}(k)$ by the inverse of the weighting factor $w_{HB}(k)$ (see clause 7.5.1) as:

$$\hat{S}_{HBm}(k) = \begin{cases} 0 & (0 \le k \le 3) \\ \dfrac{1}{w_{HB}(5)} \sin\left(\dfrac{\pi}{24}(2(k-4)+1)\right) \hat{S}_{HBw}(k) & (4 \le k \le 9) \\ \dfrac{1}{w_{HB}(4)} \hat{S}_{HBw}(k) & (10 \le k \le 15) \\ \dfrac{1}{w_{HB}(3)} \hat{S}_{HBw}(k) & (16 \le k \le 21) \\ \dfrac{1}{w_{HB}(2)} \hat{S}_{HBw}(k) & (22 \le k \le 27) \\ \dfrac{1}{w_{HB}(1)} \hat{S}_{HBw}(k) & (28 \le k \le 33) \\ \dfrac{1}{w_{HB}(0)} \hat{S}_{HBw}(k) & (34 \le k \le 39) \end{cases} \quad \text{(8-9)}$$

It should be noted that for samples of $4 \le k \le 9$, the same sin() function is multiplied as in the encoder, and this has an effect that the attenuation result of the input signal becoming $\sin^2()$.

8.3 Inverse MDCT and overlap-add

The reconstructed higher-band MDCT spectrum $\hat{S}_{HBm}(k)$ is transformed to time domain by inverse MDCT transform:

$$\hat{s}^{(m)}_{HB\_OLA}(n) = \sqrt{2} \sum_{k=0}^{39} \cos\left(\frac{\pi}{40}(k+0.5)(n+20.5)\right) \hat{S}_{HBm}(k) \quad \text{(8-10)}$$

where $\hat{s}^{(m)}_{HB\_OLA}(n)$ is an intermediate inverse-transformed signal of the current *m*-th frame before overlap-and-add (OLA).

The actual computation of the 40 coefficients $\hat{s}^{(m)}_{HB\_OLA}(n)$ is done as follows.

1) Complex pre-multiplication.

The real signal $\hat{S}_{HBm}(k)$ is modified to $V(k)$, $k = 0,...,19$, by a complex pre-multiplication which can be expanded as:

$$\begin{aligned} V_R(k) &= -(-1)^{k+1}\sqrt{2}[\cos(\theta)+\sin(\theta)]\hat{S}_{HBm}(39-2k) + (-1)^{k+1}\sqrt{2}[\sin(\theta)-\cos(\theta)]\hat{S}_{HBm}(2k) \\ V_I(k) &= (-1)^{k+1}\sqrt{2}[\sin(\theta)-\cos(\theta)]\hat{S}_{HBm}(39-2k) + (-1)^{k+1}\sqrt{2}[\cos(\theta)+\sin(\theta)]\hat{S}_{HBm}(2k) \end{aligned} \quad \text{(8-11)}$$

where $V_R(k)$ and $V_I(k)$ are the real and imaginary parts of $V(k) = V_R(k) + jV_I(k)$, with $\theta = \dfrac{2\pi(4k+1)}{320}$.

2) Complex FFT.

The complex FFT of $V(k)$ is computed to obtain the coefficients $v(n)$:

$$v(n)=\sum_{k=0}^{19}V(k).e^{-j\frac{2\pi nk}{20}} \qquad k=0,\cdots,19 \qquad (8\text{-}12)$$

This complex FFT of size 20 is implemented using the Good-Thomas algorithm.

3) Complex post-multiplication.

The complex signal $v(n)$ is modified to $v'(n)$ by a complex post-multiplication which is expanded as:

$$\begin{aligned} v'_R(n)&=\cos\left(\frac{2\pi n}{80}\right)v_R(n)+\sin\left(\frac{2\pi n}{80}\right)v_I(n)\\ v'_I(n)&=-\sin\left(\frac{2\pi n}{80}\right)v_R(n)+\cos\left(\frac{2\pi n}{80}\right)v_I(n)\end{aligned} \qquad (8\text{-}13)$$

where $v(n)=v_R(n)+jv_I(n)$ and $v'(n)=v'_R(n)+jv'_I(n)$

4) Reordering.

The time-domain signal $\hat{s}^{(m)}_{HB\_OLA}(n)$ is given by:

$$\begin{cases} \hat{s}^{(m)}_{HB\_OLA}(n)=v_R(n) \\ \hat{s}^{(m)}_{HB\_OLA}(39-n)=-v_R(n) \\ \hat{s}^{(m+1)}_{HB\_OLA}(n)=v_I(n) \\ \hat{s}^{(m+1)}_{HB\_OLA}(39-n)=v_I(n) \end{cases} \qquad n=0,\cdots,19 \qquad (8\text{-}14)$$

Then OLA is performed by:

$$\hat{s}_{HB}(n)=\frac{w_{TDAC}(n)\hat{s}^{(m)}_{HB\_OLA}(n)}{2^{\hat{\eta}_{HB}}}+\frac{w_{TDAC}(39-n)\hat{s}^{(m-1)}_{HB\_OLA}(n)}{2^{\hat{\eta}^{prev}_{HB}}} \qquad n=0,\cdots,39 \qquad (8\text{-}15)$$

where $\hat{\eta}_{HB}$ and $\hat{\eta}^{prev}_{HB}$ are the decoded scaling parameter in the current and previous frames (respectively), $w_{TDAC}(n)$ is the synthesis OLA window identical to the analysis window, and $s^{(m-1)}_{HB\_OLA}(n)$ is the latter part of the intermediate inverse-transformed signal of the previous (m-1)-th frame before OLA, which is updated after the OLA as:

$$\hat{s}^{(m-1)}_{HB\_OLA}(n)=\hat{s}^{(m+1)}_{HB\_OLA}(40+n) \qquad n=0,\cdots,39 \qquad (8\text{-}16)$$

It should be noted that in a case of bad frames, such as in a frame erasure or packet loss, the above OLA is replaced by the higher-band FERC process as described in clause 8.5.

8.4 Lower-band FERC

When a frame is erased, the lower-band signal is reconstructed by a lower-band FERC. In Figure 8-3, the high-level block diagram of the FERC algorithm is given. In case of frame erasure(s), the decoder performs an analysis of the past low band reconstructions and extrapolates the missing signal of the erased frame(s) using linear-predictive coding (LPC) based pitch repetition

and adaptive muting (clauses 8.4.1 and 8.4.2). If a good frame is received, the extrapolated signal that replaced the last erased frame is resynchronized with the properly decoded signal and is cross-faded with the decoded signal (clause 8.4.3). Before cross-fading, dynamic energy scaling is performed on the extrapolated signal (clause 8.4.4).

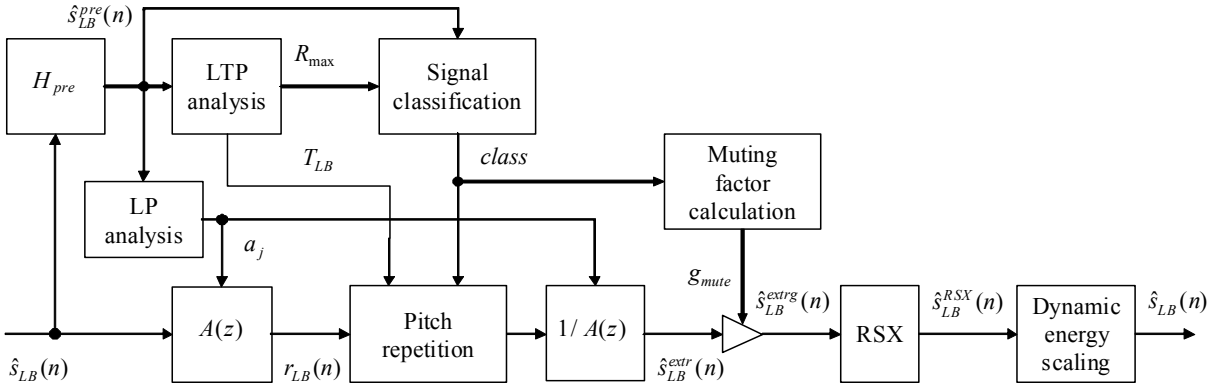


Figure 8-3 – Block diagram of lower-band extrapolation of erased frame

Note that the worst-case complexity occurs at the first erased frame after a valid frame. To reduce the worst-case complexity, the FERC procedure for the first erased frame is distributed among two processing stages executed in two successive frames. This is possible due to the one-frame look-ahead in the higher band. The first step is a preparation step which is executed at the first erased frame. This stage only performs part of the analysis (LPC and long-term prediction analysis) and does not generate any missing samples. The second stage, which is performed at the successive frame (either erased or not), completes the analysis (classification) and produces the missing samples of the signal corresponding to the first erased frame.

8.4.1 Extrapolation of erased frames: Case of erased frame after a good frame

The lower-band extrapolation of an erased frame comprises an analysis of the past valid decoded lower-band signal $\hat{s}_{LB}(n)$, $n < 0$, followed by a synthesis of the signal $\hat{s}_{LB}^{PLC}(n)$, $n = 0,\cdots,39$. The erased 5-ms frame in the lower band is denoted by $\hat{s}_{LB}(n)$.

The past signal $\hat{s}_{LB}(n)$, $n = -295,\cdots,-1$, is buffered using a buffer length of 295 samples, which can be divided as follows:

- 288 samples corresponding to twice the maximal pitch delay (2 x 144) used in the FERC algorithm;
- one sample for pitch jitter; and
- six samples used for LPC memory.

This buffer length allows to store the last two pitch periods of the lower-band reconstruction.

In case of frame erasure, the past signal $\hat{s}_{LB}(n)$ is first pre-processed by a high-pass filter H_{pre} to obtain the pre-processed signal $\hat{s}_{LB}^{pre}(n)$ (clause 8.4.1.1). Then, an LP analysis (clause 8.4.1.2) is performed to obtain a short-term analysis filter $A(z)$. A long-term prediction (LTP) analysis (clause 8.4.1.3) is also performed to obtain a pitch delay T_{LB} and the associated maximum normalized correlation $R_{\max}$. This is followed by a signal classification (clause 8.4.1.4), which determines the signal class (denoted as *class*). The past signal $\hat{s}_{LB}(n)$ is inverse filtered by the filter $A(z)$ to obtain the past excitation signal $r_{LB}(n)$, $n < 0$. Based on results of the LTP analysis and of

the signal classification, the excitation signal $r_{LB}(n)$ ($n = 0,\cdots,39$) for the erased frame is extrapolated by long-term prediction (pitch repetition, clause 8.4.1.5) from the past part of the excitation signal, $r_{LB}(n)$, $n < 0$. This extrapolated excitation signal $r_{LB}(n)$, $n = 0,\cdots,39$, signal is the input of the LP synthesis (clause 8.4.1.6) that gives the unweighted estimated missing decoded signal $\hat{s}_{LB}^{extr}(n)$. The final estimated missing decoded signal $\hat{s}_{LB}^{PLC}(n)$ is obtained after adaptive muting (clause 8.4.1.7) where the muting function $g_{mute}(n)$ depends on the signal class.

8.4.1.1 Pre-processing

The past decoded signal $\hat{s}_{LB}(n)$, $n = -295,...,-1$, is processed by first-order high-pass filter with a cut-off frequency of 50 Hz:

$$H_{pre}(z) = \frac{1 - z^{-1}}{1 - 0.97\,z^{-1}} \tag{8-17}$$

to obtain $\hat{s}_{LB}^{pre}(n)$, $n = -295,...,-1$.

8.4.1.2 LP analysis

The short-term analysis and synthesis filters, $A(z)$ and $1/A(z)$, are based on 6th-order linear prediction (LP) filters. The LP analysis filter is defined as:

$$A(z) = 1 + a_1\,z^{-1} + a_2\,z^{-2} + \cdots + a_6\,z^{-6} \tag{8-18}$$

The LP analysis consists of two parts: windowing and autocorrelation computation, and Levinson-Durbin algorithm. The autocorrelation computation is identical to clause 7.3.2, except that for the LP window $w_{LP1}(i)$ is now an asymmetrical Hamming window, $w_{LP2}(i)$ defined as:

$$w_{LP2}(i) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{(i+80)\pi}{69}\right), & i = -80,\cdots,-11 \\ 0.54 + 0.46\cos\left(\dfrac{(i+11)\pi}{10}\right), & i = -10,\cdots,-1 \end{cases} \tag{8-19}$$

and the lag window is now given by:

$$w_{lag}^{FERC}(i) = \frac{1}{1.0001}\exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right] \qquad i = 1,\cdots,6 \tag{8-20}$$

where f_0 = 60 Hz is the bandwidth expansion and f_s = 8000 Hz is the sampling frequency. Note that the 40 dB white noise correction is unchanged.

This window $w_{LP2}(i)$, which is limited to 80 samples (10 ms at 8-kHz sampling frequency) to reduce complexity, is applied to the last 10 ms of $\hat{s}_{LB}^{pre}(n)$, $n = -80,...,-1$. The Levinson-Durbin algorithm is the same as in clause 7.3.2 except that the LP order is 6, instead of 4.

After the LP analysis, the past signal $\hat{s}_{LB}(n)$ is filtered through $A(z)$ to obtain the LP residual signal $r_{LB}(n)$:

$$r_{LB}(n) = \hat{s}_{LB}(n) + \sum_{i=1}^{6} a_i\,\hat{s}_{LB}(n-i) \qquad n = -289,...,-1 \tag{8-21}$$

8.4.1.3 LTP analysis

The FERC algorithm uses pitch period repetition in LP residual domain. Therefore, the pitch lag, T_{LB} is determined on the past valid pre-processed signal just before the erasure, $\hat{s}_{LB}^{pre}(n)$, $n=-288,...,-1$. T_{LB} is estimated in open loop by an LTP analysis.

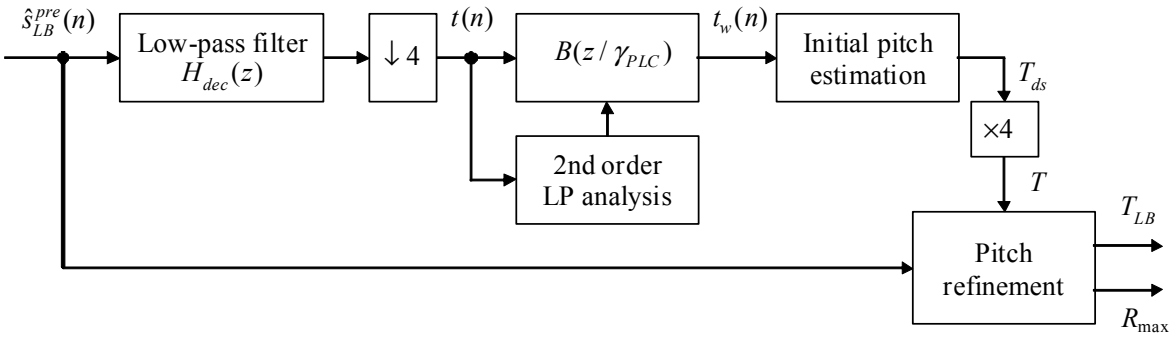


Figure 8-4 – Block diagram of LTP analysis

This pitch estimation, which is illustrated in Figure 8-4, is conducted in the following steps:

- The signal $\hat{s}_{LB}^{pre}(n)$, $n=-288,...,-1$, is low-pass filtered by $H_{dec}(z)$, where:

$$H_{dec}(z)=0.0563(1+z^{-8})+0.0945(z^{-1}+z^{-7})+0.1301(z^{-2}+z^{-6})+0.1554(z^{-3}+z^{-5})+0.1646z^{-4} \quad (8\text{-}22)$$

  is an 8th-order FIR filter, and decimated by a factor 4 to obtain a signal $t(n)$, $n=-72,...,-1$, sampled at 2 kHz. The filter memory is initialized to 0 each time.

- The signal $t(n)$, $n=-72,...,-1$, is weighted by a filter $B(z/\gamma_{PLC})$ where $B(z)=1-b_1z^{-1}-b_2z^{-2}$ and $\gamma_{PLC}=0.94$ to obtain the signal $t_w(n)$, $n=-72,...,-1$. 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 paragraph. Note that only the last 72 samples of the window $w_{LP2}(n)$, $n=-72,...,-1$, are used, which gives a 36-ms time-support at 2 kHz sampling frequency.

- A preliminary estimation of the pitch delay T_{ds} is computed in the weighted decimated signal domain by normalized cross-correlation as follows:

  a) Initialize: $T_{ds}=18$.

  b) Compute the normalized cross-correlation:

$$R'_{\text{norm}}(i)=\frac{\sum_{n=-35}^{-1}t_w(n)t_w(n-i)}{\max\left(\sum_{n=-35}^{-1}t_w^2(n),\sum_{n=-35}^{-1}t_w^2(n-i)\right)} \qquad i=1,...,35 \quad (8\text{-}23)$$

  c) Find the 1st zero crossing position n_0 of $t_w(n)$ within the range of $n=-1,...,-35$: set a temporary index $i_0=1-n_0$.

  d) Determine the first delay i_1 in $[1,35]$ such as $R'_{\text{norm}}(i)<0$. Note that if $\min_{i=1,...,35}(R_{\text{norm}}(i))\geq 0$, the two other steps are omitted.

e) Determine the lower bound i_2 for the maximum correlation search:

$$i_2 = \max(i_0, i_1, 4)$$

f) Search the maximum correlation in the range $[i_2, 35]$

$$T_{ds} = \underset{i=i_2,\cdots,35}{\arg\max} R'_{\text{norm}}(i) . \tag{8-24}$$

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 using the pre-processed signal by searching the cross-correlation maximum in the neighbourhood of $T = 4T_{ds}$ to obtain T_{LB} as follows:

$$T_{LB} = \underset{i=T-2,\cdots,T+2}{\arg\max} R_{\text{norm}}(i) \tag{8-25}$$

with

$$R_{norm}(i) = \frac{\sum_{j=-T}^{-1} \hat{s}_{LB}^{pre}(n)\hat{s}_{LB}^{pre}(n-i)}{\max\left(\sum_{n=-T}^{-1} \hat{s}_{LB}^{pre}(n)^2, \sum_{n=-T}^{-1} \hat{s}_{LB}^{pre}(n-i)^2\right)} \tag{8-26}$$

The maximum of the cross-correlation $R_{\max}$ is computed as:

$$R_{\max} = R_{\text{norm}}(T_{LB}) \tag{8-27}$$

and this will be later used in the signal classification in the next clause.

Note that, as explained in clause 8.4.1.5, the pitch delay T_{LB} may be further modified according to the signal classification.

8.4.1.4 Signal classification

In order to optimize quality, the FERC strategy depends on the signal characteristics. 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. So, a classification is performed on the signal preceding an erasure, $\hat{s}_{LB}^{pre}(n)$, $n = -288,...,-1$. The classification result is used in LP residual extrapolation (clause 8.4.1.5) and muting control (clause 8.4.1.7). There are four possible classes:

- *TRANSIENT* for transients with large energy variation (e.g., plosives).
- *UNVOICED* for 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.
- Zero-crossing rate c_{zc2} of $\hat{s}_{LB}^{pre}(n)$, $n = -80,...,-1$, defined as:

$$c_{zc2} = \sum_{n=-80}^{-1} \left[(\hat{s}_{LB}^{pre}(n) \le 0) \otimes (\hat{s}_{LB}^{pre}(n-1) > 0) \right] \tag{8-28}$$

where the comparisons ≤ and > give a binary result (1 for true, 0 for false). Note that c_{zc2} counts only zero crossings from positive value to negative value.

– Number c_{peak} of large peaks detected in last pitch period:

$$c_{peak} = \sum_{n=-T_{LB}}^{-1} \left[\frac{|r_{LB}(n)|}{8} > \max_{i=-2,\ldots,2} \left(|r_{LB}(n - T_{LB} + i)| \right) \right] \tag{8-29}$$

where the comparison > gives a binary result (1 for true, 0 for false) and T_{LB} is the pitch delay estimated by the LTP analysis.

Based on these features, the signal category *class* is obtained as follows. *class* is set to *UNVOICED* if $c_{zc2} \geq 20$; otherwise, $R_{\max}$ is compared with a threshold of 0.7. If $R_{\max} > 0.7$, *class* is *VOICED*, otherwise the decision depends whether c_{peak} is greater than 0. If $c_{peak} > 0$, then *class* is set to *TRANSIENT*; otherwise *class* is set to *WEAKLY_VOICED*. The classification decision is illustrated in the flowchart given in Figure 8-5.

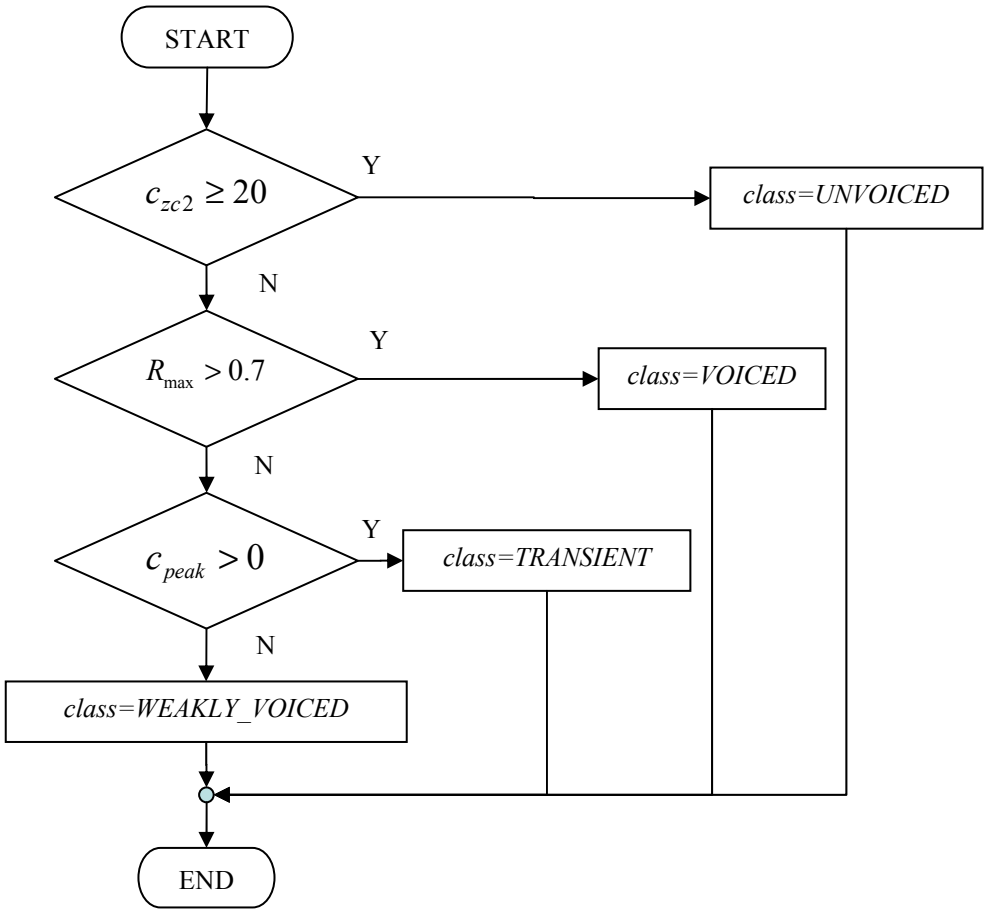


Figure 8-5 – Class decision in lower-band FERC

8.4.1.5 Pitch repetition of LP residual

A pitch repetition procedure is used to estimate the LP residual $r_{LB}(n)$, $n = 0,\ldots,39$, in the erased frame from the residual signal $r_{LB}(n)$ ($n < 0$) of the repetition period. The so-called pitch delay T_{LB} determines the repetition period used in the residual signal generation procedure. The repetition period contains the last valid T_{LB} residual samples $r_{LB}(n)$, $n = -T_{LB},\ldots,-1$.

The pitch repetition period may be modified depending on the signal classification result, *class*: both its length (T_{LB}) and the amplitudes of its samples ($r_{LB}(n)$, $n=-T_{LB},...,-1$) may be changed.

8.4.1.5.1 Pitch delay modification

The possible modifications of T_{LB} are the following:

a) If *class* =*UNVOICED* and if $T_{LB} < 32$ then $T_{LB} = 2T_{LB}$

This doubling of the pitch repetition period is performed to avoid artefacts due to low-pitch delay values.

b) If *class* =*TRANSIENT* and if $T_{LB} > 40$ then $T_{LB} = 40$

Furthermore, if *class* ≠ *VOICED* (*class* =*UNVOICED*, *TRANSIENT* or *WEAKLY_VOICED*) and if T_{LB} is even, then $T_{LB} = T_{LB} + 1$

c) If *class* = *VOICED*

It is verified whether there are two glottal pulses in the repetition period (one at the beginning, the second at the end). This can happen in cases of decreasing pitch, which is detected as follows:

First the position $n_{\max r}$, amplitude $A_{\max r}$, and its sign of the maximum amplitude sample in the repetition period is searched and stored:

$$\begin{aligned} n_{\max r} &= \underset{n=-T_{LB},\cdots,-1}{\arg\max} \left|r_{LB}(n)\right| \\ A_{\max r} &= \left|r_{LB}(n_{\max r})\right| \end{aligned} \tag{8-30}$$

The mean amplitude of the repetition period is also calculated:

$$A_{meanr} = \frac{1}{T_{LB}} \sum_{n=-T_{LB}}^{-1} \left|r_{LB}(n)\right| \tag{8-31}$$

If $A_{\max r} > 4A_{meanr}$ and the position $n_{\max r}$ is close to one of the repetition period bounds ($-T_{LB} \le n_{\max r} \le -T_{LB}+4$ or $-5 \le n_{\max r} \le -1$), then a second maximum amplitude position $n_{\max r2}$ is searched at the other bound of the repetition period:

$$\begin{aligned} n_{\max r2} &= \underset{n=n_{\max r}+T_{LB}-5,...,-1}{\arg\max} \left|r_{LB}(n)\right| && \text{if } -T_{LB} \le n_{\max r} \le -T_{LB}+4 \\ n_{\max r2} &= \underset{n=-T_{LB},...,n_{\max r}-T_{LB}+5}{\arg\max} \left|r_{LB}(n)\right| && \text{if } -5 \le n_{\max r} \le -1 \end{aligned} \tag{8-32}$$

This pulse is considered as a second glottal pulse in the repetition period if its sign is identical to that of the main pulse and $A_{\max r2} > \frac{1}{2} A_{\max r}$ where $A_{\max r2} = \left|\mathrm{r_{LB}(n_{maxr2})}\right|$.

When two glottal pulses are detected in the repetition period, the pitch delay T_{LB} is overwritten by the distance of the two found glottal pulses:

$$T_{LB} = \left|n_{\max r} - n_{\max r2}\right| \tag{8-33}$$

The pitch repetition period may be modified depending on the signal classification result, *class*: both its length (T_{LB}) and the amplitudes of its samples ($r_{LB}(n)$, $n=-T_{LB},...,-1$) may be changed.

8.4.1.5.2 Modifications of sample amplitudes in the repetition period

The possible modifications of the amplitudes of repetition period samples $r_{LB}(n)$, $n = -T_{LB},...,-1$, are the following:

If *class* is *WEAKLY_VOICED*, samples of the repetition period are modified by limiting their amplitude as follows:

$$r_{LB}(n) = \mathrm{sgn}(r_{LB}(n)) \cdot \min\left(\max_{i=-2,...,+2} \left(|r_{LB}(n - T_{LB} + i)| \right), |r_{LB}(n)| \right) \qquad n = -T_{LB},...,-1 \qquad (8\text{-}34)$$

This modification of $r_{LB}(n)$ is meant to detect and avoid repeating large magnitude transients such as plosives.

If the *class* is *UNVOICED*, samples of the repetition period with amplitude higher than Δ_e are divided by 4, where Δ_e is 2.5 times the mean amplitude of the last 10 ms (80 samples) of $r_{LB}(n)$:

$$\Delta_e = 2.5\left(\frac{1}{80} \sum_{-1}^{-80} |r_{LB}(n)| \right) \qquad (8\text{-}35)$$

$$r_{LB}(n) = r_{LB}(n)/4, \quad \text{if} \quad r_{LB}(n) > \Delta_e \qquad (8\text{-}36)$$

8.4.1.5.3 Pitch repetition procedure

The LP residual $r_{LB}(n)$, $n = 0,...,39$, in the erased frame is extrapolated from the repetition period. This pitch repetition procedure also depends on the signal classification result, *class*:

– If *class* is *VOICED*, the missing signal, $r_{LB}(n)$, is then obtained by repeating pitch-synchronously the repetition period:

$$r_{LB}(n) = r_{LB}(n - T_{LB}) \qquad n = 0,...,39 \qquad (8\text{-}37)$$

– 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 extrapolated signal, $r_{LB}(n)$, is obtained as:

$$r_{LB}(n) = r_{LB}(n - T_{LB} + (-1)^n) \qquad n = 0,...,39 \qquad (8\text{-}38)$$

Note that if $T_{LB} < 40$, the extrapolated residual signal extends the repetition period and the procedure is iterated.

8.4.1.6 LP synthesis

The reconstructed erased frame is extrapolated as follows:

$$\hat{s}_{LB}^{extr}(n) = r_{LB}(n) - \sum_{i=1}^{6} a_i \hat{s}_{LB}^{extr}(n - i) \qquad n = 0,...,39 \qquad (8\text{-}39)$$

where $r_{LB}(n)$ is the extrapolated residual signal.

8.4.1.7 Adaptive muting

The energy of the extrapolated signal $\hat{s}_{LB}^{extr}(n)$ is muted using an adaptively changing gain factor g_{mute} as:

$$\hat{s}_{LB}^{extrg}(n) = g_{mute}^{(n)} \cdot \hat{s}_{LB}^{extr}(n) \qquad (8\text{-}40)$$

In the following, the calculation of g_{mute} is based on four parameters: $\Delta\tau$, δ_1, δ_{2p} and δ_{3p}, which depend on the value of *class* as shown in Table 8-1.

Table 8-1 – Adaptive muting parameters with respect to *class*

Parameter	TRANSIENT	VOICED or WEAKLY_VOICED	UNVOICED
$\Delta\tau$	4	1	1
δ_1	$8.3313 \cdot 10^{-3}$	$3.0518 \cdot 10^{-4}$	$3.0518 \cdot 10^{-4}$
δ_{2p}	0	$3.0518 \cdot 10^{-4}$	$3.0518 \cdot 10^{-4}$
δ_{3p}	0	Adaptively calculated (see clause 8.4.1.7.2)	$2.2888 \cdot 10^{-3}$

The parameter $\Delta\tau$ represents the continuous time index of the muting factor in erased frames, and is initialized to 0 in a good frame. In case of erased frames, this parameter is updated sample by sample as:

$$\tau(n) = \tau(n-1) + \Delta\tau . \tag{8-41}$$

8.4.1.7.1 Muting factor in the first and second erased frames

In case of the first erased frame, i.e., the current *m*-th frame is erased and the previous (*m*–1)-th frame is not erased, the muting factor g_{mute} is computed as follows. Since the waveform and energy of continuous segments at intervals of one pitch often change rapidly at the offset of vowels, the energy ratio of samples from last two pitch periods is computed to dynamically update the muting factor instead of using a fixed value. This energy ratio parameter $\rho^{(m)}$ is computed in the first erased frame (frame number *m*) as:

$$\rho^{(m)} = \begin{cases} \dfrac{1 - \sqrt{E_{p1} / E_{p2}}}{T_{LB}} & \text{if } E_{p1} < E_{p2} \text{ and } E_{p2} > 360 \cdot 2^{11} T_{LB} \\ 0 & \text{otherwise} \end{cases} \tag{8-42}$$

where E_{p1} and E_{p2} are the energy in the last and second last pitch periods, respectively:

$$E_{p1} = \sum_{n=-1}^{-T_{LB}} (\hat{s}_{LB}(n))^2 \tag{8-43}$$

$$E_{p2} = \sum_{n=-(T_{LB}+1)}^{-2T_{LB}} (\hat{s}_{LB}(n))^2 \tag{8-44}$$

Then the muting factor is recursively calculated on a sample-by-sample basis as:

$$g_{mute}^{(n)} = \begin{cases} g_{mute}^{(n-1)} - \delta_1 & \text{if } \delta_1 > \rho^{(m)} \\ g_{mute}^{(n-1)} - \rho^{(m)} & \text{otherwise} \end{cases} \qquad n = 0, \cdots, 39 \tag{8-45}$$

where $g_{mute}^{(-1)}$ is set to 1. In case of the second erased frame, $\rho^{(m)}$ is updated as:

$$\rho^{(m)} = 0.5\rho^{(m-1)} \tag{8-46}$$

The muting factor is initialized as $g_{mute}^{(0),(m)} = \begin{cases} g_{mute}^{(39),(m-1)} - \delta_1 & \text{if } \delta_1 > \rho^{(m)} \\ g_{mute}^{(39),(m-1)} - \rho^{(m)} & \text{otherwise} \end{cases}$

where $g_{mute}^{(0),(m)}$ and $g_{mute}^{(39),(m-1)}$ denote the initial value in the current frame and last value in the previous frame, respectively. In the second frame, this factor is updated similarly using equation 8-45.

8.4.1.7.2 Muting factor in consecutively erased frames after the second erased frame

After two consecutive erased frames, the muting factor g_{mute} is updated sample-by-sample, depending on the value of $\tau(n)$.

Table 8-2 – Adaptive muting factor calculation

Time index $\tau(n)$	**Muting factor** $g_{mute}^{(n)}$
$80 \le \tau(n) < 160$	$g_{mute}^{(n)} = g_{mute}^{(n-1)} - (\delta_1 + \delta_{2p})$
$160 \le \tau(n) < 480$	$g_{mute}^{(n)} = g_{mute}^{(n-1)} - (\delta_1 + \delta_{2p} + \delta_{3p})$
$480 \le \tau(n)$	$g_{mute}^{(n)} = 0$

It should be noted that, for *TRANSIENT*, the muting factor time index $\tau(n)$ is already 320 at the beginning of the third erased frame, and the first condition, i.e., $80 \le \tau(n) < 160$, is not applicable.

If *class* is *VOICED* or *WEAKLY_VOICED*, δ_{3p} is calculated adaptively as:

$$\delta_{3p} = \begin{cases} \max\left(\frac{g_{mute}^{(n)}}{320} - (\delta_1 + \delta_{2p}), 0\right) & \text{if } g_{mute}^{(n)} < 0.9277 \\ 2.2888 \cdot 10^{-3} & \text{otherwise} \end{cases} \quad (8\text{-}47)$$

The muting factor with respect to sample number is illustrated in Figure 8-6.

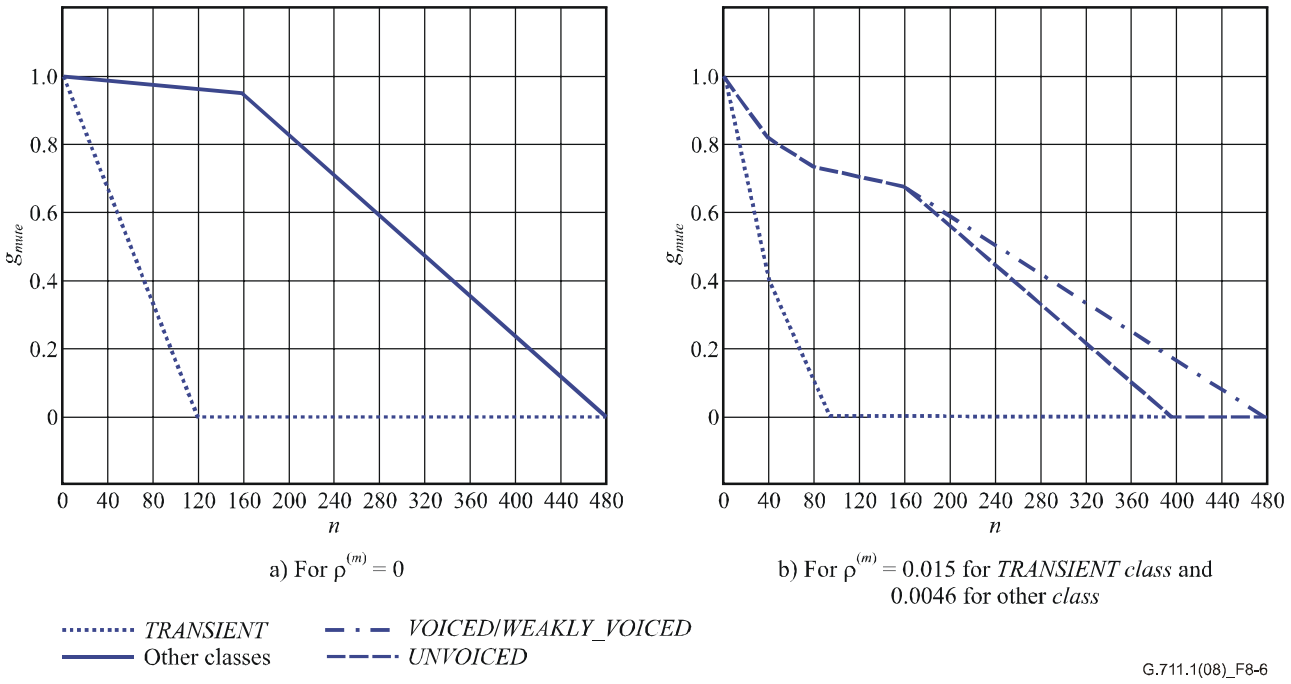


Figure 8-6 – Muting factor g_{mute} as a function of the sample number n

8.4.2 Extrapolation of erased frames: Case of erased frame after another erased frame

If both the current and previous frames are erased ones, the analysis parameters ($\{a_j\}_{j=0,\cdots,6}$, T_{LB}, *class*) computed for the first erased frame are reused. Still, the LP filter $A(z)$ is slightly attenuated. For the m-th consecutive erased frame the new filter is obtained by:

$$a_j^{(m)} = 0.99^j \, a_j^{(m-1)} \qquad j = 1,\cdots,6 \qquad (8\text{-}48)$$

8.4.3 Signal resynchronization algorithm and cross-fading

During concealment of erased voiced frames, the past signal is repeated using an estimated pitch lag. When the first good frame after a series of erasures is received, pitch discontinuity may appear and produce an annoying artefact. To prevent this problem, signal resynchronization is performed for voiced signals in the first good frame after a series of erasures. The resynchronization is applied in the last concealment frame and the first good decoded frame to smooth out signal transitions and avoid the origin of artefacts. The principle of the signal resynchronization is shown in Figure 8-7.

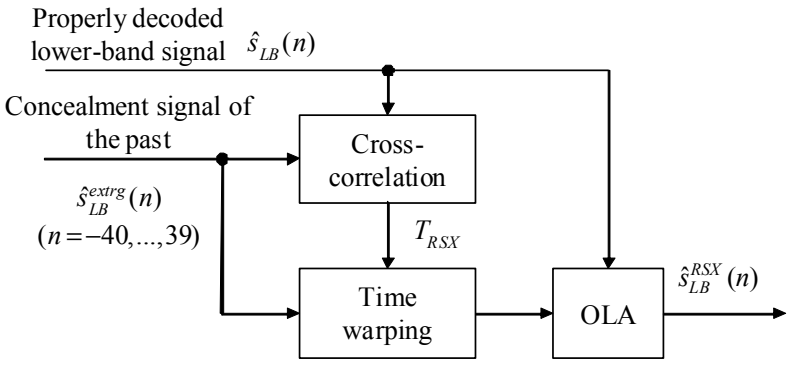


Figure 8-7 – Details of the signal resynchronization algorithm

The steps are as follows. If a first good frame is received and $\hat{s}_{LB}(n)$ is properly decoded, one extra frame is generated by the FERC algorithm, i.e., yet another extrapolation of the concealment signal in the previously lost frame. A cross-correlation analysis is performed between the two signals in the current frame: the decoded signal of the good frame $\hat{s}_{LB}(n)$, $n = 0,...,39$ and the concealment signal extended to the present (good) frame $\hat{s}_{LB}^{extrg}(n)$, $n = -40,...39$. A pitch delay T_{RSX} is extracted based on the cross-correlation analysis. Based on this delay, the concealment signal (corresponding to the concatenation of the previous and the current frame) is synchronized with the decoded signal by time-warping (compressing or expanding) the extended concealment signal. This is to align the phase of the concealment signal with that of the good decoded signal. After time-warping the concealment signal, the part corresponding to the previous frame is extracted and output, and the part corresponding to the current frame is cross-faded with the decoded good frame in order to produce a smooth transition between the extrapolated concealment signal and the properly decoded signal.

In the first decoded frame after a series of erased frames, the concealment algorithm generates an extra frame of concealment signal (in the same way as if the decoded frame were erased). A cross-correlation analysis is then performed between the concealment signal $\hat{s}_{LB}^{extrg}(n)$ and the decoded signal $\hat{s}_{LB}(n)$ in the range [–5, 5]. The correlation function is defined as:

$$R_{RSX}(i) = \sum_{n=0}^{34} \hat{s}_{LB}^{extrg}(n+i)\hat{s}_{LB}(n) \qquad i = -5,\cdots,5 \tag{8-49}$$

The maximum R_{RSX} of the correlation function and the delay T_{RSX} corresponding to this maximum are respectively retrieved by:

$$\begin{aligned} R_{RSX\max} &= \max_{i=-5,...,5}\left(R_{RSX}(i)\right) \\ T_{RSX} &= \arg\max_{i=-5,...,5}\left(R_{RSX}(i)\right) \end{aligned} \tag{8-50}$$

To normalize the maximum correlation, the following two energies are calculated:

$$\begin{aligned} E_0^{RSX} &= \sum_{n=0}^{39}\left(\hat{s}_{LB}(n)\right)^2 \\ E_1^{RSX} &= \sum_{n=0}^{39}\left(\hat{s}_{LB}^{extrg}(T_{RSX}+n)\right)^2 \end{aligned} \tag{8-51}$$

and $R_{RSX\max}$ is divided by the square root of their product:

$$C_{RSX} = \frac{R_{RSX\max}}{\sqrt{E_0^{RSX}E_1^{RSX}}} \tag{8-52}$$

The resynchronization is not applied when there is a big discrepancy between the energies of the extrapolated frame and the good frame. Therefore, an energy ratio r_{RSX} is calculated according to

$$r_{RSX} = \frac{\max\left(E_0^{RSX}, E_1^{RSX}\right)}{\min\left(E_0^{RSX}, E_1^{RSX}\right)}. \tag{8-53}$$

The condition to proceed with the resynchronization is defined as

[($class == VOICED$) AND ($C_{RSX} > 0.7$) AND ($r_{RSX} > 0.5$)].

If this condition is not satisfied, no resynchronization is performed (i.e., $\hat{s}_{LB}^{RSX}(n) = \hat{s}_{LB}^{extrg}(n)$, $n = -40,...,39$). Otherwise, if this condition is satisfied, the concealment signal is extended or shortened (compressed) depending on the delay parameter found earlier. Note that this is done for the whole concealment signal $\hat{s}_{LB}^{extrg}(n)$, $n = -40,...,39$.

The signal compression or expansion is performed by means of "resampling" using linear interpolation. If the distance between adjacent samples of the original signal is considered as "1", the distance between adjacent samples of the resampled signal is defined as:

$$\Delta_{RSX} = \frac{79 - T_{RSX}}{79} \tag{8-54}$$

Since d^{RSX} is allowed to vary only in the range [–5, 5] Δ_{RSX} may vary only in the range [0.9367, 1.0633]. The values of the re-sampled signal are calculated from the values of the original signal at positions given by multiples of Δ_{RSX}, i.e.:

$$p(n) = (n + 40) \cdot \Delta_{RSX} \qquad n = -40,...,39 \tag{8-55}$$

As mentioned previously, the resampling is carried out on the whole concealment signal $\hat{s}_{LB}^{extrg}(n)$, $n = -40,...,39$. The re-sampled concealment signal $\hat{s}_{LB}^{RSX}(n)$ is given by:

$$\hat{s}_{LB}^{RSX}(n) = \left(1 - \alpha^{RSX}(n)\right)\hat{s}_{LB}^{extrg}(\lfloor p(n) \rfloor) + \alpha^{RSX}(n)\hat{s}_{LB}^{extrg}(\lfloor p(n) \rfloor + 1) \qquad n = -40,..., \min(79, 79 + d^{RSX}) \tag{8-56}$$

with $\alpha_{RSX}(n) = p(n) - \lfloor p(n) \rfloor$.

If $T_{RSX} > 0$, the missing samples $\hat{s}_{LB}^{RSX}(n)$, $n = 39 - T_{RSX},...,39$, are set to zero. This will not cause a problem since the cross-fading operation (OLA) that follows the resynchronization uses a triangular window and usually the last samples of the resynchronized signal are multiplied by a factor close to zero. The principle of resynchronization is illustrated in Figure 8-8, where an extension by two samples is exercised.

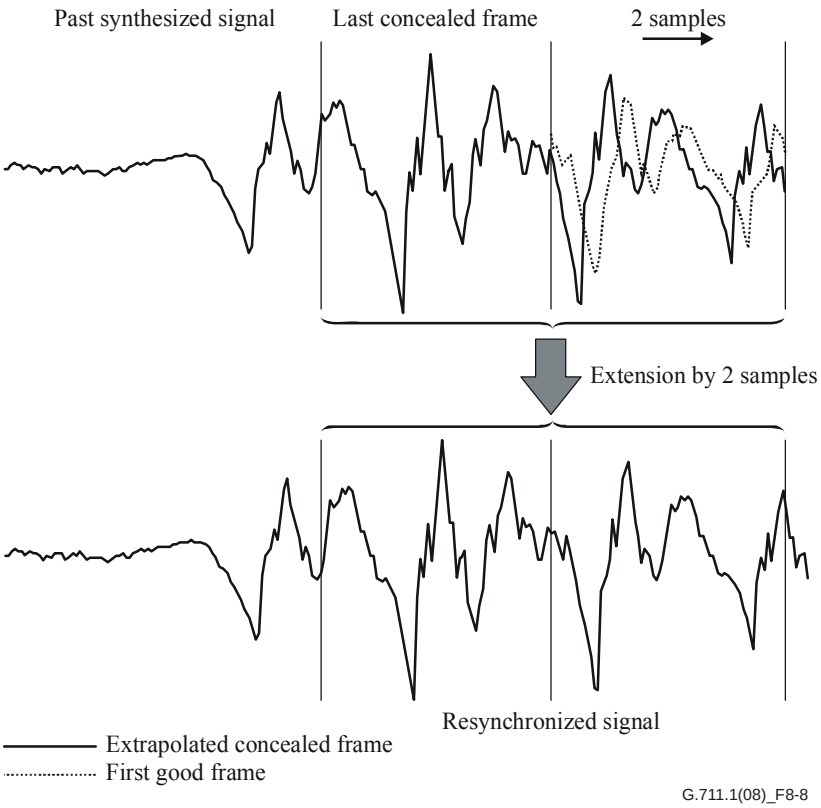


Figure 8-8 – Principle of resynchronization

After the signal is resynchronized, its energy is scaled according to the dynamic energy-scaling algorithm and a cross-fading (OLA) operation is performed between the resynchronized scaled signal and the decoded signal. This is explained in clauses 8.4.4 and 8.4.5.

8.4.4 Dynamic energy scaling algorithm

It often occurs that the energy of the resynchronized concealment signal $\hat{s}_{LB}^{RSX}(n)$, n = 0,…,39 does not match the energy of the newly decoded signal $\hat{s}_{LB}(n)$. This often makes an artefact for continuous stationary voice frames. To make the reconstructed signal smoother in energy with previous frames and the newly decoded frame, dynamic energy scaling is performed after resynchronization.

The energy ratio between the newly decoded signal and the reconstructed signal is used to perform dynamic energy scaling. To avoid waveform distortion and noise due to unexpected amplitude variance, dynamic energy scaling is just enabled in case of $E_1^{RSX} > E_0^{RSX}$.

In this case, the dynamic energy scaling factor is calculated as follows:

$$\alpha_{es} = \frac{1}{80}\sqrt{1 - 1/r_{RSX}} \tag{8-57}$$

where E_0^{RSX} and E_1^{RSX} are defined in equation 8-51 and r_{RSX} is defined in equation 8-53. Then, the reconstructed signal is scaled as follows:

$$\hat{s}_{LB}^{SYN}(n) = (1 - (n+40)\alpha_{es})\hat{s}_{LB}^{RSX}(n) \qquad n = -40,...,39 \tag{8-58}$$

The principle and effects of the dynamic energy scaling algorithm are illustrated in Figure 8-9. The energy of the resynchronized signal in the first good frame does not match with the energy of the

original signal. After dynamic energy scaling and cross-fading (cross-fading is the following operation, explained in clause 8.4.5), the reconstructed signal is closer to the original signal.

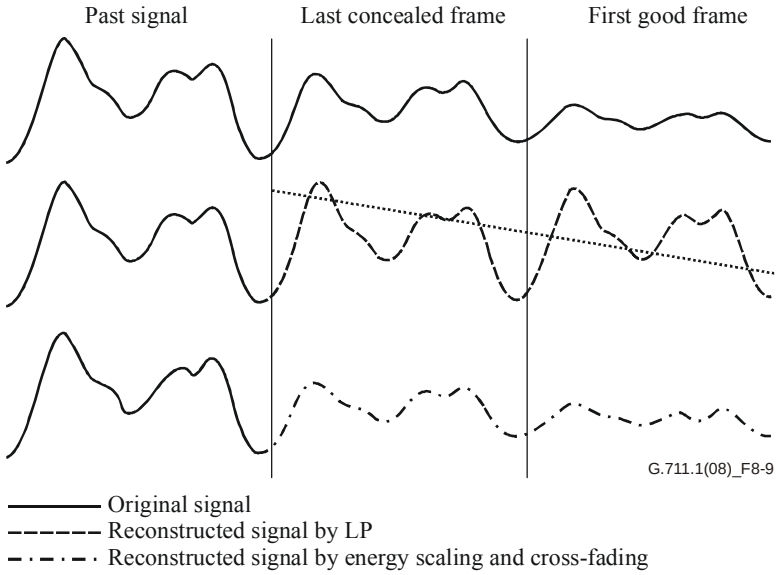


Figure 8-9 – Principle of dynamic energy scaling

After finding the resynchronized, energy-scaled concealment signal, the part corresponding to the past frame is extracted, i.e., $\hat{s}_{LB}^{SYN}(n)$, $n=-40,...,-1$, and the part corresponding to the current frame, $\hat{s}_{LB}^{SYN}(n)$, $n=0,...,39$, is cross-faded with the decoded signal $\hat{s}_{LB}(n)$, $n=0,...,39$, as is explained in the next clause.

8.4.5 Cross-fading between the concealment signal and the decoded signal

The cross-fading operation is applied in one frame length, i.e., between 40 samples of the resynchronized, energy-scaled concealment signal $\hat{s}_{LB}^{SYN}(n)$ and the newly decoded signal $\hat{s}_{LB}(n)$. The cross-faded signal replaces the decoded signal $\hat{s}_{LB}(n)$ in the following way:

$$\hat{s}_{LB}(n) = \left(1 - w_{cf}(n)\right)\hat{s}_{LB}^{SYN}(n) + w_{cf}(n)\hat{s}_{LB}(n) \qquad n=0,...,39 \qquad (8\text{-}59)$$

Here, $w_{cf}(n)$ is a triangular window which is defined by:

$$w_{cf}(n) = \frac{(n+1)}{40} \qquad n=0,...,39 \qquad (8\text{-}60)$$

8.5 Higher-band FERC

In the case of an erased frame, the higher-band FERC algorithm is used to recover the intermediate signal. The higher-band FERC differentiates between pitch-like higher-band signal and noise-like higher-band signal in order to minimize potential quality impairments in the recovered signal generated for concealment.

When previously decoded higher-band signal exhibits a high correlation, the higher-band pitch lag is estimated around the pitch parameter calculated for the lower-band FERC. The samples of the previous pitch period in the higher-band FERC history buffer are employed as the basic inverse MDCT data of the current erased frame. A sinusoid window is applied to the basic inverse MDCT data and OLA is performed sequentially.

Otherwise, an attenuated inverse MDCT signal from the last good frame is copied and used as the inverse MDCT data for erased frames. OLA is performed to generate the recovered higher-band signal.

Figure 8-10 shows the diagram of the higher-band FERC.

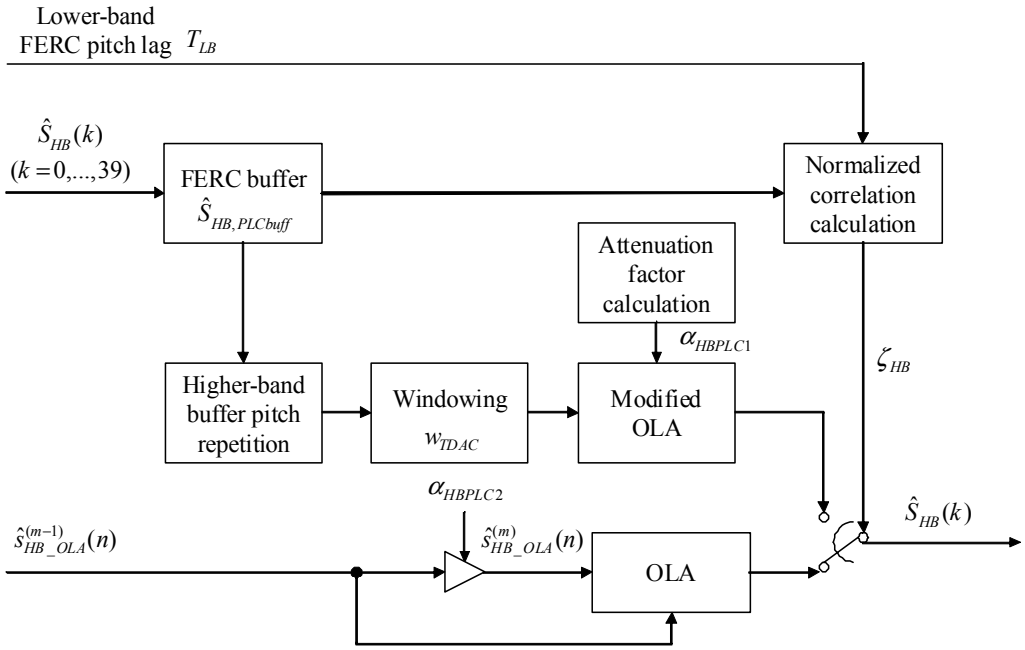


Figure 8-10 – Block diagram of higher-band FERC

8.5.1 Update of higher-band signal buffer in case of a good frame

The decoded higher-band signal $\hat{s}_{HB}(n)$ is always stored for the higher-band reconstruction in case of erasure in future frames. The higher-band FERC buffer $\hat{s}_{HB,PLCbuff}(n)$ is initialized to zero. The buffer length is $T_{HBmax}+120$, where $T_{HBmax}=144$ is the maximum pitch lag. This buffer has two parts. The first part is used to store previous decoded higher-band signal, the latter is used to generate pitch repeated higher-band signal. The lengths of these two parts are $T_{HBmax}+40$ and 80, respectively.

In case of a good frame, the higher-band FERC buffer $\hat{s}_{HB,PLCbuff}$ is updated as follows:

$$\hat{s}_{HB,PLCbuff}(n)=\hat{s}_{HB,PLCbuff}(n+40) \qquad n=0,\ldots,T_{HBmax}-1 \qquad \text{(8-61)}$$

$$\hat{s}_{HB,PLCbuff}(n+T_{HBmax})=\hat{s}_{HB}(n) \qquad n=0,\ldots,39 \qquad \text{(8-62)}$$

8.5.2 Computation of normalized correlation and estimation of higher-band pitch lag in case of an erased frame after a good frame

The higher-band pitch lag T_{HB} is estimated from a normalized correlation by the following steps:

1) Computation of the normalized correlations around the lower-band pitch lag:

$$R_{HB}(i) = \frac{\sum_{n=0}^{39} \hat{s}_{HB,PLCbuff}(T_{HBmax}+n) \cdot \hat{s}_{HB,PLCbuff}(T_{HBmax}+n-i)}{\sqrt{\sum_{n=0}^{39} \hat{s}_{HB,PLCbuff}{}^{2}(T_{HBmax}+n) \cdot \sum_{n=0}^{39} \hat{s}_{HB,PLCbuff}{}^{2}(T_{HBmax}+n-i)}} \qquad (8\text{-}63)$$

$$\max(T_{LB}-3, T_{HBmin}) \le i \le \min(T_{LB}+3, T_{HBmax})$$

where T_{LB} is the lower-band pitch lag obtained from lower-band FERC module, $T_{HBmin}=16$ is the minimal pitch lag, and $R_{HB}(i)$ is the normalized correlation function. Initially, T_{HB} is set to T_{LB}.

2) Search of the maximum correlation:

$$R_{HB\_MAX} = \max\left(R_{HB}(i)\right) \qquad i = \max(T_{LB}-3, T_{HB\min}), \ldots, \min(T_{LB}+3, T_{HB\max}) \qquad (8\text{-}64)$$

3) Estimation of higher-band pitch lag:

$$T_{HB} = \arg\max_{i}\left(R_{HB}(i)\right) \qquad i = \max(T_{LB}-3, T_{HB\min}), \ldots, \min(T_{LB}+3, T_{HB\max}) \qquad (8\text{-}65)$$

4) If $\left|R_{HB\_MAX}\right| \ge 0.7$, set high correlation flag $\zeta_{HB}=1$; otherwise, $\zeta_{HB}=0$.

It should be noted that the above processing is only performed in obtaining the high correlation flag ζ_{HB} in the first erased frame. For consecutive erased frames, the high correlation flag value is repeatedly used.

8.5.3 High correlation case

When the signal is highly correlated, ($\zeta_{HB}=1$), the higher-band signal is recovered by a pitch repetition of the previous frame, attenuation, and an OLA.

If the previous frame was also an erased frame, the first part of the higher-band FERC buffer is reproduced by shifting the buffer contents as follows:

$$\hat{s}_{HB,PLCbuff}(n) = \hat{s}_{HB,PLCbuff}(n+40) \qquad n = 0, \ldots, T_{HBmax}+39 \qquad (8\text{-}66)$$

Then, the second part of $\hat{s}_{HB,PLCbuff}(n)$ is generated by pitch repetition of the first part:

$$\hat{s}_{HB,PLCbuff}(n+T_{HBmax}+40) = \hat{s}_{HB,PLCbuff}(n-T_{HB}+T_{HBmax}+40) \qquad n = 0, \ldots, 79 \qquad (8\text{-}67)$$

The intermediate inverse-transformed higher-band signal of the erased (and current) frame $\hat{s}^{(m)}_{HB\_OLA}(n)$ is estimated by applying the OLA weighting window $w_{TDAC}(n)$:

$$\hat{s}^{(m)}_{HB\_OLA}(n) = \hat{s}_{HB,PLCbuff}(n+T_{HB\max}+40) w_{TDAC}(n) \qquad n = 0, \ldots, 79 \qquad (8\text{-}68)$$

Then the higher-band signal $\hat{s}_{HB}(n)$ is calculated with a modified OLA window with attenuation:

$$\hat{s}_{HB}(n)=\begin{cases}\alpha_{HBPLC1}^{(n)}\left(w_{TDAC}(n+40)\hat{s}_{HB\_OLA}^{(m-1)}(n)+w_{TDAC}(n)\hat{s}_{HB\_OLA}^{(m)}(n)\right) & \text{if } \alpha_{HBPLC1}^{(0)}\geq 0.2\\ 0 & \text{otherwise}\end{cases} \tag{8-69}$$

$$n=0,\ldots,39$$

where $\alpha_{HBPLC1}^{(n)}$ is a sample-by-sample attenuation weighting factor. This attenuation weighting factor $\alpha_{HBPLC1}^{(n)}$ is recursively updated as:

$$\begin{cases}\alpha_{HBPLC1}^{(n)}=\alpha_{HBPLC1}^{(n-1)}-0.005 & \text{if } \alpha_{HBPLC1}^{(0)}\geq 0.2\\ \alpha_{HBPLC1}^{(n)}=\alpha_{HBPLC1}^{(0)} & \text{otherwise}\end{cases} \qquad n=0,\ldots,39 \tag{8-70}$$

where $\alpha_{HBPLC1}^{(0)}$ is initialized with $\alpha_{HBPLC1}^{(39)}-0.005$ at the end of the previous frame. If the frame is the first erased frame after a good frame, $\alpha_{HBPLC1}^{(0)}$ is reset to 1 at the end of the frame.

Then, $\hat{s}_{HB\_OLA}^{(m-1)}$ is updated as:

$$\hat{s}_{HB\_OLA}^{(m-1)}(n)=\hat{s}_{HB\_OLA}^{(m)}(n+40) \qquad n=0,\ldots,39 \tag{8-71}$$

In case of a good frame after an erased frame, the intermediate inverse-transformed higher-band is calculated with:

$$\hat{s}_{HB\_OLA}^{(m-1)}(n)=\alpha_{HBPLC1}^{(39),(m-1)}\hat{s}_{HB\_OLA}^{(m-1)}(n) \qquad n=0,\ldots,39 \tag{8-72}$$

where $\alpha_{HBPLC1}^{(39),(m-1)}$ is the last coefficient in the previous $(m-1)$ th frame.

8.5.4 Low correlation case

When the signal is not highly correlated ($\zeta_{HB}=0$), the intermediate inverse-transformed higher-band signal $\hat{s}_{HB\_OLA}^{(m)}(n)$ is recovered using an attenuated version of the previous signal $\hat{s}_{HB\_OLA}^{(m-1)}(n)$ by:

$$\hat{s}_{HB\_OLA}^{(m)}(n)=\alpha_{HBPLC2}\hat{s}_{HB\_OLA}^{(m-1)}(n) \qquad n=0,\ldots,79 \tag{8-73}$$

where $\alpha_{HBPLC2}=0.875$ is the attenuation factor. Then, to synthesize output signal $\hat{s}_{HB}(n)$, an OLA is performed using equation 8-15.

8.6 Synthesis QMF

A synthesis QMF is used to synthesize the 16-kHz-sampled output signal from the lower-band decoded signals and higher-band decoded signals. Both the 8-kHz-sampled decoded signals in the lower and higher bands are upsampled by a factor of 2. Then, the upsampled signals are filtered through the synthesis filter for each band. The coefficients of those two synthesis filters are given by:

$$\begin{cases}h_L^{qmfS}(i)=h_L^{qmfA}(i)\\ h_H^{qmfS}(i)=-h_H^{qmfA}(i)\end{cases} \qquad i=0,\cdots,31 \tag{8-74}$$

where h_L^{qmfS} and h_H^{qmfS} are the coefficients of the lower-band and higher-band synthesis filters, and h_L^{qmfA} and h_H^{qmfA} are those of the analysis QMF described in clause 7.2, respectively. The 16-kHz-sampled outputs $\hat{s}_{QMF}$ are obtained by adding the two filtered signals as follows:

$$\hat{s}_{QMF}(n) = \sum_{i=0}^{31} h_L^{qmfS}(i)\hat{s}_{LW}(n-i) + \sum_{i=0}^{31} h_L^{qmfS}(n)\hat{s}_{HW}(n-i) \qquad n = 0,\cdots,79 \qquad (8\text{-}75)$$

where $\hat{s}_{LW}(n)$ and $\hat{s}_{HW}(n)$ are the upsampled signals in the lower band and 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:

$$\begin{aligned} \hat{s}_{sum}(n) &= \sum_{i=0}^{15} h_0^{qmf}(i)\left(\hat{s}_{LB}(n-i) + \hat{s}_{HB}(n-i)\right) \\ \hat{s}_{diff}(n) &= \sum_{i=0}^{15} h_1^{qmf}(n)\left(\hat{s}_{LB}(n-i) - \hat{s}_{HB}(n-i)\right) \end{aligned} \qquad n = 0,\cdots,39, \qquad (8\text{-}76)$$

where $\hat{s}_{LB}(n)$ is the decoded lower band signal, $\hat{s}_{HB}(n)$ is the decoded higher band signal and h_0^{qmf} and h_1^{qmf} are the filter coefficients described in Table 7-1. Then the intermediate signals $\hat{s}_{sum}(n)$ and $\hat{s}_{diff}(n)$ are interleaved to obtain the 16-kHz sampled signal $\hat{s}_{QMF}(n)$ as follows:

$$\begin{aligned} \hat{s}_{QMF}(2n) &= 2\hat{s}_{diff}(n) \\ \hat{s}_{QMF}(2n+1) &= 2\hat{s}_{sum}(n) \end{aligned} \qquad n = 0,\cdots,39 \qquad (8\text{-}77)$$

8.7 Noise gate

To further increase the cleanness of the synthesized signal during quasi-silent periods, a noise gate is added at the decoder. The function of the noise gate is to attenuate the output signal when the frame energy is very low. This attenuation is progressive in both level and time. The level of attenuation is signal-dependant and is gradually modified on a sample-by-sample basis.

First, the lower-band synthesized signal $\hat{s}_{LB}(n)$ is filtered by a first-order high-pass FIR filter:

$$z(n) = \hat{s}_{LB}(n) - 0.75 \cdot \hat{s}_{LB}(n-1) \qquad n = 0,\cdots,39 \qquad (8\text{-}78)$$

and its energy is calculated by:

$$E_0 = \sum_{n=0}^{39} z^2(n) \qquad (8\text{-}79)$$

In order to avoid fast switching of the noise gate, energy E_{-1} of the previous frame is added to the energy of the current frame, which gives the total energy:

$$E_t = E_0 + E_{-1} \qquad (8\text{-}80)$$

Note that E_{-1} is initialized to 0 at the start, and E_{-1} is replaced by E_0 at the end of each frame. Based on the information about signal energy, a target gain is calculated using the square root of the total energy, but limited not to below 0.25, as:

$$g_t = \max\left(\frac{\sqrt{E_t}}{2^7}, 0.25\right). \qquad (8\text{-}81)$$

The noise gate may be progressively deactivated (opened) by setting the target gain to $g_t = 1.0$. This happens when the synthesized signal in the decoder has its energy concentrated in the higher band, i.e., 4000-8000 Hz. To detect this situation, a power measure is calculated for the lower-band and the higher-band synthesized signal in the current frame. Specifically, the power of the lower-band signal $\hat{s}_{LB}(n)$ is given by:

$$P_{LB} = \sum_{n=0}^{39} |\hat{s}_{LB}(n)| \tag{8-82}$$

and the power of the higher-band signal $\hat{s}_{HB}(n)$ is given by:

$$P_{HB} = \sum_{n=0}^{39} |\hat{s}_{HB}(n)| \tag{8-83}$$

Note that if layer 2 is not decoded, i.e., the signal is narrow-band, the noise gate is always active and its attenuation effect depends on the value of the calculated target gain. On the other hand, when layer 2 is decoded and the following conditions:

$$P_{HB} > \frac{800}{2^{31}-1} \quad \text{and} \quad P_{HB} > 16 P_{LB} \tag{8-84}$$

are satisfied, target gain is set to $g_t = 1.0$, which progressively deactivates (opens) the noise gate.

Finally, each sample of the output synthesized signal $\hat{s}_{QMF}(n)$ is multiplied by a gain:

$$g_{NG}^{(n)} = 0.9893 g_{NG}^{(n-1)} + 0.0107 g_t \qquad n \begin{cases} 0,\cdots,39 & \text{for narrow-band output} \\ 0,\cdots,79 & \text{for wideband output} \end{cases} \tag{8-85}$$

which is updated on a sample-by-sample basis. Gain $g_{NG}^{(-1)}$ is initialized to 1.0. It can be seen that this gain converges slowly towards the target gain g_t. Thus, the output signal for a wideband output is updated by:

$$\hat{s}_{WB}(n) = \begin{cases} g_{NG}^{(n)} \hat{s}_{QMF}(n) & \text{if } g_{NG}^{(n)} < 1.0 \\ \hat{s}_{QMF}(n) & \text{otherwise} \end{cases} \qquad n = 0,\cdots,79 \tag{8-86}$$

and likewise for a narrow-band output by:

$$\hat{s}_{NB}(n) = \begin{cases} g_{NG}^{(n)} \hat{s}_{LB}(n) & \text{if } g_{NG}^{(n)} < 1.0 \\ \hat{s}_{LB}(n) & \text{otherwise} \end{cases} \qquad n = 0,\cdots,39 \tag{8-87}$$

9 Description of the transmitted parameter indices

The bitstream ordering of R3 mode is given in Table 9-1 in descending order. For each parameter, the MSB is transmitted first.

Table 9-1 – Description of transmitted parameter indices in R3 mode

Layer	Symbol	Description	Bits
Layer 0	$I_{L0}(0)$	Lower-band core (G.711-compatible) – 1st sample in frame	8
	$I_{L0}(1)$	Lower-band core (G.711-compatible) – 2nd sample in frame	8
	…	…	…
	$I_{L0}(39)$	Lower-band core (G.711-compatible) – 40th sample in frame	8
Layer 1	$I_{L1}(0)$	Lower-band enhancement bit(s) – 1st sample in frame	$0 \le B_A(0) \le 3$
	$I_{L1}(1)$	Lower-band enhancement bit(s) – 2nd sample in frame	$0 \le B_A(1) \le 3$
	…	…	…
	$I_{L1}(39)$	Lower-band enhancement bit(s) – 40th sample in frame	$0 \le B_A(39) \le 3$
Layer 2	$I_{Hp0}(0)$	Higher-band MDCT sign index – 1st subvector	1
	$I_{Hs0}(0)$	Higher-band MDCT vector code – 1st subvector	5
	$I_{Hp1}(0)$	Higher-band MDCT sign index – 1st subvector	1
	$I_{Hs1}(0)$	Higher-band MDCT vector code – 1st subvector	5
	$I_{Hp0}(1)$	Higher-band MDCT sign index – 2nd subvector	1
	$I_{Hs0}(1)$	Higher-band MDCT vector code – 2nd subvector	5
	$I_{Hp1}(1)$	Higher-band MDCT sign index – 2nd subvector	1
	$I_{Hs1}(1)$	Higher-band MDCT vector code – 2nd subvector	5
	…	…	…
	$I_{Hp0}(5)$	Higher-band MDCT sign index – 6th subvector	1
	$I_{Hs0}(5)$	Higher-band MDCT vector code – 6th subvector	5
	$I_{Hp1}(5)$	Higher-band MDCT sign index – 6th subvector	1
	$I_{Hs1}(5)$	Higher-band MDCT vector code – 6th subvector	5
	I_{Hg}	Higher-band MDCT gain	8

For each frame, the layers should be placed in the order indicated in Figure 9-1.

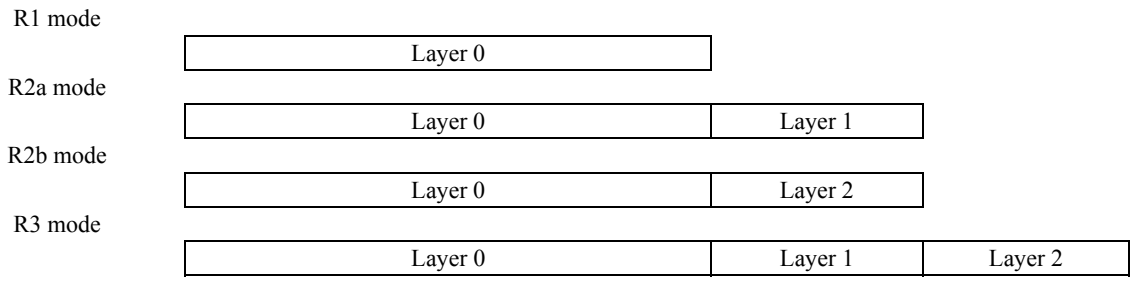


Figure 9-1 – Order for layer placement for the various modes

10 Bit-exact description of the G.711.1 codec

The ANSI C code simulating the G.711.1 codec in 16-bit fixed-point is available as an electronic attachment to this Recommendation. The following clauses summarize the use of this simulation code and how the software is organized.

10.1 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] <law> <infile> <codefile>

where

law	is the desired G.711 law (A or u)
infile	is the name of the input file to be encoded
codefile	is the name of the output bitstream file

Options:

-mode #	encoder mode (1, 2, 3, 4). Mode numbers 1, 2, 3, 4 correspond to R1, R2a, R2b, R3, respectively. When not specified, R3 will be generated by default.
-nb	narrow-band input
-hardbit	output bitstream file is in multiplexed hardbit format
-quiet	quiet processing

The command line for the decoder is as follows:

decoder [-options] <law> -mode <modenum> <codefile> <outfile>

where

law	is the desired G.711 law (A or u)
modenum	is the mode of the bitstream file (1, 2, 3, 4). Mode numbers 1, 2, 3, 4 correspond to R1, R2a, R2b, R3, respectively.
codefile	is the name of the input bitstream file
outfile	is the name of the decoded output file

Options:

-decmode #	is to specify the decoder mode (1, 2, 3, 4). Mode numbers 1, 2, 3, 4 correspond to R1, R2a, R2b, R3, respectively. When not specified, it will be the same as modenum (mode of the bitstream file).
-hardbit	input bitstream file is in multiplexed hardbit format
-quiet	quiet processing

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 [b-ITU-T G.192] bitstream format by default.

The mapping table of the encoded bitstream is contained in the simulation software.

10.2 Organization of the simulation software

Tables 10-1 to 10-4 describe the organization of the simulation software.

Table 10-1 – Tables in C-code

Table name	Symbol	Size	Format	Description
sQmf0	$h_0^{qmf}(i)$	16	Q15	QMF filter coefficients (polyphase representation)
sQmf1	$h_1^{qmf}(i)$	16	Q15	QMF filter coefficients (polyphase representation)
NS_window	$w_{LP1}(i)$	80	Q15	Window for LPC analysis in noise shaping
NS_lag_h	$w_{lag}(i)$	4	Q15	Lag windowing in noise shaping (MSB of double precision format)
NS_lag_l		4	Q16	Lag windowing in noise shaping (LSB of double precision format)
LBFEC_lag_h	$w_{lag}^{FERC}(i)$	16	Q15	Lag windowing in low-band FERC (MSB and LSB of double precision format)
LBFEC_lag_l		16	Q16	Lag windowing in low-band FERC (LSB of double precision format)
LBFEC_lpc_win_80	$w_{LP2}(i)$	80	Q15	Window for LPC analysis in low-band FEC
LBFEC_fir_lp	$H_{dec}(z)$	9	Q16	FIR decimation filter coefficients in low-band FERC
MDCT_xcos	–	25	Q15	Cosine table for FFT
MDCT_xsin	–	25	Q15	Sine table for FFT
MDCT_tab_map	–	10	Q0	Index mapping table for Good-Thomas FFT
MDCT_tab_map2	–	10	Q0	Index mapping table for Good-Thomas FFT
MDCT_tab_rev_ipp	–	1	Q0	Table for Good-Thomas FFT
MDCT_tab_rev_i	–	1	Q0	Table for Good-Thomas FFT
MDCT_rw1	–	20	Q15	FFT twiddle factors (cosine part)
MDCT_rw2	–	20	Q15	FFT twiddle factors (sine part)
MDCT_wcos	W_{80}^{n}	20	Q15	Cosine table for MDCT and iMDCT
MDCT_wsin		20	Q15	Sine table for MDCT and iMDCT
MDCT_wetr	$\frac{(-1)^{k+1} W_8^{-1} W_{320}^{4k+1}}{80}$	20	Q21	Table for complex post-multiplication in MDCT (real part)
MDCT_weti		20	Q21	Table for complex post-multiplication in MDCT (imaginary part)
MDCT_wetrm1	–	20	Q14	Table for complex pre-multiplication in iMDCT (real part)
MDCT_wetim1	–	20	Q14	Table for complex pre-multiplication in iMDCT (imaginary part)

Table 10-1 – Tables in C-code

Table name	Symbol	Size	Format	Description
MDCT_h	$w_{TDAC}(i)$	80	Q14	MDCT window
gsCodebook_0ch	$C_{H0w}(i,j)$	32*6	Q12	Interleave CSVQ sub-codebook 0
gsCodebook_1ch	$C_{H1w}(i,j)$	32*6	Q12	Interleave CSVQ sub-codebook 1
glCodebook_0ch_pow	$\sigma^2_{H0w}(i)$	32 (x2)	Q22	Power-term for CSVQ pre-selection
glCodebook_1ch_pow	$\sigma^2_{H1w}(i)$	32 (x2)	Q22	Power-term for CSVQ pre-selection
gsCodebook_cross	$R_{H0H1}(i_0,i_1)$	32*32	Q9	Cross-terms for CSVQ main selection
max_err_quant	$d'_{\max}(j)$	16	Q0	A-law quantization step size
Hann_sh16	$w_a(n)$	64	Q15	Asymmetric Hanning window (for 64-point FFT processing)
Hann_sh16_p6	$\gamma_w(n)$	7	Q15	Half of 16-point Hanning window for filter interpolation
Hann_sh16_p6m1	$1-\gamma_w(n)$	7	Q15	Complementary half of Hanning window for filter interpolation
r	–	64	Q0	Index mapping table for FFT
w	–	16	Q15	Twiddle factors for FFT
WinFilt	$Wt_p(n)$	17	Q15	Truncating window for impulse response of noise reduction filter
Table_sqrt_w	–	49	Q15	Table for square-root operation
Table_isqrt	–	49	Q15	Table for inverse square-root operation

Table 10-2 – Summary of encoder specific routines

Filename	Description
encoder.c	G.711.1 encoder interface
g711wbeenc.c	G.711.1 main encoder
prehpf.c	High-pass pre-filter
lowband_enc.c	Lower-band encoder
highband_enc.c	Higher-band encoder
mulaw.c	Mu-law compression of higher-band gain
norm_spectrum.c	MDCT spectrum normalization by RMS
vq_mainselect.c	CSVQ main selection routine
vq_preselect.c	CSVQ pre-selection routine
vqenc_spectrum.c	Interleave CSVQ spectrum coding

Table 10-3 – Summary of decoder specific routines

Filename	Description
decoder.c	G.711.1 decoder interface
g711wbedec.c	G.711.1 main decoder
fec_lowband.c	Frame erasure concealment (FERC) for lower-band signal
lowband_dec.c	Lower-band decoder
fec_highband.c	Frame erasure concealment (FERC) for higher-band signal
highband_dec.c	Higher-band decoder
mulawinv.c	Inverse mu-law compression of higher-band gain
vqdec_spectrum.c	Interleave CSVQ spectrum decoding

Table 10-4 – Summary of common routines

Filename	Description
qmfilt.c	QMF filterbank
softbit.c	Conversion between hardbit and softbit
table_qmfilt.c	Tables for QMF filterbank
autocorr_ns.c	Autocorrelation of signal for noise shaping
g711a.c	Embedded PCM coder and decoder (A-law)
g711mu.c	Embedded PCM coder and decoder (mu-law)
lpctools.c	Linear prediction tools
table_lowband.c	Tables for lower-band modules
cfft.c	20-point complex FFT/iFFT
mdct.c	80-point MDCT and iMDCT
mux_bits.c	Multiplexing/demultiplexing of higher-band parameters
table_highband.c	Tables for higher-band coding
table_mdct.c	Tables for MDCT computation
dsputil.c	Fixed-point utility routines
errexit.c	Exit routine
mathtool.c	Square-root routines
oper_32b.c	Basic operators in double precision (32 bits)
table_mathtool.c	Tables for square-root routines

Annex A

Floating-point implementation

(This annex forms an integral part of this Recommendation)

This annex describes the reference floating-point implementation of the G.711.1 coding algorithm.

A.1 Algorithmic description

This floating-point version of Recommendation ITU-T G.711.1 has the same algorithm steps as the fixed-point version. Similarly, the bit stream is identical to that of G.711.1. For algorithmic details, see main body.

A.2 ANSI C code

ANSI C source code simulating the floating-point version of Recommendation ITU-T G.711.1 defined in this annex is available as an electronic attachment to Annex A. The current version of this ANSI C source code is Version 1.0 of 18 September 2008. The structure of the floating-point source code is related to the corresponding fixed point source code. Tables A.1 to A.4 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 A.1 – Summary of encoder specific routines

Filename	Description
encoder.c	G.711.1 encoder interface
g711wbeenc.c	G.711.1 main encoder
prehpf.c	High-pass pre-filter
lowband_enc.c	Lower-band encoder
highband_enc.c	Higher-band encoder
mulaw.c	Mu-law compression of higher-band gain
norm_spectrum.c	MDCT spectrum normalization by RMS
vq_mainselect.c	CSVQ main selection routine
vq_preselect.c	CSVQ pre-selection routine
vqenc_spectrum.c	Interleave CSVQ spectrum coding

Table A.2 – Summary of decoder specific routines

Filename	Description
decoder.c	G.711.1 decoder interface
g711wbedec.c	G.711.1 main decoder
fec_lowband.c	Frame erasure concealment (FERC) for lower-band signal
lowband_dec.c	Lower-band decoder
fec_highband.c	Frame erasure concealment (FERC) for higher-band signal
highband_dec.c	Higher-band decoder
mulawinv.c	Inverse mu-law compression of higher-band gain
vqdec_spectrum.c	Interleave CSVQ spectrum decoding

Table A.3 – Summary of common routines

Filename	Description
qmfilt.c	QMF filter bank
softbit.c	Conversion between hardbit and softbit
table_qmfilt.c	Tables for QMF filter bank
autocorr_ns.c	Autocorrelation of signal for noise shaping
g711a.c	Embedded PCM coder and decoder (A-law)
g711mu.c	Embedded PCM coder and decoder (mu-law)
lpctools.c	Linear prediction tools
table_lowband.c	Tables for lower-band modules
cfft.c	20-point complex FFT/iFFT
mdct.c	80-point MDCT and iMDCT
mux_bits.c	Multiplexing/demultiplexing of higher-band parameters
table_highband.c	Tables for higher-band coding
table_mdct.c	Tables for MDCT computation
dsputil.c	Fixed-point utility routines
errexit.c	Exit routine
floatutil.c	Floating point specific routines
mathtool.c	Square-root routines
oper_32b.c	Basic operators in double precision (32 bits)
table_mathtool.c	Tables for square-root routines

Table A.4 – Summary of Appendix I specific routines

Filename	Description
post.c	Main routine that calls all post-processing subroutines
post_anasyn.c	Analysis/synthesis subroutines for post-processing
post_gainfct.c	Subroutines for estimating of the post-processing filter
post_rfft.c	64-point real FFT and iFFT
table_post.c	Tables for postfilter

Annex B

G.711.1 usage in H.245

(This annex forms an integral part of this Recommendation)

B.1 Scope

This annex defines the RTP payload format and capability signalling for the G.711.1 audio codec for usage with [ITU-T H.245]. Both the format and the capability parameters are fully compatible with the corresponding G.711.1 RTP definitions to allow seamless interoperability.

B.2 Packet structure for G.711.1 frames

The RTP [IETF RFC 3550] payload format for G.711.1 is specified in [IETF RFC 5391].

This payload format supports the multi-rate capability of the codec and allows for a rate change per packet by an additional payload header which conveys the "Mode Index" (MI). This parameter describes the modes of the following audio frame. Table B.1 gives the corresponding MI number to the bit-rate modes defined in Table 6-1.

Table B.1 – G.711.1 modes in mode index payload header

Mode index	G.711.1 bit-rate mode
1	R1
2	R2a
3	R2b
4	R3

All other values in MI must not be used, and payloads with an undefined MI value must be discarded. If a restricted mode-set has been set up by the signalling, as given in the G.711.1 capability parameter "modeSet", payloads received with an MI value not in this set must be discarded.

B.3 G.711.1 capability definitions for H.245

The G.711.1 capability is defined as a H.245 generic capability, following Appendix VII of [ITU-T H.245].

Tables B.1 and B.2 define the G.711.1 capability identifier, based on the G.711 core encoding laws. Two G.711.1 capability parameters are defined in Tables B.4 and B.5. Those two capability parameters apply to both core-encoding (μ-law and A-law) capability identifiers.

Table B.2 – G.711.1 (μ-law core) capability identifier

Capability name	G.711.1Ulaw
Capability class	Audio
Capability identifier type	Standard
Capability identifier value	{ itu-t(0) recommendation(0) g(7) g711(711) dot(1) part1(1) generic-capabilities(1) u-law(0) 0 }
maxBitRate	Shall be present. Possible values are 640, 800 and 960 (units 100 bit/s). This value must be consistent with the capability parameter "modeSet", if indicated. The general value for maxBitRate is 960 (= 96 kbit/s).
collapsing	The field shall contain the G.711.1 capability parameters from Table B.4.
nonCollapsing	This field shall contain the G.711.1 capability parameters from Table B.5.
nonCollapsingRaw	This field shall not be included.
Transport	This field shall not be included.

Table B.3 – G.711.1 (A-law core) capability identifier

Capability name	G.711.1Alaw
Capability class	Audio
Capability identifier type	Standard
Capability identifier value	{ itu-t(0) recommendation(0) g(7) g711(711) dot(1) part1(1) generic-capabilities(1) a-law(1) 0 }
maxBitRate	Shall be present. Possible values are 640, 800 and 960 (units 100 bit/s). This value must be consistent with the capability parameter "modeSet", if indicated. The general value for maxBitRate is 960 (= 96 kbit/s).
collapsing	The field shall contain the G.711.1 capability parameters from Table B.4.
nonCollapsing	This field shall contain the G.711.1 capability parameters from Table B.5.
nonCollapsingRaw	This field shall not be included.
Transport	This field shall not be included.

Table B.4 – G.711.1 capability parameter – maxAL-sduAudioFrames

Parameter name	maxAL-sduAudioFrames
Parameter description	This is a collapsing GenericParameter. It indicates the maximum number of audio frames per AL-SDU.
Parameter identifier value	1
Parameter status	Optional
Parameter type	unsignedMin
Supersedes	-

Table B.5 – G.711.1 capability parameter – modeSet

Parameter name	modeSet
Parameter description	This is a nonCollapsing GenericParameter. It indicates a supported bit-rate mode index from Table B.1. Each parameter can appear several times.
Parameter identifier value	2
Parameter status	Optional. If this parameter is omitted, it means that one is capable of all modes. Multiple instances of this parameter may be included, specifying the supported bit-rate modes in preference order. Bit-rate modes that exceed maxBitRate should not be signalled.
Parameter type	unsignedMin
Supersedes	-

B.4 Interoperability with G.711

The core of G.711.1 is fully compatible with G.711, and the transcoding effort can be minimized. The basic G.711 capabilities are defined in the H.245 AudioCapability structure. G.711.1 in μ-law core is compatible with g711Ulaw64k and G.711.1 with A-law core is compatible with g711Alaw64k. The only constraint is that G.711.1 has a 5-ms frame length, instead of sample-by-sample encoding in 8-kHz sampling-rate of G.711.

In the order of preference, the G.711.1 should appear before G.711 because it provides better quality.

Note well that the RTP payload format defined in [IETF RFC 5391] shall only be used for communications between two G.711.1 endpoints because it will not be understood by a legacy G.711 endpoint.

Appendix I

Lower-band postfiltering for R1 mode

(This appendix does not form an integral part of this Recommendation)

Quantization noise postfiltering may optionally be performed to reduce the PCM quantization noise in the lower band, when only the Layer 0 is decoded (R1 mode) which is compatible with a legacy G.711 bitstream. The main aim of the postfiltering is to reduce audible quantization noise in signals encoded with a legacy G.711 encoder. Note that the use of this postfilter in tandeming scenario should be avoided, e.g., in conference bridges, but may optionally be used at the decoder in an end-user terminal; it is assumed that such a terminal uses full 16-bit linear PCM resolution in analogue-to-digital conversion, without word truncation to 13 or 14 bits.

The input signal to this postfiltering is the signal decoded by the embedded PCM coder at 64 kbit/s (4 kHz bandwidth). Note that the postfilter is applicable to R1 mode, and $\hat{s}_{LB}(n)$ must be computed using equation 8-1, not equation 8-3. First, the characteristics of the coding noise are determined using *a priori* information about the properties of G.711. Then, a time-varying filter is computed. Finally, an enhanced signal is obtained by applying this filter to the decoded signal to reduce the PCM quantization noise.

As depicted in Figure I.1, the method for reducing noise in successive frames consists of applying the following process:

- The spectrum $\hat{s}_{LB}(k)$ of the input signal $\hat{s}_{LB}(n)$ is calculated by transformation into the frequency domain (see clause I.1).
- A frequency-dependent noise power spectral density (PSD) estimation $|N_{LB}(k)|^2$ is obtained after estimating the load factor Γ_{LB} of the current frame (see clause I.2).
- A transfer function of a noise-reduction filter $W_{p2}(k)$ is obtained by using those power spectral densities. The computation is based on a two-step procedure (see clause I.3).
- Finally, a block filtering operation (w_p), based on this transfer function and using overlap and save (OLS), is applied to the PCM decoded signal $\hat{s}_{LB}(n)$ to produce a signal with reduced quantization noise, $\hat{s}'_{LB}(n)$ (see clause I.4).

The procedures are common to both G.711 companding laws.

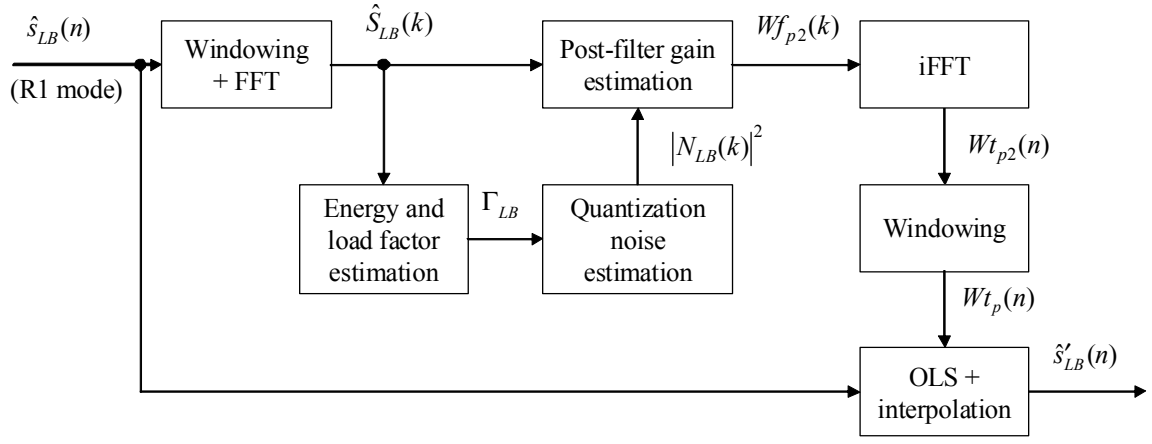


Figure I.1 – Block diagram of R1 postfilter

I.1 Windowing and FFT

The postfilter is estimated in the frequency domain using the fast Fourier transform (FFT). The FFT is computed on a block of 64 samples of the decoded signal, $\hat{s}_{LB}(n)$, $n=-24,\cdots,39$. Note that the negative indices refer to the past signal (the range $-24,\cdots,-1$ represents the most recent samples of the past frame). The 64-sample block $\hat{s}_{LB}(n)$, $n=-24,\ldots,39$ comprising 40 samples of the current frame and 24 samples of the previous frame, is transformed into a block $\hat{S}_{LB}(k)$, $k=0,\ldots,63$ of 64 frequency bins after windowing by a 64-point Hanning asymmetrical window (3/4 – 1/4), $w_a(n)$, which favours the most recent samples:

$$\begin{cases} w_a(n)=0.5-0.5\cos\left(\dfrac{\pi n}{48}\right) & n=0,...,47 \\ w_a(n)=0.5+0.5\cos\left(\dfrac{\pi(n-47)}{16}\right) & n=48,...,63 \end{cases} \tag{I-1}$$

$$\hat{S}_{LB}(k)=\mathrm{FFT}\{\hat{s}_{LB}(n-24)\cdot w_a(n)\} \tag{I-2}$$

Because of the symmetry of the Fourier transform (input data is real), only the first half of bin frequencies are considered, meaning that $k=0,\ldots,32$.

A very low complexity FFT implementation is used that takes into account that the input signal is real to save some computation complexity. The output spectrum is saved in the following sequence: [Re(0) Re(1) ... Re(32) Im(31) ... Im(1)].

I.2 PSD of the quantization noise

According to the properties of the G.711 coding, an estimation of the signal to quantization noise ratio (SNR), and then an estimation of the quantization noise PSD, can be obtained from a load factor Γ_{LB}. The load factor Γ_{LB} is defined as the inverse square root of the variance of the uncoded input signal $S_{LB}(n)$, and the estimation of the quantization noise PSD $|N_{LB}(k)|^2$ is directly derived from the variance of this signal. However, since in most cases the variance of the quantization noise is very low compared to the variance of the signal to be coded, the post-processing assumes that the variance of the input signal $S_{LB}(n)$ is equal to the variance of the PCM decoded signal $\hat{s}_{LB}(n)$, available at the decoder.

Based on the above assumption, the noise PSD $|N_{LB}(k)|^2$ is estimated from the variance of the PCM decoded signal $\hat{s}_{LB}(n)$ as follows:

1) The energy E_{LB} of the current frame of PCM decoded signal is computed with:

$$E_{LB}=\sum_{n=0}^{39}\hat{s}_{LB}^2(n) \tag{I-3}$$

2) The load factor Γ_{LB} is related to the frame energy as follows:

$$\Gamma_{LB}^2=\frac{40}{E_{LB}} \tag{I-4}$$

3) The load factor Γ_{LB} is compared with a threshold to determine if the signal has been coded on the uniform part of the quantizer or on its logarithmic part. So the PSD $|N_{LB}(k)|^2$ of the quantization noise is computed as follows:

- If $\Gamma_{LB}^2 < 4.8018 \cdot 10^2$ (i.e., $E_{LB} > 8.3303 \cdot 10^{-2}$), the signal has been coded in the logarithmic part of the quantizer and $|N_{LB}(k)|^2$ is computed as:

$$|N_{LB}(k)|^2 = 1.5276 \cdot 10^{-4} \cdot E_{LB} \tag{I-5}$$

- If $10^5 \geq \Gamma_{LB}^2 > 4.8018 \cdot 10^2$ (i.e., $4 \cdot 10^{-4} \leq E_{LB} < 8.3303 \cdot 10^{-2}$), the signal has been coded in the uniform part of the quantizer, and $|N_{LB}(k)|^2$ is computed as:

$$|N_{LB}(k)|^2 = 3.1789 \cdot 10^{-7} \tag{I-6}$$

- If $\hat{\Gamma}_{LB}^2 > 10^5$ (i.e., $E_{LB} < 4 \cdot 10^{-4}$), the signal has also been coded in the uniform part of the quantizer, but equation I-6 may not yield a good estimation. This case occurs for the highest load factor (i.e. the lowest frame energy). In that case, the variance of the quantization noise is forced to be 15 dB less than the signal variance:

$$|N_{LB}(k)|^2 = 3.1623 \cdot 10^{-2} \cdot E_{LB} \tag{I-7}$$

NOTE – In order to avoid time-consuming division operations in the ANSI C source code, the load factor Γ_{LB} is not computed and comparisons are performed instead with the frame energy (as expressed by the corresponding comparisons indicated above in parentheses).

I.3 Postfilter computation

I.3.1 Postfilter computation in frequency domain

As the variance of the noise has been estimated, a filter can be designed to reduce this quantization noise. The filter computation is obtained by a short-term spectral attenuation technique using a "two-step" procedure to estimate a Wiener filter.

The estimated PSD $|N_{LB}(k)|^2$ of the quantization noise yields the same noise level for each frequency bin. Without extra information on the noise spectral shape, the Wiener filter works as if the noise were white. Yet the processing is robust to limited deviations from the white noise assumption. Moreover, the robustness is increased by some *a posteriori* controls to avoid excessive attenuation and to limit the distortion that may be introduced by estimation errors.

During the first step, the *a posteriori* SNR for each frequency bin, $SNR_{post}(k)$, is estimated as:

$$SNR_{post}(k) = \frac{|\hat{S}_{LB}(k)|^2}{|N_{LB}(k)|^2} \qquad k = 0,...,32 \tag{I-8}$$

Note that in order to reduce complexity by limiting the number of divisions, the SNRs are "virtually multiplied" by $|N_{LB}(k)|^2$ in the ANSI C source code. Consequently the above equation becomes:

$$SNR_{post}(k) = |\hat{S}_{LB}(k)|^2 \qquad k = 0,...,32 \tag{I-9}$$

As a consequence, all the following SNR values do not correspond exactly to a signal-to-noise ratio.

Then, the first step *a priori* SNR on the current frame of index *m*, $SNR_{prio1}^{(m)}(k)$, is estimated with the decision-directed approach (with $\beta = 0.98$) using a first desired signal estimation, $\widetilde{S}_1^{(m-1)}(k)$, which is computed in the previous frame (see equation I-15):

$$SNR_{prio1}^{(m)}(k) = \beta \cdot \left|\widetilde{S}_1^{(m-1)}(k)\right|^2 + (1-\beta)\max\left[SNR_{post}^{(m)}(k) - \left|N_{LB}^{(m)}(k)\right|^2, 0\right], \qquad k = 0,...,32 \qquad \text{(I-10)}$$

Note that in this clause the frame indices *m* and (*m*-1) are only used for this equation because of time-dependency of the variable $\widetilde{S}_1^{(m-1)}(k)$ estimated at the previous frame. Thus, for clarity, frame indices are omitted in other equations for variables in the current frame *m*, e.g., $SNR_{post}(k) = SNR_{post}^{(m)}(k)$ in equation I-10.

Then a transfer function of a first noise reduction filter, $Wf_{p1}(k)$, is computed in the frequency domain using the first step *a priori* SNR, $SNR_{prio1}(k)$:

$$Wf_{p1}(k) = \frac{SNR_{prio1}(k)}{\left|N_{LB}(k)\right|^2 + SNR_{prio1}(k)} \qquad k = 0,...,32 \qquad \text{(I-11)}$$

A second desired signal estimation $\widetilde{S}_2(k)$ is computed in the frequency domain by combining the spectrum of the input signal $\hat{S}_{LB}(k)$ and the transfer function of the first noise reduction filter $Wf_{p1}(k)$:

$$\left|\widetilde{S}_2(k)\right| = \left|\hat{S}_{LB}(k)\right| \cdot \left|Wf_{p1}(k)\right| \qquad k = 0,...,32 \qquad \text{(I-12)}$$

During the second step, the *a priori* SNR estimation is refined by using $\widetilde{S}_2(k)$ as desired signal estimator:

$$SNR_{prio2}(k) = \left|\widetilde{S}_2(k)\right|^2, \qquad k = 0,...,32 \qquad \text{(I-13)}$$

Then a transfer function of a second noise reduction filter, $Wf_{p2}(k)$, is computed in the frequency domain using the second step *a priori* SNR, $SNR_{prio2}(k)$. Nevertheless, such a filter may attenuate too much some frequency bins. A threshold is then applied to avoid $Wf_{p2}(k)$ being lower than –6 dB:

$$Wf_{p2}(k) = \max\left(\frac{SNR_{prio2}(k)}{\left|N_{LB}(k)\right|^2 + SNR_{prio2}(k)}, 10^{-\frac{6}{20}}\right) \qquad k = 0,...,32 \qquad \text{(I-14)}$$

At this stage of the processing, the computation of the postfilter $Wf_{p2}(k)$ in the frequency domain for the current frame of index *m* is completed.

However, a first desired signal estimation $\widetilde{S}_1^{(m)}(k)$ is computed that will be used at the next frame (frame *m*+1) to estimate the first step *a priori* SNR, $\widetilde{S}_1^{(m-1)}(k)$ in equation I-10, which is obtained by multiplying $\hat{S}_{LB}(k)$ by $Wf_{p2}(k)$:

$$\left|\widetilde{S}_1^{(m)}(k)\right| = \left|\hat{S}_{LB}(k)\right| \cdot \left|Wf_{p2}(k)\right| \qquad k = 0,...,32 \qquad \text{(I-15)}$$

I.3.2 Postfilter computation in time domain

The filter $Wf_{p2}(k)$ is real (as the phase information is not processed). To compute the time filter coefficients, the algorithm processes the iFFT of $Wf_{p2}(k)$ with a zero-imaginary part, as described below.

$$Wt_{p2}(n) = iFFT\{Wf_{p2}(k)\} \qquad k = 0,...,32,\ n = 0,...,63 \qquad \text{(I-16)}$$

Since $Wf_{p2}(k)$ is real, the causality of the time filter must be restored by a swap procedure of the coefficients as defined below:

$$Wt_c(n) = \begin{cases} Wt_{p2}(n+32) & n = 0,\cdots,31 \\ Wt_{p2}(n-32) & n = 32,\cdots,63 \end{cases} \qquad \text{(I-17)}$$

The time impulse response is then truncated to 33 taps and weighted by a symmetric Hanning window to obtain a FIR filter of 33 taps with a delay of 16 samples, $Wt_p(n)$:

$$Wt_p(n) = \left(0.5 - 0.5\cos\left(\frac{\pi n}{16}\right)\right) \cdot Wt_c(n+15) \qquad n = 0,...,32 \qquad \text{(I-18)}$$

A very low complexity iFFT implementation is used that takes into account that the input spectrum is real and symmetrical, hence its output is also real. As the input spectrum is real and symmetrical, only 33 real input values are needed, the remaining 31 values are generated by mirroring and the imaginary part is considered as 0. The output is a 33-elements real-time domain signal.

I.4 Time domain postfiltering (time OLS with interpolation)

Finally, the PCM decoded signal $\hat{s}_{LB}(n)$ is filtered by the noise reduction filter Wt_p to produce a signal with reduced quantization noise, $\hat{s}'_{LB}(n)$.

An OLS method in the time domain is used to perform this filtering. In addition, as the filter Wt_p varies from one frame to the other, some audible discontinuities may occur; therefore, the transition between the filters is smoothed. For the first eight output samples $\hat{s}'_{LB}(n)$ of the current frame, the filter $\widetilde{W}t_{p,n}$ applied is a combination of the filters of the previous and current frames, $\widetilde{W}t_p^{(m-1)}$ and $\widetilde{W}t_p^{(m)}$, respectively. The filter $\widetilde{W}t_{p,n}$ applied to the input samples $\hat{s}_{LB}(n)$ to produce the *n-th* sample $\hat{s}'_{LB}(n)$ is given by:

$$\widetilde{W}t_{p,n}(i) = \gamma_w(n) \cdot \widetilde{W}t_p^{(m)}(i) + (1 - \gamma_w(n)) \cdot \widetilde{W}t_p^{(m-1)}(i) \qquad i = 0,...,32,\ n = 0,...,7 \qquad \text{(I-19)}$$

where the weighting coefficient $\gamma_w(n)$ is the first half of a 16-length Hanning window:

$$\gamma_w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{15}\right) \qquad n = 0,...,7 \qquad \text{(I-20)}$$

Then the eight first filtered samples $\hat{s}'_{LB}(n-16)$ are given by the following convolution:

$$\hat{s}'_{LB}(n-16) = \sum_{i=0}^{32} \widetilde{W}t_{p,n}(i) \cdot \hat{s}_{LB}(n-i) \qquad n = 0,...,7 \qquad \text{(I-21)}$$

The other filtered samples $\hat{s}'_{LB}(n-16)$, for n = 8,...,39, are obtained by filtering the input data with the current frame filter $\tilde{W}t_p^{(m)}$ also noted Wt_p:

$$\hat{s}'_{LB}(n-16) = \sum_{i=0}^{32} Wt_p(i) \cdot \hat{s}_{LB}(n-i) \qquad n = 8,...,39 \qquad \text{(I-22)}$$

Note that the first 16 samples $\hat{s}'_{LB}(n-16)$ obtained at frame m, correspond to the last 16 samples of the previous frame (m-1) ($\hat{s}'^{(m-1)}_{LB}(n)$, for $n = 24,\cdots,39$) while the last 24 samples obtained at frame m correspond to the first 24 samples of the current frame m: $\hat{s}'^{(m)}_{LB}(n)$, $n = 0,\cdots,23$.

I.5 Amplitude modification control

The final stage of the postfiltering consists in limiting the amplitude modification. It consists in limiting the modification introduced by the postfiltering with respect to the decoded signal $\hat{s}_{LB}(n)$. This limitation is only applied when $\left|\hat{s}'_{LB}(n)\right| > 24$. The limited postfiltered signal $\hat{s}''_{LB}(n)$ is either the current amplitude of the postfiltered signal $\hat{s}'_{LB}(n)$ if $\left|\hat{s}'_{LB}(n) - \hat{s}_{LB}(n)\right| \le d'_{\max}(\tilde{\eta}_{LB}(n))$ or otherwise an intermediary value between $\hat{s}'_{LB}(n)$ and $\hat{s}_{LB}(n)$:

If $\hat{s}'_{LB}(n) < \hat{s}_{LB}(n) - d'_{\max}(\tilde{\eta}_{LB}(n))$, $\hat{s}''_{LB}(n) = \hat{s}_{LB}(n) - d'_{\max}(\tilde{\eta}_{LB}(n))$

If $\hat{s}'_{LB}(n) > \hat{s}_{LB}(n) + d'_{\max}(\tilde{\eta}_{LB}(n))$, $\hat{s}''_{LB}(n) = \hat{s}_{LB}(n) + d'_{\max}(\tilde{\eta}_{LB}(n))$

where $\tilde{\eta}_{LB}(n) = 14 - \lfloor \log_2(\left|\hat{s}_{LB}(n)\right|) \rfloor$ and the allowed maximum distance $d'_{\max}$ are given in Table I.1.

Table I.1 – Allowed maximum distances for postfiltering

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$d'_{\max}(j)$	512	256	128	64	32	16	8	8	8	8	8	8	8	8	8

A consequence of the combination of linear phase FIR filtering and this limitation is that the post-processing can be seamlessly switched off and on at any time.

I.6 Bit-exact description of the postfilter

The ANSI C code simulating the postfilter to G.711.1 decoder in 16-bit fixed-point is contained in the electronic attachment to this Recommendation. In order to enable the postfilter, the G.711.1 decoder source code must be compiled with "-DAPPENDIX_I_POSTFILTER" compiling directive. The following clauses summarize the use of this simulation code, and how the software is organized.

I.6.1 Use of the simulation software

With the postfilter enabled, the command line for the decoder can now accept a new option "-r1pf", as shown as follows:

decoder [-options] <law> -mode <modenum> <codefile> <outfile>

where

law	is the desired G.711 law (A or u)
modenum	is the mode of the bitstream file (1, 2, 3, 4) Mode numbers 1, 2, 3, 4 correspond to R1, R2a, R2b, R3 respectively.
codefile	is the name of the input bitstream file

outfile	is the name of the decoded output file
Options:	
-decmode #	is to specify the decoder mode (1, 2, 3, 4).
	Mode numbers 1, 2, 3, 4 correspond to R1, R2a, R2b, R3 respectively.
	When not specified, it will be the same as modenum (mode of the bitstream file).
-r1pf	R1 postfilter enabled
	This option is for R1 mode only.
-hardbit	Input bitstream file is in multiplexed hardbit format.
-quiet	quiet processing

I.6.2 Organization of the simulation software

Table I.2 describes the summary of the postfilter simulation software routines. Note that when compiled without the "-DAPPENDIX_I_POSTFILTER" directive, those files do not need to be compiled together with the other parts of the G.711.1 decoder.

Table I.2 – Summary of Appendix I specific routines

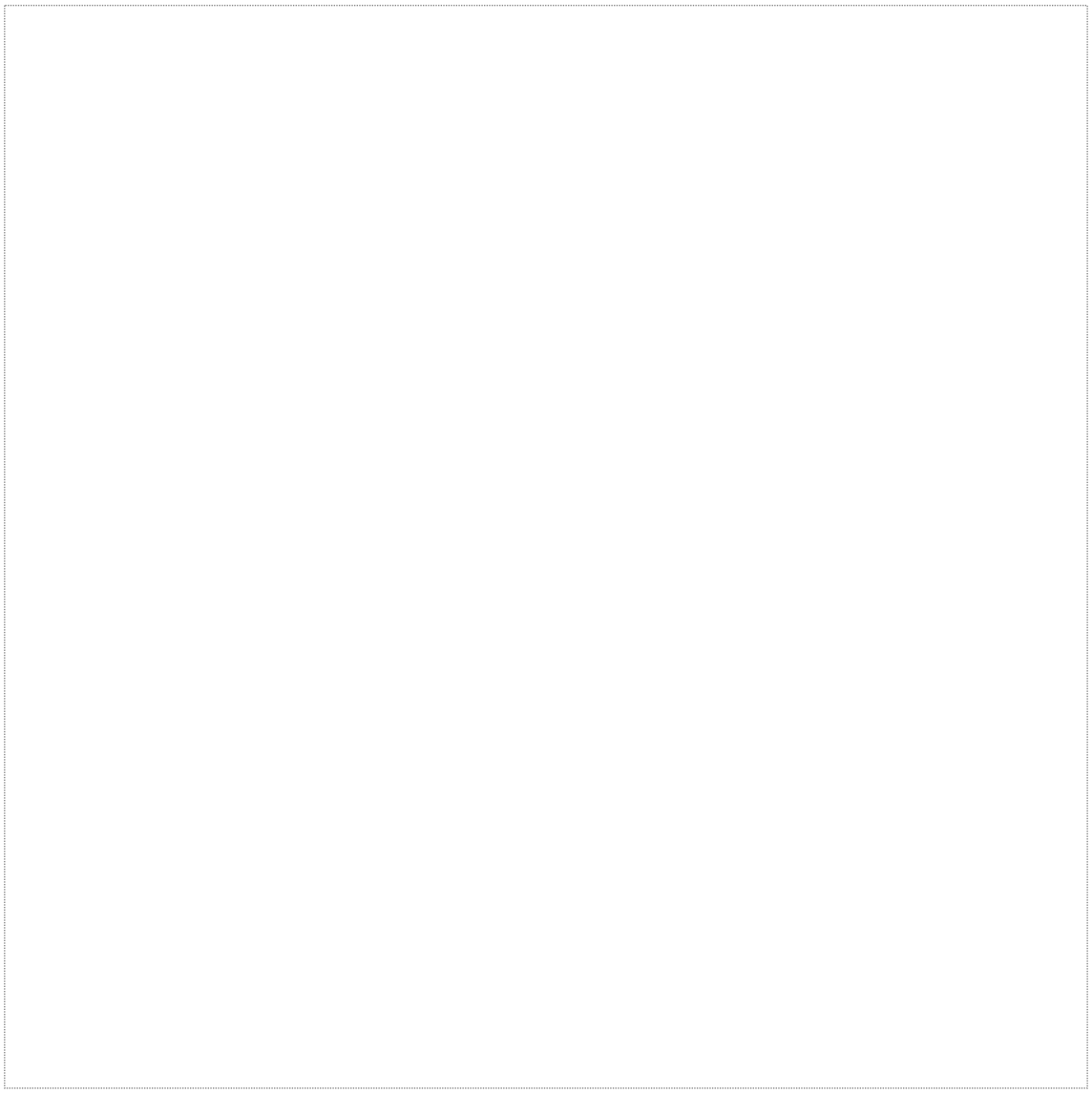
Filename	Description
post.c	Main routine that calls all post-processing subroutines
post_anasyn.c	Analysis/synthesis subroutines for post-processing
post_gainfct.c	Subroutines for estimating of the post-processing filter
post_rfft.c	64-point real FFT and iFFT
table_post.c	Tables for postfilter

Bibliography

[b-ITU-T G.192] Recommendation ITU-T G.192 (1996), *A common digital parallel interface for speech standardization activities*.

[b-Blahut] Blahut, R.E. (1985), *Fast Algorithms for Digital Signal Processing*, Addison-Wesley.

[b-Levinson] Levinson, N. (1947), *The Wiener RMS (root mean square) error criterion in filter design and prediction*, Journal of Mathematical Physics, Vol. 25, No. 4, pp. 261-278.



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	Telephone transmission quality, telephone installations, local line networks
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, 2009