3RD GENERATION
PARTNERSHIP
PROJECT 2
"3GPP2"

Enhanced Variable Rate Codec, Speech Service Options 3, 68, 70, and 73 for Wideband Spread Spectrum Digital Systems

REVISION HISTORY

Revision		Date
C.S0014-0 v1.0	Enhanced Variable Rate Codec (EVRC)	December 1999
C.S0014-0-1	Enhanced Variable Rate Codec, Speech Service Option 3 for Wideband Spread Spectrum Systems – Addendum 1	December 1999
C.S0014-0-2	Enhanced Variable Rate Codec, Speech Service Option 3 for Wideband Spread Spectrum Digital Systems – Addendum 2	December 1999
C.S0014-0-3 v1.0	EVRC TTY/TDD Extension – Release 0 – Addendum 3	May 2001
C.S0014-A v1.0	Enhanced Variable Rate Codec, Speech Service Option 3 for Wideband Spread Spectrum Digital Systems	May 2004
C.S0014-B v1.0	Enhanced Variable Rate Codec, Speech Service Options 3 and 68 for Wideband Spread Spectrum Digital Systems	May 2006
C.S0014-C v1.0	Enhanced Variable Rate Codec, Speech Service Options 3, 68, and 70 for Wideband Spread Spectrum Digital Systems	February 2007
C.S0014-D v1.0	Enhanced Variable Rate Codec, Speech Services Options 3, 68, 70, and 73 for Wideband Spread Spectrum Digital Systems	May 2009
C.S0014-D v2.0	Enhanced Variable Rate Codec, Speech Services Options 3, 68, 70, and 73 for Wideband Spread Spectrum Digital Systems	January 2010
C.S0014-D v3.0	Enhanced Variable Rate Codec, Speech Services Options 3, 68, 70, and 73 for Wideband Spread Spectrum Digital Systems	October 2010

FOREWORD

(This foreword is not part of this Standard.)

The scope of this document specifies the technical requirements that form a standard for Service Options 3, 68, 70, and 73, enhanced variable rate, two-way speech service options known as EVRC-A, EVRC-B, EVRC-WB, and EVRC-NW respectively. These service options communicate at one of four channel rates corresponding to the 9600 bps, 4800 bps, 2400 bps, and 1200 bps frame rates. However, Service Options 3 and 70 do not use 2400 bps channel frame rate. The speech coder source bit-rate corresponding to the above mentioned 4 channel rates are 8550 bps, 4000 bps, 2000 bps, and 800 bps. Service Options 68, 70, and 73 have the capability to operate in multiple capacity operating points at different average rates of operation that can be used for obtaining different system capacity while trading off speech quality gracefully.

This standard does not address the quality or reliability of Service Options 3, 68, 70, and 73 nor does it cover equipment performance or measurement procedures.

NOTES

1. The appropriate version of the Minimum Performance Standard for the enhanced variable rate codecs, EVRC-A, EVRC-B, EVRC-WB and EVRC-NW provides specifications and measurement methods.

2. "Base station" refers to the functions performed on the landline side, which are typically distributed among a cell, a sector of a cell, a mobile switching center, and a personal communications switching center.

3. ~~This document uses the following verbal forms: "Shall" and "shall not" identify requirements to be followed strictly to conform to the standard and from which no deviation is permitted. "Should" and "should not" indicate that one of several possibilities is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited. "May" and "need not" indicate a course of action permissible within the limits of the standard. "Can" and "cannot" are used for statements of possibility and capability, whether material, physical, or causal.~~

~~4.~~3. Footnotes appear at various points in this specification to elaborate and further clarify items discussed in the body of the specification.

~~5.~~4. Unless indicated otherwise, this document presents numbers in decimal form.

Binary numbers are distinguished in the text by the use of single quotation marks. In some tables, binary values may appear without single quotation marks if table notation clearly specifies that values are binary. The character 'x' is used to represent a binary bit of unspecified value. For example 'xxx00010' represents any 8-bit binary value such that the least significant five bits equal '00010'.

Hexadecimal numbers (base 16) are distinguished in the text by use of the form 0xh…h, where h…h represents a string of hexadecimal digits. For example, 0x2FA1 represents a number whose binary value is '10111110100001' and whose decimal value is 12913.

~~6.~~5. "Full-Rate" and "Rate 1" are used interchangeably in this document. The same rule applies to "Half-Rate" and "Rate 1/2," "Quarter-Rate" and "Rate 1/4," and "Eighth-Rate" and "Rate 1/8."

~~7.~~6. The following conventions apply to mathematical expressions in this standard:

   o $\lfloor x \rfloor$ indicates the largest integer less than or equal to x: $\lfloor 1.1 \rfloor = 1$, $\lfloor 1.0 \rfloor = 1$, and $\lfloor -1.1 \rfloor = -2$.

   o $\lceil x \rceil$ indicates the smallest integer greater than or equal to x: $\lceil 1.1 \rceil = 2$, $\lceil 2.0 \rceil = 2$, and $\lceil -1.1 \rceil = -1$.

   o |x| indicates the absolute value of x: $|-17| = 17$, $|17| = 17$.

   o $\oplus$ indicates exclusive OR.

   o min(x, y) indicates the minimum of x and y.

   o max(x, y) indicates the maximum of x and y.

   o In figures, × indicates multiplication. In formulas within the text, multiplication is implicit. For example, if h(n) and $p_L(n)$ are functions, then $h(n)\, p_L(n) = h(n) \times p_L(n)$.

   o x mod y indicates the remainder after dividing x by y: $x \bmod y = x - (y \lfloor x/y \rfloor)$.

   o round(x) is traditional rounding: $\text{round}(x) = \text{sign}(x) \lfloor |x| + 0.5 \rfloor$, where

$$\text{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}.$$

o $\sum$ indicates summation. If the summation symbol specifies initial and terminal values, and the initial value is greater than the terminal value, then the value of the summation is 0. For example, if N=0, and if f(n) represents an arbitrary function, then

$$\sum_{n=1}^{N} f(n) = 0.$$

o The bracket operator, [ ], isolates individual bits of a binary value. VAR[n] refers to bit n of the binary representation of the value of the variable VAR, such that VAR[0] is the least significant bit of VAR. The value of VAR[n] is either 0 or 1.

o Unless otherwise specified log(x) denotes logarithm at base 10 throughout this document.

REFERENCES

The following standards contain provisions which, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

NORMATIVE REFERENCES

N1.   C.S0005-E, *Upper Layer (Layer 3) Signalling Standard for cdma2000 Spread Spectrum Systems*, September 2009.

N2.   C.S0076-0, *Discontinuous Transmission (DTX) of Speech in cdma2000 Systems*, December 2005.

N3.   TIA-95-B (R2004), *Mobile Station-Base Station Compatibility Standard for Wideband Spread Spectrum Cellular Systems*, October 2004.

N4.   C.S003-E, *Medium Access Control (MAC) Standard for cdma2000 Spread Spectrum Systems*, September 2009.

N5.   ANSI/EIA/TIA-579-A, *Telecommunications Telephone Terminal Equipment Transmission Requirements for Digital Wireline Telephones*, October 1998.

N6.   ITU-T Recommendation G.711, *Pulse Code Modulation (PCM) of Voice Frequencies*, November 1988.

N7.   ITU-T Recommendation G.712, *Transmission Performance Characteristics of Pulse Code Modulation*, November 2001.

N8.   IEEE Standard 269-2002, *IEEE Standard Methods for Measuring Transmission Performance of Analog and Digital Telephone Sets, Handsets, and Headsets*, April 2003.

N9.   IEEE Standard 661-1979 (R2008), *Method for Determining Objective Loudness Ratings of Telephone Connections*, September 2008.

N10.   C.S0002-E, *Physical Layer Standard for cdma2000 Spread Spectrum Systems*, September 2009.

N11.   ITU-T Recommendation V.18, *Operational and interworking requirements for DCEs operating in the text telephone mode*, November 2000.

N12.   ITU-T Recommendation P.311, *Transmission Characteristics for Wideband (150-7000 Hz) Digital Handset Telephones*, June 2005.

N13.   ITU-T Recommendation P.341, *Transmission Characteristics for Wideband (150-7000 Hz) Digital Hands-free Telephony Terminals*, June 2005.

INFORMATIVE REFERENCES

S1.   C.S0014-0 v1.0, *Enhanced Variable Rate Codec (EVRC)*, December 1999.

S2.   C.R1001-G, *Administration of ParameterValue Assignments for cdma2000 Spread Spectrum Standards*, June 2009.

S3.   ITU-T Recommendation G.168, *Digital network echo cancellers*, March 2009.

R1.   Proakis, J. G. and Manolakis, D. G., *Introduction to Digital Signal Processing*, Macmillan, New York, 1988.

R2.   Rabiner, L. R. and Schafer, R. W., *Digital Processing of Speech Signals*, Prentice-Hall Inc., New Jersey 1978.

R3.   Crochiere, R. E. and Rabiner L. R., *Multirate Digital Signal Processing*, Prentice-Hall Inc., New Jersey 1983.

R4.   Oppenheim, A. V. and Schafer, R. W., *Digital Signal Processing*, Prentice-Hall Inc., New Jersey 1975.

R5.   Dellar, J. R., Proakis, J. G., Hansen, J. H. L., *Discrete-Time Processing of Speech Signals*, Macmillan, New York, 1993.Reserve.

R6.   Alejandro, A., *Acoustical and Environmental Robustness in Automatic Speech Recognition*, Kluwer Academic Publishers, Boston, 1993.

R7.   Atal, B. S. and Schroeder, M. R., "*Stochastic coding of speech at very low bit rates,*" Proc Int. Conf. Comm., Amsterdam, 1984, pp 1610-1613.

R8.   Kleijn, W. B., Kroon, P., and Nahumi, D., "*The RCELP Speech-Coding Algorithm,*" European Transactions on Telecommunications, Vol 5, No 5. Sept/Oct 1994, pp 573-582.

R9.   Nahumi, D and Kleijn, W. B., "*An Improved 8 kb/s RCELP Coder,*" IEEE Workshop on Speech Coding, 1995.

R10.   U. Mittal, J. P. Ashley, and E. M. Cruz-zeno, "*Low Complexity Joint Optimization of Excitation Parameters in Analysis-by-Synthesis Speech Coding,*" Proceedings Eurospeech 2003, pp. 1261-1264.

R11.   Laflamme, C., Adoul, J-P., Salami, R., Morissette, S., and Mabilleau, P., "*16 kbps wideband speech coding technique based on algebraic CELP,*" Proc. ICASSP 1991, pp. 13-16.

R12.   Salami, R., Laflamme, C., Adoul, J-P., et.al, "*A Toll Quality 8 kbits Speech coder for Personal Communication Systems (PCS),*" IEEE Trans. on Vehicle Technology, 1994.

R13.   Ashley, J.P., Cruz-Zeno, E.M., Mittal, U., Peng,W., "*Wideband coding of speech using a scalable pulse codebook,*" 2000 IEEE Workshop on Speech Coding, September 2000, pp 148-150.

R14.   Gerson, I.A.; Jasiuk, M.A., "*Techniques for improving the performance of CELP-type speech coders,*" Selected Areas in Communications, IEEE Journal, Volume 10, Issue 5, June 1992 Page(s):858 - 865.

TABLE OF CONTENTS

LIST OF FIGURES

5

LIST OF TABLES

1

2

3

4

5

6   This page intentionally left blank.

1 GENERAL

1.1 Scope

Service Options 3, 68, 70, and 73 provide two-way voice communications between the base station and the mobile station using the dynamically variable data rate speech codec algorithm described in this standard. The transmitting speech codec takes voice samples and generates an encoded speech packet for every Traffic Channel frame.[†] The receiving station generates a speech packet from every Traffic Channel frame and supplies it to the speech codec for decoding into voice samples.

Speech codecs communicate at one of four rates corresponding to the 9600 bps, 4800 bps, 2400 bps, and 1200 bps frame rates. However, Service Options 3 and 70 do not use 2400 bps frame rate.

The specifications defined in Sections 4 and 5 of this document provide the detailed algorithmic description of the EVRC. In the case of a discrepancy between the floating point and algorithmic descriptions, the reference floating point c-code will prevail. The specifications defined in Input Signal Preprocessing (see 4.4), Determining the Data Rate (see 4.7), and Adaptive Postfilter (see 5.8 and 5.9) are optional for implementations intended for varying operational environments (such as in-vehicle hands-free). Any implementations which deviate from the algorithms specified in this standard, shall meet the minimum performance requirements defined in the associated Minimum Performance Specifications.

1.2 Requirements Language

"Shall" and "shall not" identify requirements to be followed strictly to conform to this document and from which no deviation is permitted. "Should" and "should not" indicate that one of several possibilities is recommended as particularly suitable, without mentioning or excluding others, that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited. "May" and "need not" indicate a course of action permissible within the limits of the document. "Can" and "cannot" are used for statements of possibility and capability, whether material, physical or causal.

1.3 References

The following standards contain provisions which, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

1.3.1 Normative References

N1.   C.S0005-E, *Upper Layer (Layer 3) Signalling Standard for cdma2000 Spread Spectrum Systems*, September 2009.

N2.   C.S0076-0, *Discontinuous Transmission (DTX) of Speech in cdma2000 Systems*, December 2005.

N3.   TIA-95-B (R2004), *Mobile Station-Base Station Compatibility Standard for Wideband Spread Spectrum Cellular Systems*, October 2004.

[†] Common speech codec terminology also uses the term frame to represent a quantum of processing. For Service Options 3, 68, 70, and 73, the speech codec frame corresponds to speech sampled over 20 ms. The speech samples are processed into a packet. This packet is transmitted in a Traffic Channel frame.

N4. C.S003-E, *Medium Access Control (MAC) Standard for cdma2000 Spread Spectrum Systems*, September 2009.

N5. ANSI/EIA/TIA-579-A, *Telecommunications Telephone Terminal Equipment Transmission Requirements for Digital Wireline Telephones*, October 1998.

N6. ITU-T Recommendation G.711, *Pulse Code Modulation (PCM) of Voice Frequencies*, November 1988.

N7. ITU-T Recommendation G.712, *Transmission Performance Characteristics of Pulse Code Modulation*, November 2001.

N8. IEEE Standard 269-2002, *IEEE Standard Methods for Measuring Transmission Performance of Analog and Digital Telephone Sets, Handsets, and Headsets*, April 2003.

N9. IEEE Standard 661-1979 (R2008), *Method for Determining Objective Loudness Ratings of Telephone Connections*, September 2008.

N10. C.S0002-E, *Physical Layer Standard for cdma2000 Spread Spectrum Systems*, September 2009.

N11. ITU-T Recommendation V.18, *Operational and interworking requirements for DCEs operating in the text telephone mode*, November 2000.

N12. ITU-T Recommendation P.311, *Transmission Characteristics for Wideband (150-7000 Hz) Digital Handset Telephones*, June 2005.

N13. ITU-T Recommendation P.341, Transmission Characteristics for Wideband (150-7000 Hz) Digital Hands-free Telephony Terminals, June 2005.

1.3.2 Informative References

S1. C.S0014-0 v1.0, *Enhanced Variable Rate Codec (EVRC),* December 1999.

S2. C.R1001-G, *Administration of ParameterValue Assignments for cdma2000 Spread Spectrum Standards*, June 2009.

S3. ITU-T Recommendation G.168, *Digital network echo cancellers*, March 2009.

R1. Proakis, J. G. and Manolakis, D. G., *Introduction to Digital Signal Processing,* Macmillan, New York, 1988.

R2. Rabiner, L. R. and Schafer, R. W., *Digital Processing of Speech Signals*, Prentice-Hall Inc., New Jersey 1978.

R3. Crochiere, R. E. and Rabiner L. R., *Multirate Digital Signal Processing*, Prentice-Hall Inc., New Jersey 1983.

R4. Oppenheim, A. V. and Schafer, R. W., *Digital Signal Processing*, Prentice-Hall Inc., New Jersey 1975.

R5. Dellar, J. R., Proakis, J. G., Hansen, J. H. L., *Discrete-Time Processing of Speech Signals*, Macmillan, New York, 1993.Reserve.

R6. Alejandro, A., *Acoustical and Environmental Robustness in Automatic Speech Recognition*, Kluwer Academic Publishers, Boston, 1993.

R7. Atal, B. S. and Schroeder, M. R., "*Stochastic coding of speech at very low bit rates,*" Proc Int. Conf. Comm., Amsterdam, 1984, pp 1610-1613.

R8. Kleijn, W. B., Kroon, P., and Nahumi, D., "*The RCELP Speech-Coding Algorithm,*" European Transactions on Telecommunications, Vol 5, No 5. Sept/Oct 1994, pp 573-582.

R9. Nahumi, D and Kleijn, W. B., "*An Improved 8 kb/s RCELP Coder,*" IEEE Workshop on Speech Coding, 1995.

R10. U. Mittal, J. P. Ashley, and E. M. Cruz-zeno, "*Low Complexity Joint Optimization of Excitation Parameters in Analysis-by-Synthesis Speech Coding,*" Proceedings Eurospeech 2003, pp. 1261-1264.

R11. Laflamme, C., Adoul, J-P., Salami, R., Morissette, S., and Mabilleau, P., "*16 kbps wideband speech coding technique based on algebraic CELP,*" Proc. ICASSP 1991, pp. 13-16.

R12. Salami, R., Laflamme, C., Adoul, J-P., et.al, "*A Toll Quality 8 kbits Speech coder for Personal Communication Systems (PCS),*" IEEE Trans. on Vehicle Technology, 1994.

R13. Ashley, J.P., Cruz-Zeno, E.M., Mittal, U., Peng, W., "*Wideband coding of speech using a scalable pulse codebook,*" 2000 IEEE Workshop on Speech Coding, September 2000, pp 148-150.

R14. Gerson, I.A.; Jasiuk, M.A., "*Techniques for improving the performance of CELP-type speech coders,*" Selected Areas in Communications, IEEE Journal, Volume 10, Issue 5, June 1992 Page(s):858 – 865.

~~1.2~~ 1.4 Service Option Number

The variable data rate two-way voice service option using the speech codec algorithm described by this standard shall use service option numbers 3, 68, 70, and 73 and shall be called Service Options 3, 68, 70, and 73 respectively.

~~1.3~~ 1.5 Allowable Delays

~~1.3.1~~ 1.5.1 Allowable Transmitting Speech Codec Encoding Delay

The transmitting speech codec shall supply a packet to the multiplex sublayer no later than 20 ms after it has obtained the last input sample for the current speech frame.

~~1.3.2~~ 1.5.2 Allowable Receiving Speech Codec Decoding Delay

The receiving decoder shall generate the first sample of speech using parameters from a packet supplied to it by the multiplex sublayer not later than 3 ms after being supplied the packet.

~~1.4~~ 1.6 Special Cases

~~1.4.1~~ 1.6.1 Blanked Packets

A blanked frame occurs when the transmitting station uses the entire frame for either signaling traffic or secondary traffic. The EVRC does no special encode processing during the generation of a blank packet, i.e., the generated voice packet is simply not used. The decoder, in turn, treats a blank packet in the same manner as a frame erasure.

~~1.4.2~~ 1.6.2 Null Traffic Channel Data

A Rate 1/8 packet with all bits set to '1' is considered as null Traffic Channel data. This packet is declared an erased packet and handled as described in Section 5. For EVRC-A, if more than 2 consecutive all-ones Rate 1/8 packets are received, the decoder's output shall be muted until a valid packet is received.

~~1.4.3~~ 1.6.3 All Zeros Packet

Rate 1 and Rate 1/2 packets with all bits set to '0' shall be considered erased frames and shall be handled as described in Section 5. In addition, for EVRC-B, EVRC-WB, and EVRC-NW, Rate 1/4 and Rate 1/8 packets with all bits set to '0' shall be considered erased and shall be handled as described in Section 5.

~~1.5~~ 1.7 Terms and Numeric Information

ACB. Adaptive Codebook.

ACELP. Algebraic Code Excited Linear Predictive Coding, the algorithm that is used by the EVRC to generate the stochastic component of the excitation.

Autocorrelation Function. A function showing the relationship of a signal with a time-shifted version of itself.

Base Station. A station in the Domestic Public Radio Telecommunications Service, other than a mobile station, used for radio communications with mobile stations.

CELP. See Code Excited Linear Predictive Coding.

Codec. The combination of an encoder and decoder in series (encoder/decoder).

Code Excited Linear Predictive Coding (CELP). A speech coding algorithm. CELP codecs use codebook excitation, a long-term pitch prediction filter, and a short-term formant prediction filter.

Codebook. A set of vectors used by the speech codec. For each speech codec codebook subframe, one particular vector is chosen and used to excite the speech codec's filters. The codebook vector is chosen to minimize the weighted error between the original and synthesized speech after the pitch and formant synthesis filter coefficients have been determined.

Decoder. Generally, a device for the translation of a signal from a digital representation into an analog format. For this standard, a device which converts speech encoded in the format specified in this standard to analog or an equivalent PCM representation.

DFS. Discrete Fourier Series

DFT. See Discrete Fourier Transform.

Discrete Fourier Transform (DFT). A method of transforming a time domain sequence into a corresponding frequency domain sequence.

Encoder. Generally, a device for the translation of a signal into a digital representation. For this standard, a device which converts speech from an analog or its equivalent PCM representation to the digital representation described in this standard.

EVRC. Enhanced Variable Rate Codec, a family of source-controlled variable rate codecs including EVRC-A, EVRC-B, EVRC-WB, and EVRC-NW.

EVRC-A. A narrowband Enhanced Variable Rate Codec corresponding to speech service option 3.

EVRC-B. A narrowband Enhanced Variable Rate Codec corresponding to speech service option 68, capable to operate at multiple average rates or capacity operating points.

EVRC-WB. A wideband Enhanced Variable Rate Codec, corresponding to speech service option 70, capable to operate at one wideband mode and two narrowband modes interoperable with EVRC-B.

EVRC-NW. A narrowband-wideband Enhanced Variable Rate Codec, corresponding to speech service option 73, combining the wideband capability of EVRC-WB, the multiple capacity operating point capability of EVRC-B, and capability for discontinuous transmission in circuit-switch domain.

FCB. Fixed Codebook.

FFT. See Fast Fourier Transform.

Fast Fourier Transform (FFT). An efficient implementation of the Discrete Fourier Transform.

Formant. A resonant frequency of the human vocal tract causing a peak in the short-term spectrum of speech.

IDFT. See Inverse Discrete Fourier Transform.

IIR Filter. An infinite-duration impulse response filter is a filter for which the output, in response to an impulse input, never totally dies away. This term is usually used in reference to digital filters.

Interpolation. In the speech coder context, a means of smoothing the transitions of estimated parameters from one set to another. Usually a linear function.

Inverse Discrete Fourier Transform (IDFT). A method of transforming a frequency domain sequence into a corresponding time domain sequence.

ITU. International Telecommunication Union.

Linear Predictive Coding (LPC). A method of predicting future samples of a sequence by a linear combination of the previous samples of the same sequence. Linear Predictive Coding is frequently used in reference to a class of speech codecs.

Line Spectral Pair (LSP). A representation of digital filter coefficients in a pseudo-frequency domain. This representation has good quantization and interpolation properties.

LPC. See Linear Predictive Coding.

LSB. Least significant bit.

LSP. See Line Spectral Pair.

Mobile Station. A station in the Domestic Public Radio Telecommunications Service intended to be used while in motion or during halts at unspecified points. It is assumed that mobile stations include portable units (e.g., hand-held personal units) and units installed in vehicles

MSB. Most significant bit.

Packet. The unit of information exchanged between service option applications in the base station and the mobile station.

Pitch. The fundamental frequency in speech caused by the periodic vibration of the human vocal cords.

PPP. Prototype Pitch Period

PSTN. Public Switched Telephone Network.

Quantization. A process that allows one or more data elements to be represented at a lower resolution for the purpose of reducing the effective bandwidth required for transmission or storage of the respective data elements.

RCELP. Relaxed Code Excited Linear Predictive Coding, the speech coding algorithm on which the EVRC is based.

Receive Objective Loudness Rating (ROLR). A measure of receive audio sensitivity. ROLR is a frequency-weighted ratio of the line voltage input signal to a reference encoder to the acoustic output of the receiver. [N8] defines the measurement of sensitivity and [N9] defines the calculation of objective loudness rating.

ROLR. See Receive Objective Loudness Rating.

TOLR. See Transmit Objective Loudness Rating.

Transmit Objective Loudness Rating (TOLR). A measure of transmit audio sensitivity. TOLR is a frequency-weighted ratio of the acoustic input signal at the transmitter to the line voltage output of the reference decoder. [N8] defines the measurement of sensitivity and [N9] defines the calculation of objective loudness rating.

WAEPL. Weighted Acoustic Echo Path Loss. A measure of the echo performance under normal conversation. [N5] defines the measurement of WAEPL.

Zero Input Response (ZIR). The filter output caused by the non-zero initial state of the filter when no input is present.

Zero State Response (ZSR). The filter output caused by an input when the initial state of the filter is zero.

1   **ZIR**.   See Zero Input Response.

2   **ZSR**.   See Zero State Response.

2   REQUIRED MULTIPLEX OPTION SUPPORT

Service Options 3, 68, 70, and 73 shall support an interface with Multiplex Option 1. Speech packets for Service Options 3, 68, 70, and 73 shall only be transported as primary traffic. This Specification is consistant with the context of [N3] and [N4].

2.1 Interface to Multiplex Option 1

2.1.1 Transmitted Packets

The speech codec shall generate and supply exactly one packet to the multiplex sublayer every 20 milliseconds. The packet contains the service option information bits, which are transmitted as primary traffic. The packet shall be one of Rate 1, Rate 1/2, and Rate 1/8 for SO 3 and SO 70, the first four types for SO 68, and the first five types for SO 73, as shown in Table 2.1.1-1. The number of bits supplied to the multiplex sublayer for each type of packet shall also be as shown in Table 2.1.1-1. Unless otherwise commanded, the speech codec may supply a Rate 1, Rate 1/2, Rate 1/4 or Rate 1/8 packet. Upon command, the speech codec shall generate a Blank packet. Also upon command, the speech codec shall generate a non-blank packet with a maximum rate of Rate 1/2.

A Blank packet contains no bits and is used for blank-and-burst transmission of signaling traffic or secondary traffic (see [N1], [N3], and [N4]).

For Service Option 73, for each generated Rate 1/8 Non-Critical packet, the multiplex sublayer transmits the Rate 1/8 Non-Critical type packet as a Rate 1/8 packet if it needs to transmit for the current frame; the multiplex sublayer suppresses the Rate 1/8 Non-Critical packet if it does not need to transmit for the current frame. ~~The system operates in non-DTX mode if the multiplex sublayer transmits all Rate 1/8 Non-Critical type packets. The system operates in DTX mode if the multiplex sublayer suppresses some or all Rate 1/8 Non-Critical type packets.~~ See Section 4.17 for the descriotion of Rate 1/8 Non-Critical packet type.

Table 2.1.1-1 Packet Types Supplied by Service Options 3, 68, 70, and 73 to the Multiplex Sublayer

Packet type	Bits per packet
Rate 1	171
Rate 1/2	80
Rate 1/4	40
Rate 1/8	16
Rate 1/8 Non-Critical	16
Blank	0

2.1.2 Received Packets

The multiplex sublayer in the receiving station categorizes every received Traffic Channel frame, and supplies the packet type and accompanying bits, if any, to the speech codec as shown in Table 2.1.2-1. The speech codec processes the bits of the packet as described in Section 5. The received packet types shown in Table 2.1.2-1 correspond to the transmitted packet types shown in Table 2.1.1-1. The Blank packet type occurs when the receiving station determines that a blank-and-burst frame for signaling traffic or secondary traffic was transmitted. The Rate 1 with bit errors packet type occurs when the receiving station determines that the frame was transmitted at 9600 bps

1   and the frame has one or more bit errors. The insufficient frame quality packet type occurs when the mobile station
2   is unable to decide upon the data rate of the received frame or when the mobile station detects a frame in error,
3   which does not belong to the Rate 1 with bit errors packet type. Although the Service Options 3 and 70 do not
4   utilize Rate 1/4 packets, Multiplex Option 1 is not required to recognize this fact; Service Options 3 and 70 are,
5   therefore, responsible for declaring Rate 1/4 frames as having insufficient frame quality (erasure). Service Options
6   68 and 73 utilize Rate 1/4 packets. For Service Option 73, each non-transmitted Rate 1/8 Non-Critical frame can be
7   treated as either a Blank or erasure packet type, and each correctly received Rate 1/8 or Rate 1/8 Non-Critical frame
8   is considered as packet type Rate 1/8 shown in Table 2.1.2-1.

9   **Table 2.1.2-1 Packet Types Supplied by the Multiplex Sublayer to Service Options 3, 68, 70 and 73**

Packet type	Bits per packet
Rate 1	171
Rate 1/2	80
Rate 1/4	40
Rate 1/8	16
Blank	0
Rate 1 with bit errors	171
Insufficient frame quality (erasure)	0

2.2 Negotiation for Service Option 3

11 The mobile station and base station can negotiate for Service Option 3 using either service option negotiation or
12 service negotiation, as described in [N3] and [N1].

2.2.1 Procedures Using Service Option Negotiation

14 The mobile station shall perform service option negotiation for Service Option 3 as described in [N3] and [N1]. The
15 base station shall perform service option negotiation for Service Option 3 as described in [N3] and [N1].

2.2.1.1 Initialization and Connection

2.2.1.1.1 Initialization and Connection in the Mobile Station

18 If the mobile station sends a *Service Option Response Order* accepting Service Option 3 in response to receiving a
19 *Service Option Request Order*, (see [N3] and [N1]), the mobile station shall initialize and connect Service Option 3
20 according to the following:

21 • If the mobile station is in the *Conversation Substate*, the mobile station shall complete the initialization and
22   connection of the transmitting and receiving sides within 200 ms of:

23   o The implicit or explicit action time associated with the *Service Option Request Order* (see [N3] and [N1]),
24   or

25   o The time that the mobile station sends the *Service Option Response Order* accepting Service Option 3,
26   whichever is later.

27 • If the mobile station is not in the *Conversation Substate*, the mobile station shall complete the initialization and
28   connection of the transmitting side within 200 ms of:

o The implicit or explicit action time associated with the *Service Option Request Order*,

o The time that the mobile station sends the *Service Option Response Order* accepting Service Option 3, or

o The time that the mobile station enters the *Conversation Substate*, whichever is later.

- If the mobile station is not in the *Conversation Substate*, the mobile station shall complete the initialization and connection of the receiving side within 200 ms of:

  o The implicit or explicit action time associated with the *Service Option Request Order*,

  o The time that the mobile station sends the *Service Option Response Order* accepting Service Option 3, or

  o If not in the *Conversation Substate*, the time that the mobile station enters the *Waiting for Answer Substate*, whichever is later.

If the mobile station receives a *Service Option Response Order* accepting its request for Service Option 3 (see [N3] and [N1]), the mobile station shall initialize and connect Service Option 3 according to the following:

- If the mobile station is in the *Conversation Substate*, the mobile station shall complete the initialization and connection of the transmitting and receiving sides within 200 ms of:

  o The implicit or explicit action time associated with the *Service Option Response Order* (see [N3] and [N1]).

- If the mobile station is not in the *Conversation Substate*, the mobile station shall complete the initialization and connection of the transmitting side within 200 ms of:

  o The implicit or explicit action time associated with the *Service Option Response Order*, or

  o The time that the mobile station enters the *Conversation Substate*, whichever is later.

- If the mobile station is not in the *Conversation Substate*, the mobile station shall complete the initialization and connection of the receiving side within 200 ms of:

  o The implicit or explicit action time associated with the *Service Option Response Order*, or

  o The time that the mobile station enters the *Waiting for Answer Substate*, whichever is later.

Service Option 3 initializations are described in Sections 4 and 5.

When the transmitting side of Service Option 3 is connected, Service Option 3 shall generate and transfer packets to the multiplex sublayer. When the receiving side is connected, Service Option 3 shall transfer and process packets from the multiplex sublayer. Refer to [N3] and [N1] when the transmitting side of a service option is not connected.

2.2.1.1.2 Initialization and Connection in the Base Station

The base station should wait until the action time associated with the most recently transmitted *Service Option Response Order* or *Service Option Request Order* before initializing and connecting Service Option 3.

If the base station accepts Service Option 3 (by sending a *Service Option Response Order* as described in [N3] and [N1], it should initialize and connect both the transmitting and receiving side of Service Option 3 before the called party is connected, so that both the base station and mobile station speech codecs can stabilize. The base station may defer connecting the land party audio to the speech codec.

If the base station receives an acceptance of its request for Service Option 3 (by receiving a *Service Option Response Order* as described in [N3] and [N1], it should initialize and connect both the transmitting and receiving side of

Service Option 3 before the called party is connected so that both the base station and mobile station speech codecs can stabilize. The base station may defer connecting the land party audio to the speech codec.

When the transmitting side of Service Option 3 is connected, Service Option 3 shall generate and transfer packets to the multiplex sublayer. When the receiving side is connected, Service Option 3 shall transfer and process packets from the multiplex sublayer. Refer to [N3] and [N1] when the transmitting side of a service option is not connected.

2.2.1.2 Service Option Control Orders

The base station may send a *Service Option Control Order* to the mobile station on the Forward Traffic Channel (see [N3] and [N1]). In addition to pending ACTION_TIMEs for messages or orders not related to the *Service Option Control Order* for Service Option 3, the mobile station shall support at least one pending ACTION_TIME for *Service Option Control Orders* for Service Option 3. The mobile station shall not send a *Service Option Control Order* for this service option.

If Service Option 3 is active, the mobile station shall treat the ORDQ field in the *Service Option Control Order* as follows:

If ORDQ equals 'xxx000x1', then the mobile station shall initialize both the transmitting and receiving sides of the speech codec as described in Section 4 and 5. The initializations shall begin at the implicit or explicit action time (see [N3] and [N1]) and shall be completed within 40 ms. In addition, if ORDQ equals 'xxx00011' then the mobile station should disable the audio output of the speech codec for 1 second after initialization.

If Service Option 3 is active and the mobile station receives a *Service Option Control Order* having an ORDQ field in which the 3 MSBs have values given in Table 2.2.1.2-1, then the mobile station shall generate the fraction of those packets normally generated as Rate 1 packets (see 4.3) at either Rate 1 or Rate 1/2 as specified by the corresponding line in the table. The mobile station shall continue to use these fractions until either of the following events occurs:

- While Service Option 3 is active, the mobile station receives a *Service Option Control Order* that specifies different fractions, or

- Service Option 3 is initialized.

Table 2.2.1.2-1 Fraction of Packets at Rate 1 and Rate 1/2 with Rate Reduction

ORDQ (binary)	Fraction of normally Rate 1 packets to be Rate 1	Fraction of normally Rate 1 packets to be Rate 1/2
000XXXXX	1	0
001XXXXX	3/4	1/4
010XXXXX	1/2	1/2
011XXXXX	1/4	3/4
100XXXXX	0	1

The mobile station may use the following procedure to perform this rate reduction: Sequences of *N* packets as are formed as shown in Table 2.2.1.2-2. The first *L* packets in this sequence are allowed to be at Rate 1, the next *N-L* packets are forced to be Rate 1/2. Whenever the rate determination process (see 4.3) selects a rate other than Rate 1, the sequence is reset. This ensures that the first packet in a talk spurt will be at Rate 1, unless ORDQ equals '100XXXXX' or the speech codec has been commanded by the multiplex sublayer to generate other than a Rate 1 packet (see 2.1.1).

1 **Table 2.2.1.2-2 Sequence Parameters for Rate Reduction**

ORDQ (binary)	Sequence length, N	Maximum number of contiguous Rate 1 packets in a Sequence, L	Number of contiguous Rate 1/2 packets in a Sequence, N-L
000XXXXX	1	1	0
001XXXXX	4	3	1
010XXXXX	2	1	1
011XXXXX	4	1	3
100XXXXX	1	0	1

2 Any other *Service Option Control Order* referring to Service Option 3 and having an ORDQ field other than those
3 described in this section shall be rejected using the *Mobile Station Reject Order* with an ORDQ field equal to
4 '00000100' (see [N3] and [N1]).

2.2.2 Procedures Using Service Negotiation

6 The mobile station and base station shall perform service negotiation for Service Option 3 as described in [N3] and
7 [N1], and the negotiated service configuration shall include only valid attributes for the service option as specified in
8 Table 2.2.2-1.

9 **Table 2.2.2-1 Valid Service Configuration Attributes for Service Option 3**

Service configuration attribute	Valid selections
Forward Multiplex Option	Multiplex Option 1
Reverse Multiplex Option	Multiplex Option 1
Forward Transmission Rates	Rate Set 1 with all rates enabled
Reverse Transmission Rates	Rate Set 1 with all rates enabled
Forward Traffic Type	Primary Traffic
Reverse Traffic Type	Primary Traffic

2.2.2.1 Initialization and Connection

Mobile Station Requirements

12 If the mobile station accepts a service configuration, as specified in a *Service Connect Message*, that includes a
13 service option connection using Service Option 3, the mobile station shall perform the following:

14 ● If the service option connection is new (that is, not part of the previous service configuration), the mobile station
15 shall perform speech codec initialization (see Sections 4 and 5) at the action time associated with the *Service*
16 *Connect Message*. The mobile station shall complete the initialization within 40 ms.

17 ● Commencing at the action time associated with the *Service Connect Message* and continuing for as long as the
18 service configuration includes the service option connection, Service Option 3 shall process received packets and
19 generate and supply packets for transmission as follows:

20 o If the mobile station is in the *Conversation Substate*, Service Option 3 shall process the received packets and
21 generate and supply packets for transmission in accordance with this standard.

o   If the mobile station is not in the *Conversation Substate*, Service Option 3 shall process the received packets in accordance with this standard, and shall generate and supply Rate 1/8 Packets with all bits set to '1' for transmission, except when commanded to generate a Blank packet.

Base Station Requirements

If the base station establishes a service configuration, as specified in a *Service Connect Message*, that includes a service option connection using Service Option 3, the base station shall perform the following:

- If the service option connection is new (that is, not part of the previous service configuration), the base station shall perform speech codec initialization (see Sections 4 and 5) no later than the action time associated with the *Service Connect Message.*

- Commencing at the action time associated with the *Service Connect Message* and continuing for as long as the service configuration includes the service option connection, Service Option 3 shall process received packets and generate and supply packets for transmission in accordance with this standard. The base station may defer enabling the audio input and output.

2.2.2.2 Service Option Control Messages

Mobile Station Requirements

The mobile station shall support one pending *Service Option Control Message* for Service Option 3.

If the mobile station receives a *Service Option Control Message* for Service Option 3, then, at the action time associated with the message, the mobile station shall process the message as follows:

1. If the MOBILE_TO_MOBILE field is equal to '1', the mobile station should disable the audio output of the speech codec for 1 second after initialization.

2. If the MOBILE_TO_MOBILE field is equal to '0', the mobile station shall process each received packet as described in Section 5.

3. If the INIT_CODEC field is equal to '1', the mobile station shall perform speech codec initialization (see Sections 4 and 5). The mobile station shall complete the initialization within 40 ms.

4. If the RATE_REDUC field is equal to a value defined in Table 2.2.2.2-3, Service Option 3 shall generate the fraction of those packets normally generated as Rate 1 packets (see 4.3) at either Rate 1 or Rate 1/2 as specified by the corresponding line in Table 2.2.2.2-3. Service Option 3 shall continue to use these fractions until either of the following events occur:

    o   The mobile station receives a *Service Option Control Message* specifying a different RATE_REDUC, or

    o   Service Option 3 is initialized.

Service Option 3 may use the following procedure to perform this rate reduction: Sequences of N packets as are formed as shown in Table 2.2.2.2-1. The first L packets in this sequence are allowed to be at Rate 1, the next $N-L$ packets are forced to be Rate 1/2. Whenever the rate determination process (see 4.3) selects a rate other than Rate 1, the sequence is reset. This ensures that the first packet in a talk spurt will be at Rate 1, unless RATE_REDUC equals '100' or the speech codec has been commanded by the multiplex sublayer to generate other than a Rate 1 packet (see 2.1.1).

Table 2.2.2.2-1 Sequence Parameters for Rate Reduction

RATE_REDUC (binary)	Sequence length, N	Maximum number of contiguous Rate 1 packets in a Sequence, L	Number of contiguous Rate 1/2 packets in a Sequence, N-L
'000'	1	1	0
'001'	4	3	1
'010'	2	1	1
'011'	4	1	3
'100'	1	0	1

If the RATE_REDUC field is not equal to a value defined in Table 2.2.2.2-1, the mobile station shall reject the message by sending a *Mobile Station Reject Order* with the ORDQ field set equal to '00000100'.

Base Station Requirements

The base station may send a *Service Option Control Message* to the mobile station. If the base station sends a *Service Option Control Message*, the base station shall include the following type-specific fields for Service Option 3:

Table 2.2.2.2-2 Service Option Control Message Type-Specific Fields

Field	Length (bits)
RATE_REDUC	3
RESERVED	3
MOBILE_TO_MOBILE	1
INIT_CODEC	1

RATE_REDUC – Rate reduction.

    The base station shall set this field to the RATE_REDUC value from Table 2.2.2.2-1 corresponding to the rate reduction that the mobile station is to perform.

RESERVED – Reserved bits.

    The base station shall set this field to '000'.

MOBILE_TO_MOBILE – Mobile-to-mobile processing.

    If the mobile station is to perform mobile-to-mobile processing (see 2.2.2.2.1), the base station shall set this field to '1'. In addition, if the mobile station is to disable the audio output of the speech codec for 1 second after initialization, the base station shall set the INIT_CODEC field and the MOBILE_TO_MOBILE field to '1'. If the mobile station is not to perform mobile-to-mobile processing, the base station shall set the MOBILE_TO_MOBILE field to '0'.

INIT_CODEC – Initialize speech codec.

    If the mobile station is to initialize the speech codec (see Sections 4 and 5), the base station shall set this field to '1'. Otherwise, the base station shall set this field to '0'.

Table 2.2.2.2-3 Fraction of Packets at Rate 1 and Rate 1/2 with Rate Reduction

RATE_REDUC (binary)	Fraction of normally Rate 1 packets to be Rate 1	Fraction of normally Rate 1 packets to be Rate 1/2
'000'	1	0
'001'	3/4	1/4

RATE_REDUC (binary)	Fraction of normally Rate 1 packets to be Rate 1	Fraction of normally Rate 1 packets to be Rate 1/2
'010'	1/2	1/2
'011'	1/4	3/4
'100'	0	1
All other RATE_REDUC values are reserved.		

2.3 Service Option Group Assignments

A service option group, see [S2], is a bit map that identifies a logically related set of service options. The bit map may contain 4, 8, or 12 bits, depending upon the number of logically related service options in the group. Each bit in the bit map corresponds to one service option.

A mobile station or a base station indicates that it supports a service option in the group by setting the bit corresponding to the service option to '1'. A mobile station or base station can also indicate that it does not support a service option by setting the bit to '0'. All reserved bits are set to '0'.

Table 2.3-1 shows the service option group for SO 68. The most significant bit of the service option group map is on the left, and the least significant bit is on the right.

Table 2.3-1 Service Option Group Assignments

Service option group (decimal)	Type of services in the group	Service option group bit map and associated service option assignments
0	Voice Services	- - - x - - - - SO 68

2.4 Negotiation for Service Option 68

The mobile station and base station can negotiate for Service Option 68 service negotiation, as described in [N1].

2.4.1 Procedures Using Service Negotiation

The mobile station and base station shall perform service negotiation for Service Option 68 as described in [N1], and the negotiated service configuration shall include only valid attributes for the service option as specified in Table 2.4.1-1.

Table 2.4.1-1 Valid Service Configuration Attributes for Service Option 68

Service configuration attribute	Valid selections
Forward Multiplex Option	Multiplex Option 1
Reverse Multiplex Option	Multiplex Option 1
Forward Transmission Rates	Rate Set 1 with all rates enabled
Reverse Transmission Rates	Rate Set 1 with all rates enabled
Forward Traffic Type	Primary Traffic
Reverse Traffic Type	Primary Traffic

2.4.1.1 Initialization and Connection

Mobile Station Requirements

If the mobile station accepts a service configuration, as specified in a *Service Connect Message, General Handoff Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service Option 68, the mobile station shall perform the following:

- If the service option connection is new, i.e., not part of the previous service configuration, the mobile station shall perform speech codec initialization at the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated. The mobile station shall initialize its EVRC-B encoder mode of operation to a default value of 0. The mobile station shall complete the initialization within 40 ms.

- Beginning at the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated, and continuing for as long as the service configuration includes the service option connection, Service Option 68 shall process received packets and generate and supply packets for transmission as follows:

  o If the *Call Control Instance* is in the *Conversation Substate*, Service Option 68 shall process the received packets and generate and supply packets for transmission in accordance with this standard.

  o If the *Call Control Instance* is not in the *Conversation Substate*, Service Option 68 shall process the received packets in accordance with this standard, and shall generate and supply Rate 1/8 packets with all bits set to 1 for transmission, except when commanded to generate a blank packet.

Base Station Requirements

If the base station establishes a service configuration, as specified in a *Service Connect Message*, *General Handoff Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service Option 68, the base station shall perform the following:

- If the service option connection is new, i.e., not part of the previous service configuration, the base station shall perform speech codec initialization no later than the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated. The base station shall initialize its EVRC-B encoder mode of operation to a default value of 0.

- Commencing at the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated, and continuing for as long as the service configuration includes the service option connection, Service Option 68 shall process received packets, and shall generate and supply packets for transmission in accordance with this standard. The base station may defer enabling the audio input and output.

2.4.1.2 Service Option Control Messages

Mobile Station Requirements

The mobile station shall support one pending *Service Option Control Message* for Service Option 68.

If the mobile station receives a *Service Option Control Message* for Service Option 68, then, at the action time associated with the message, the mobile station shall process the message as follows:

If the mobile station has an application which requires the reverse link to operate at capacity operating point 0, the mobile station may reject the message by sending a mobile station Reject Order with an ORDQ of 00000100 while that application is active.

If the mobile station accepts the *Service Option Control Message* the following occurs:

1. If the MOBILE_TO_MOBILE field is equal to 1, the mobile station should disable the audio output of the speech codec for 1 second after initialization.

2. If the MOBILE_TO_MOBILE field is equal to 0, the mobile station shall process each received packet as described in Section 5.

3. If the INIT_CODEC field is equal to 1, the mobile station shall perform speech codec initialization. The mobile station shall complete the initialization within 40 ms.

4. EVRC-B accepts as input, an encoder capacity operating point through the RATE_REDUC field as defined in Table 2.4.1.2-1Table 2.4.1.2-2. Using this capacity operating point, EVRC-B generates Rate 1, Rate 1/2, and Rate 1/4 packets in a proportion that results in the average data rate given by Table 2.4.1.2-1.

5. Service Option 68 shall continue to use this proportion until either of the following events occur:

   a. The mobile station receives a *Service Option Control Message* specifying a different RATE_REDUC, or

   b. Service Option 68 is initialized.

Service Option 68 was developed using reduced rate operation (encoder capacity operating point) as a network control criteria. Unlike the rate reduction mechanism in EVRC-A, Service Option 68 does not use a deterministic pattern of Rate 1 and Rate 1/2 to reduce the average encoding rate. Instead, the EVRC-B codec selects the encoding rate based upon the characteristics of the input speech; voiced unvoiced, transient, stationary, as well as the encoding mode selected.

Table 2.4.1.2-1 Service Option 68 Encoding Rate Control Parameters

RATE_REDUC (binary)	Encoder capacity operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)
'000'	0	9.3	8.3 kbps
'001'	1	8.5	7.57 kbps
'010'	2	7.5	6.64 kbps
'011'	3	7.0	6.18 kbps
'100'	4	6.6	5.82 kbps
'101'	5	6.2	5.45 kbps
'110'	6	5.8	5.08 kbps
'111'	7 (Rate 1/2 max)	4.8	4.0 kbps

Quarter-Rates are not used when RATE_REDUC = '000' and RATE_REDUC = '111'. Therefore, RATE_REDUC = '000' and RATE_REDUC = '111' could be used in [N3] and [N10] systems where Quarter-Rate frames are sometimes disallowed.

The mobile station may send a *Service Option Control Message* for Service Option 68 to the BS ordering the BS to utilize capacity operating point 0 in the forward direction under the following condition:

1. The mobile has an application for which the BS needs to use capacity operating point 0 in the forward direction, and

2. The mobile has successfully received a Service Connect Message or has successfully completed a hard handoff.

If the mobile station sends a *Service Option Control Message*, the mobile station shall include the following type-specific fields for Service Option 68.

Table 2.4.1.2-2 Service Option Control Message Type-Specific Fields

Field	Length (bits)
RATE_REDUC	3
RESERVED	5

RATE_REDUC – Service Option 68 mode of operation; the mobile station shall set this field to value '000', corresponding to the mode 0, in which the BS is to operate on the forward link.

RESERVED – reserved bits; the mobile station shall set this field to 00000

Base Station Requirements

The base station may send a *Service Option Control Message* to the mobile station. If the base station sends a *Service Option Control Message*, the base station shall include the following type-specific fields for Service Option 68.

Table 2.4.1.2-3 Service Option Control Message Type-Specific Fields

Field	Length (bits)
RATE_REDUC	3
RESERVED	3
MOBILE_TO_MOBILE	1
INIT_CODEC	1

RATE_REDUC – Service Option 68 encoder capacity operating point; the base station shall set this field to the value from Table 2.4.1.2-1Table 2.4.1.2-4 corresponding to the capacity operating point in which the mobile station is to operate

RESERVED – reserved bits; the base station shall set this field to 000

MOBILE_TO_MOBILE – mobile-to-mobile processing; if the mobile station is to perform mobile-to-mobile processing, the base station shall set this field to 1; in addition, if the mobile station is to disable the audio output of the speech codec for 1 second after initialization, the base station shall set the INIT_CODEC field and the MOBILE_TO_MOBILE field to 1; if the mobile station is not to perform mobile-to-mobile processing, the base station shall set the MOBILE_TO_MOBILE field to 0

INIT_CODEC – initialize speech codec; if the mobile station is to initialize the speech codec, the base station shall set this field to 1; otherwise, the base station shall set this field to 0

Table 2.4.1.2-4 Service Option 68 Encoding Rate Control Parameters

RATE_REDUC (binary)	Encoder operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)
'000'	0	9.3	8.3 kbps
'001'	1	8.5	7.57 kbps
'010'	2	7.5	6.64 kbps
'011'	3	7.0	6.18 kbps
'100'	4	6.6	5.82 kbps
'101'	5	6.2	5.45 kbps
'110'	6	5.8	5.08 kbps
'111'	7 (Rate 1/2 max)	4.8	4.0 kbps

Quarter-Rates are not used when RATE_REDUC = '000' and RATE_REDUC = '111'. Therefore, RATE_REDUC = '000' and RATE_REDUC = '111' could be used in [N3] and [N10] systems where Quarter-Rate frames are sometimes disallowed.

The base station should support one pending *Service Option Control Message* for Service Option 68.

If the base station receives a *Service Option Control Message* for Service Option 68, then, at the action time associated with the message, the base station should process the message as follows:

The base station shall use mode 0 and begin transmission to the mobile station in the forward direction using this mode.

The base station shall continue to use this mode for Service Option 68 with the mobile station for the duration of the call.

2.5 Negotiation for Service Option 70

The mobile station and base station can negotiate for Service Option 70 service negotiation, as described in [N1].

2.5.1 Procedures Using Service Negotiation

The mobile station and base station shall perform service negotiation for Service Option 70 as described in [N1], and the negotiated service configuration shall include only valid attributes for the service option as specified in Table 2.5.1-1.

Table 2.5.1-1 Valid Service Configuration Attributes for Service Option 70

Service configuration attribute	Valid selections
Forward Multiplex Option	Multiplex Option 1
Reverse Multiplex Option	Multiplex Option 1
Forward Transmission Rates	Rate Set 1 with all rates enabled
Reverse Transmission Rates	Rate Set 1 with all rates enabled
Forward Traffic Type	Primary Traffic
Reverse Traffic Type	Primary Traffic

2.5.1.1 Initialization and Connection

Mobile Station Requirements

If the mobile station accepts a service configuration, as specified in a *Service Connect Message, General Handoff Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service Option 70, the mobile station shall perform the following:

- If the service option connection is new, i.e., not part of the previous service configuration, the mobile station shall perform speech codec initialization at the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated. The mobile station shall initialize its EVRC-WB encoder mode of operation to a default value of 0. The mobile station shall complete the initialization within 40 ms.

- Beginning at the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated, and continuing for as long as the service configuration includes the service option connection, Service Option 70 shall process received packets and generate and supply packets for transmission as follows:

  - If the *Call Control Instance* is in the *Conversation Substate*, Service Option 70 shall process the received packets and generate and supply packets for transmission in accordance with this standard.

  - If the *Call Control Instance* is not in the *Conversation Substate,* Service Option 70 shall process the received packets in accordance with this standard, and shall generate and supply Rate 1/8 packets with all bits set to 1 for transmission, except when commanded to generate a blank packet.

Base Station Requirements

If the base station establishes a service configuration, as specified in a *Service Connect Message*, *General Handoff Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service Option 70, the base station shall perform the following:

- If the service option connection is new, i.e., not part of the previous service configuration, the base station shall perform speech codec initialization no later than the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated. The base station shall initialize its EVRC-WB encoder mode of operation to a default value of 0.

- Commencing at the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated, and continuing for as long as the service configuration includes the service option connection, Service Option 70 shall process received packets, and shall generate and supply packets for transmission in accordance with this standard. The base station may defer enabling the audio input and output.

2.5.1.2 Service Option Control Messages

Mobile Station Requirements

The mobile station shall support one pending *Service Option Control Message* for Service Option 70.

If the mobile station receives a *Service Option Control Message* for Service Option 70, then, at the action time associated with the message, the mobile station shall process the message as follows:

If the mobile station has an application which requires the reverse link to operate at capacity operating point 04, the mobile station may switch to or continues operate at capacity operating point 4 and reject the message by sending an mobile station Reject Order with an ORDQ of 00000100 while that application is active.

If the mobile station accepts the *Service Option Control Message* the following occurs:

1. If the MOBILE_TO_MOBILE field is equal to 1, the mobile station should disable the audio output of the speech codec for 1 second after initialization.

2. If the MOBILE_TO_MOBILE field is equal to 0, the mobile station shall process each received packet as described in Section 5.

3. If the INIT_CODEC field is equal to 1, the mobile station shall perform speech codec initialization. The mobile station shall complete the initialization within 40 ms.

4. Service Option 70 accepts as input, an encoder capacity operating point through the RATE_REDUC field as defined in Table 2.5.1.2-1Table 2.5.1.2-2. Using this capacity operating point, Service Option 70 generates Rate 1 and Rate 1/2 packets in a proportion that results in the average data rate given by Table 2.5.1.2-1.

5. Service Option 70 shall continue to use this proportion until either of the following events occur:

   a. The mobile station receives a *Service Option Control Message* specifying a different RATE_REDUC, or

   b. Service Option 70 is initialized.

Table 2.5.1.2-1 Service Option 70 Encoding Rate Control Parameters

RATE_REDUC (binary)	Encoder capacity operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)	Description of the capacity operating point
'000'	0	8.5 kbps	7.5 kbps	This is a wideband capacity operating point. The bit-stream carries information about both Lowband and Highband of the 16 kHz signal.
'001'				RESERVED
'010'				RESERVED
'011'				RESERVED
'100'	4	9.3 kbps	8.3 kbps	This is a narrowband capacity operating point that is interoperable with Capacity Operating Point 0 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Half-Rate CELP, and Eighth-Rate used in SO 68.
'101'				RESERVED
'110'				RESERVED

RATE_REDUC (binary)	Encoder capacity operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)	Description of the capacity operating point
'111'	7 (Rate 1/2 max)	4.8 kbps	4.0 kbps	This is a narrowband capacity operating point, using a maximum of Rate 1/2. This mode is interoperable with Capacity Operating Point 7 of SO 68 because the bit-stream carries information about a narrowband signal using the Half-Rate CELP, and Eighth-Rate used in SO 68..

1

2 The mobile station may send a *Service Option Control Message* for Service Option 70 to the base station ordering
3 the base station to utilize capacity operating point 0 4 in the forward direction under the following condition:

4 1. The mobile has an application for which the base station needs to use capacity operating point 0 4 in the
5 forward direction, and

6 2. The mobile has successfully received a *Service Connect Message* or has successfully completed a hard handoff.

7 If the mobile station sends a *Service Option Control Message*, the mobile station shall include the following type-
8 specific fields for Service Option 70.

9 **Table 2.5.1.2-2 Service Option Control Message Type-Specific Fields**

Field	Length (bits)
RATE_REDUC	3
RESERVED	5

10

11 RATE_REDUC – Service Option 70 mode of operation; the mobile station shall set this field to
12 value '000' '100', corresponding to the mode 0 4, in which the base station is to
13 operate on the forward link.
14 RESERVED – reserved bits; the mobile station shall set this field to 00000

15 **Base Station Requirements**

16 The base station may send a *Service Option Control Message* to the mobile station. If the base station sends a
17 *Service Option Control Message*, the base station shall include the following type-specific fields for Service Option
18 70.

19 **Table 2.5.1.2-3 Service Option Control Message Type-Specific Fields**

Field	Length (bits)
RATE_REDUC	3
RESERVED	3
MOBILE_TO_MOBILE	1
INIT_CODEC	1

20

RATE_REDUC – Service Option 70 encoder capacity operating point; the base station shall set this field to the value from Table 2.5.1.2-1~~Table 2.5.1.2-4~~ corresponding to the capacity operating point in which the mobile station is to operate

RESERVED – reserved bits; the base station shall set this field to 000

MOBILE_TO_MOBILE – mobile-to-mobile processing; if the mobile station is to perform mobile-to-mobile processing, the base station shall set this field to 1; in addition, if the mobile station is to disable the audio output of the speech codec for 1 second after initialization, the base station shall set the INIT_CODEC field and the MOBILE_TO_MOBILE field to 1; if the mobile station is not to perform mobile-to-mobile processing, the base station shall set the MOBILE_TO_MOBILE field to 0

INIT_CODEC – initialize speech codec; if the mobile station is to initialize the speech codec, the base station shall set this field to 1; otherwise, the base station shall set this field to 0

Table 2.5.1.2-4 Service Option 70 Encoding Rate Control Parameters

RATE_REDUC (binary)	Encoder operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)	Description of the operating point
'000'	0	8.5 kbps	7.5 kbps	This is a wideband operating point. The bit-stream carries information about both Lowband and Highband of the 16 kHz signal.
'001'				RESERVED
'010'				RESERVED
'011'				RESERVED
'100'	4	9.3 kbps	8.3 kbps	This is a narrowband operating point that is interoperable with Operating Point 0 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Half-Rate CELP, and Eighth-Rate used in SO 68.
'101'				RESERVED
'110'				RESERVED
'111'	7 (Rate 1/2 max)	4.8 kbps	4.0 kbps	This is a narrowband operating point, using a maximum of Rate 1/2. This mode is interoperable with Operating Point 7 of SO 68 because the bit-stream carries information about a narrowband signal using the Half-Rate CELP, and Eighth-Rate used in SO 68..

The base station should support one pending *Service Option Control Message* for Service Option 70.

If the base station receives a *Service Option Control Message* for Service Option 70, then, at the action time associated with the message, the base station should process the message as follows:

1 The base station shall use mode 0 4 and begin transmission to the mobile station in the forward direction using
2 this mode.

3 The base station shall continue to use this mode for Service Option 70 with the mobile station for the duration
4 of the call.

2.6 Negotiation for Service Option 73

6 The mobile station and base station can negotiate for Service Option 73 service negotiation, as described in [N1].
7 The support of the encoding and decoding of capacity operating point 0 is optional.

2.6.1 Procedures Using Service Negotiation

9 The mobile station and base station shall perform service negotiation for Service Option 73 as described in [N1], and
10 the negotiated service configuration shall include only valid attributes for the service option as specified in Table
11 2.6.1-1.

12 **Table 2.6.1-1 Valid Service Configuration Attributes for Service Option 73**

Service configuration attribute	Valid selections
Forward Multiplex Option	Multiplex Option 1
Reverse Multiplex Option	Multiplex Option 1
Forward Transmission Rates	Rate Set 1 with all rates enabled
Reverse Transmission Rates	Rate Set 1 with all rates enabled
Forward Traffic Type	Primary Traffic
Reverse Traffic Type	Primary Traffic

2.6.1.1 Initialization and Connection

14 **Mobile Station Requirements**

15 If the mobile station accepts a service configuration, as specified in a *Service Connect Message*, *General Handoff*
16 *Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service
17 Option 73, the mobile station shall perform the following:

18 • If the service option connection is new, i.e., not part of the previous service configuration, the mobile station
19 shall perform speech codec initialization at the time specified by the maximum of the action time associated with
20 the message carrying the service configuration record, and the time that the corresponding *Call Control Instance*
21 is instantiated. The mobile station shall initialize its EVRC-NW encoder mode of operation to a default value of
22 0 4. The mobile station shall complete the initialization within 40 ms.

23 • Beginning at the time specified by the maximum of the action time associated with the message carrying the
24 service configuration record, and the time that the corresponding *Call Control Instance* is instantiated, and
25 continuing for as long as the service configuration includes the service option connection, Service Option 73
26 shall process received packets and generate and supply packets for transmission as follows:

27 o If the *Call Control Instance* is in the *Conversation Substate*, Service Option 73 shall process the received
28 packets and generate and supply packets for transmission in accordance with this standard.

o If the *Call Control Instance* is not in the *Conversation Substate,* Service Option 73 shall process the received packets in accordance with this standard, and shall generate and supply Rate 1/8 packets with all bits set to 1 for transmission, except when commanded to generate a blank packet.

Base Station Requirements

If the base station establishes a service configuration, as specified in a *Service Connect Message*, *General Handoff Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service Option 73, the base station shall perform the following:

- If the service option connection is new, i.e., not part of the previous service configuration, the base station shall perform speech codec initialization no later than the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated. The base station shall initialize its EVRC-NW encoder mode of operation to a default value of 0̶4.

- Commencing at the time specified by the maximum of the action time associated with the message carrying the service configuration record, and the time that the corresponding *Call Control Instance* is instantiated, and continuing for as long as the service configuration includes the service option connection, Service Option 73 shall process received packets, and shall generate and supply packets for transmission in accordance with this standard. The base station may defer enabling the audio input and output.

2.6.1.2 Service Option Control Messages

Mobile Station Requirements

The mobile station shall support one pending *Service Option Control Message* for Service Option 73.

If the mobile station receives a *Service Option Control Message* for Service Option 73, then, at the action time associated with the message, the mobile station shall process the message as follows:

If the mobile station receives a *Service Option Control Message* with RATE_REDUC field set to a value of '000' and if the mobile station does not support capacity operating point '0' of Table 2.6.1.2-1, the mobile station shall send a Reject Order with an ORDQ of 00000100.

If the mobile station has an application which requires the reverse link to operate at capacity operating point 0̶1, the mobile station may switch to or continues operate at capacity operating point 1 and reject the message by sending an mobile station Reject Order with an ORDQ of 00000100 while that application is active.

If the mobile station accepts the *Service Option Control Message* the following occurs:

1. If the MOBILE_TO_MOBILE field is equal to 1, the mobile station should disable the audio output of the speech codec for 1 second after initialization.

2. If the MOBILE_TO_MOBILE field is equal to 0, the mobile station shall process each received packet as described in Section 5.

3. If the INIT_CODEC field is equal to 1, the mobile station shall perform speech codec initialization. The mobile station shall complete the initialization within 40 ms.

4. Service Option 73 accepts as input, an encoder capacity operating point through the RATE_REDUC field as defined in Table 2.6.1.2-1T̶a̶b̶l̶e̶ ̶2̶.̶6̶.̶1̶.̶2̶-̶2̶. Using this capacity operating point, Service Option 73 generates Rate 1, Rate 1/2, and Rate 1/4 packets in a proportion that results in the average data rate given by Table 2.6.1.2-1.

1 5. Service Option 73 shall continue to use this proportion until either of the following events occur:

2 a. The mobile station receives a *Service Option Control Message* specifying a different RATE_REDUC, or

3 b. Service Option 73 is initialized.

4

5 **Table 2.6.1.2-1 Service Option 73 Encoding Rate Control Parameters**

RATE_REDUC (binary)	Encoder capacity operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)	Description of the capacity operating point
'000'	0	8.5 kbps	7.5 kbps	This is a wideband capacity operating point. The bit-stream carries information about both Lowband and Highband of the 16 kHz signal. This is Capacity Operating Point 0 of SO 70.
'001'	1	9.3 kbps	8.3 kbps	This is a narrowband capacity operating point that is interoperable with Capacity Operating Point 0 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Half-Rate CELP, and Eighth-Rate used in SO 68. This is Capacity Operating Point 4 of SO 70.
'010'	2	7.5	6.64 kbps	This is a narrowband capacity operating point that is interoperable with Capacity Operating Point 2 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-RateFull-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'011'	3	7.0	6.18 kbps	This is a narrowband capacity operating point that is interoperable with Capacity Operating Point 3 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'100'	4	6.6	5.82 kbps	This is a narrowband capacity operating point that is interoperable with Capacity Operating Point 4 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'101'	5	6.2	5.45 kbps	This is a narrowband capacity operating point that is interoperable with Capacity Operating Point 5 of SO 68 because the bit-stream carries information about a

RATE_REDUC (binary)	Encoder capacity operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)	Description of the capacity operating point
				narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'110'	6	5.8	5.08 kbps	This is a narrowband capacity operating point that is interoperable with Capacity Operating Point 6 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'111'	7 (Rate 1/2 max)	4.8 kbps	4.0 kbps	This is a narrowband capacity operating point, using a maximum of Rate 1/2. This mode is interoperable with Capacity Operating Point 7 of SO 68 because the bit-stream carries information about a narrowband signal using the Half-Rate CELP and Eighth-Rate used in SO 68. This is also Capacity Operating Point 7 of SO 70.

The mobile station may send a *Service Option Control Message* for Service Option 73 to the base station ordering the base station to utilize capacity operating point ~~0~~ 1 in the forward direction under the following condition:

1. The mobile has an application for which the base station needs to use capacity operating point ~~0~~ 1 in the forward direction, and

2. The mobile has successfully received a *Service Connect Message* or has successfully completed a hard handoff.

If the mobile station sends a *Service Option Control Message*, the mobile station shall include the following type-specific fields for Service Option 73.

Table 2.6.1.2-2 Service Option Control Message Type-Specific Fields

Field	Length (bits)
RATE_REDUC	3
RESERVED	5

RATE_REDUC – Service Option 73 mode of operation; the mobile station shall set this field to value ~~'000'~~'001', corresponding to the mode ~~0~~1, in which the base station is to operate on the forward link.

RESERVED – reserved bits; the mobile station shall set this field to 00000

Base Station Requirements

The base station may send a *Service Option Control Message* to the mobile station. If the base station sends a *Service Option Control Message*, the base station shall include the following type-specific fields for Service Option 73.

Table 2.6.1.2-3 Service Option Control Message Type-Specific Fields

Field	Length (bits)
RATE_REDUC	3
RESERVED	3
MOBILE_TO_MOBILE	1
INIT_CODEC	1

RATE_REDUC – Service Option 73 encoder capacity operating point; the base station shall set this field to the value from Table 2.6.1.2-1~~Table 2.6.1.2-4~~ corresponding to the capacity operating point in which the mobile station is to operate

RESERVED – reserved bits; the base station shall set this field to 000

MOBILE_TO_MOBILE – mobile-to-mobile processing; if the mobile station is to perform mobile-to-mobile processing, the base station shall set this field to 1; in addition, if the mobile station is to disable the audio output of the speech codec for 1 second after initialization, the base station shall set the INIT_CODEC field and the MOBILE_TO_MOBILE field to 1; if the mobile station is not to perform mobile-to-mobile processing, the base station shall set the MOBILE_TO_MOBILE field to 0

INIT_CODEC – initialize speech codec; if the mobile station is to initialize the speech codec, the base station shall set this field to 1; otherwise, the base station shall set this field to 0

~~Table 2.6.1.2-4 Service Option 73 Encoding Rate Control Parameters~~

~~RATE_REDUC (binary)~~	~~Encoder operating point~~	~~Estimated average encoding rate for active speech (channel encoding rates)~~	~~Estimated average encoding rate for active speech (source encoding rates)~~	~~Description of the operating point~~
~~'000'~~	~~0~~	~~8.5 kbps~~	~~7.5 kbps~~	~~This is a wideband operating point. The bit-stream carries information about both Lowband and Highband of the 16 kHz signal. This is Operating Point 0 of SO 70.~~
~~'001'~~	~~1~~	~~9.3 kbps~~	~~8.3 kbps~~	~~This is a narrowband operating point that is interoperable with Operating Point 0 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Half-Rate CELP, and Eighth-Rate used in SO 68.~~
~~'010'~~	~~2~~	~~7.5~~	~~6.64 kbps~~	~~This is a narrowband operating point that is interoperable with Operating Point 2 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.~~
~~'011'~~	~~3~~	~~7.0~~	~~6.18 kbps~~	~~This is a narrowband operating point that is interoperable with Operating Point 3 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-~~

RATE_REDUC (binary)	Encoder operating point	Estimated average encoding rate for active speech (channel encoding rates)	Estimated average encoding rate for active speech (source encoding rates)	Description of the operating point
				Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'100'	4	6.6	5.82 kbps	This is a narrowband operating point that is interoperable with Operating Point 4 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'101'	5	6.2	5.45 kbps	This is a narrowband operating point that is interoperable with Operating Point 5 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'110'	6	5.8	5.08 kbps	This is a narrowband operating point that is interoperable with Operating Point 6 of SO 68 because the bit-stream carries information about a narrowband signal using the Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate used in SO 68.
'111'	7 (Rate 1/2 max)	4.8 kbps	4.0 kbps	This is a narrowband operating point, using a maximum of Rate 1/2. This mode is interoperable with Operating Point 7 of SO 68 because the bit-stream carries information about a narrowband signal using the Half-Rate CELP and Eighth-Rate used in SO 68. This is also Operating Point 7 of SO 70.

The base station should ensure that a mobile station supports capacity operating point 0 before it enables wideband speech service by sending a *Service Option Control Message with* RATE_REDUC field set to '000' to the mobile station.[1]

The base station should support one pending *Service Option Control Message* for Service Option 73.

If the base station receives a *Service Option Control Message* for Service Option 73, then, at the action time associated with the message, the base station should process the message as follows:

> The base station shall use mode 0 1 and begin transmission to the mobile station in the forward direction using this mode.

---

[1] This guarantees that both endpoints operate in capacity operating point 0 in the use case of Transcoder Free Operation (TrFO) and/or Tandem Free Operation.

1   The base station shall continue to use this mode for Service Option 73 with the mobile station for the duration

2   of the call.

1

2

3

4

5

6   This page intentionally left blank.

3 AUDIO INTERFACES

3.1 Input Audio Interface

3.1.1 Input Audio Interface in the Mobile Station

The input audio may be either an analog or digital signal.

3.1.1.1 Conversion and Scaling

Whether the input is analog or digital, the signal presented to the input of the speech codec shall be sampled at a rate of 8000 samples per second and shall be quantized to a uniform PCM format with at least 13 bits of dynamic range. For SO 70 and SO 73, the input signal shall be sampled at either 8000 samples per second or 16000 samples per second.

The quantities in this standard assume a 16-bit integer input normalization with a range from -32,768 through +32,767. The following speech codec discussion assumes this 16-bit integer normalization. If an input audio interface uses a different normalization scheme, then appropriate scaling should be used.

3.1.1.2 Digital Audio Input

If the input audio is an 8-bit μlaw PCM signal, it shall be converted to a uniform PCM format according to Table 2 in [N6].

3.1.1.3 Analog Audio Input

If the input is in analog form, the mobile station shall sample the analog speech and shall convert the samples to a digital format for speech codec processing. This shall be done by either the following or an equivalent method. First, the input gain audio level is adjusted. Then, the signal is bandpass filtered to prevent aliasing. Finally, the filtered signal is sampled and quantized (see 3.1.1.1).

3.1.1.3.1 Transmit Level Adjustment

The mobile station shall have a transmit objective loudness rating (TOLR) equal to -46 dB, when transmitting to a reference base station. The loudness ratings are described in [N9].

3.1.1.3.2 Band Pass Filtering

Input anti-aliasing filtering shall conform to [N7]. Additional anti-aliasing filtering may be provided by the manufacturer.

For SO 70 and SO 73, if the input speech is wideband, the signal should be filtered by a filter that generally conforms to the the mask shown in [N12] or [N13].

3.1.1.3.3 Echo Return Loss

Provision shall be made to ensure adequate isolation between receive and transmit audio paths in all modes of operation. When no external transmit audio is present, the speech codec shall not generate packets at rates higher

than Rate 1/8 due to acoustic coupling of the receive audio into the transmit audio path (specifically with the receive audio at full volume). Target levels of 45 dB WAEPL should be met. See [N5].

3.1.2 Input Audio Interface in the Base Station

3.1.2.1 Sampling and Format Conversion

The base station converts the input speech (analog, μlaw companded Pulse Code Modulation, or other format) into a uniform quantized PCM format with at least 13 bits of dynamic range. The sampling rate is 8000 samples per second. The input signal for SO 70 and SO 73 may use a sampling rate of 16000 samples per second. The sampling and conversion process shall be as in 3.1.1.1.

3.1.2.2 Transmit Level Adjust

The base station shall set the transmit level so that a 1004 Hz tone at a level of 0 dBm0 at the network interface produces a level 3.17 dB below maximum amplitude at the output of the quantizer.

3.1.2.3 Line Echo Canceling

The base station shall provide a method to cancel echoes returned by the PSTN interface.[†] The echo canceling function should provide at least 30 dB of echo return loss enhancement. The echo canceling function should work over a range of PSTN echo return delays from 0 to 48 ms; however, the latter requirement is subject to local PSTN configuration demands, i.e., a 64 ms (or greater) echo canceling capability may be required in cases where the PSTN does not provide echo cancellation for long distance service. See [S3].

3.2 Output Audio Interface

3.2.1 Output Audio Interface in the Mobile Station

3.2.1.1 Band Pass Filtering

Output reconstruction filtering shall conform to [N7]. Additional reconstruction filtering may be provided by the manufacturer.

For SO 70 and SO 73, if the output speech is wideband, the signal should be filtered by a filter that generally conforms to the mask shown in [N12] or [N13].

3.2.1.2 Receive Level Adjustment

The mobile station shall have a nominal receive objective loudness rating (ROLR) equal to 51 dB when receiving from a reference base station. The loudness ratings are described in [N9].

3.2.2 Output Audio Interface in the Base Station

Details of the digital and analog interfaces to the network are outside the scope of this document.

[†] Because of the relatively long delays inherent in the speech coding and transmitting processes, echoes that are not sufficiently suppressed are noticeable to the mobile station user.

1   ## 3.2.2.1 Receive Level Adjustment

2   The base station shall set the audio level so that a received 1004 Hz tone 3.17 dB below maximum amplitude
3   produces a level of 0 dBm0 at the network interface.

1

2

3

4

5

6
<div align="center">This page intentionally left blank.</div>

4 OVERVIEW AND SPEECH ENCODER

4.1 Service Option 3

The Enhanced Variable Rate Codec (EVRC-A) is based upon the RCELP algorithm, appropriately modified for variable rate operation and for robustness in the CDMA environment. RCELP is a generalization of the Code-Excited Linear Prediction (CELP) algorithm. Unlike conventional CELP encoders, RCELP does not attempt to match the original speech signal exactly. Instead of attempting to match the original residual signal, RCELP matches a time-warped version of the original residual that conforms to a simplified pitch contour. The pitch contour is obtained by estimating the pitch delay once in each frame and linearly interpolating the pitch from frame to frame. One benefit of using this simplified pitch representation is that more bits are available in each packet for the stochastic excitation and for channel impairment protection than would be if a traditional fractional pitch approach were used. This results in enhanced error performance without impacting perceived speech quality in clear channel conditions.

The encoder uses 3 of the 4 primary traffic packet types permitted by [N3] and [N4] Multiplex Option 1: Rate 1 (171 bits/packet), Rate 1/2 (80 bits/packet), and Rate 1/8 (16 bits/packet). Upon command, the encoder will produce a blank packet (which contains no bits) or other than a Rate 1 packet (i.e., Rate 1/2 maximum); otherwise, the encoder makes its own determination about what type of packet to generate. Bit allocations for each packet type are given in Table 4.1-1.

Table 4.1-1 Bit Allocations by Packet Type

Field (see 4.8)	Packet type			
	Rate 1	**Rate 1/2**	**Rate 1/8**	**Blank**
Spectral Transition Indicator	1			
LSP	28	22	8	
Pitch Delay	7	7		
Delta Delay	5			
ACB Gain	9	9		
FCB Shape	105	30		
FCB Gain	15	12		
Frame Energy			8	
(reserved)	1			
Total	171	80	16	0

The algorithms in the sections that follow are presented in terms of block diagrams, mathematical equations where appropriate, and in pseudo-code where equations would be cumbersome. Inputs, outputs, and processing are defined for each processing block or module. Internal variables for each module are defined when used. Modules are decomposed into sub-modules at increasing levels of detail until a defining equation or block of pseudo-code describing the lowest level of processing is reached. The algorithm descriptions are designed to maximize clarity, not to illustrate the most efficient implementation. For example, no attempt is made to illustrate memory conserving techniques such as reuse of scratch variables or buffers. Operations such as Discrete Fourier Transforms (DFTs) or Finite Impulse Response (FIR)

1 filters are described in terms of a defining equation rather than in terms of one of the fast algorithms available to carry

2 them out.

3 Figure 4.1-1 is a high-level view of the EVRC-A speech encoder showing all major modules. Inputs to the encoder are

4 the speech signal vector, $\{s(n)\}$, and an external rate command signal. The external rate command may direct the

5 encoder to produce a blank packet or a packet other than Rate 1. If an external rate command is received, it will

6 supersede the encoder's internal rate selection mechanism.
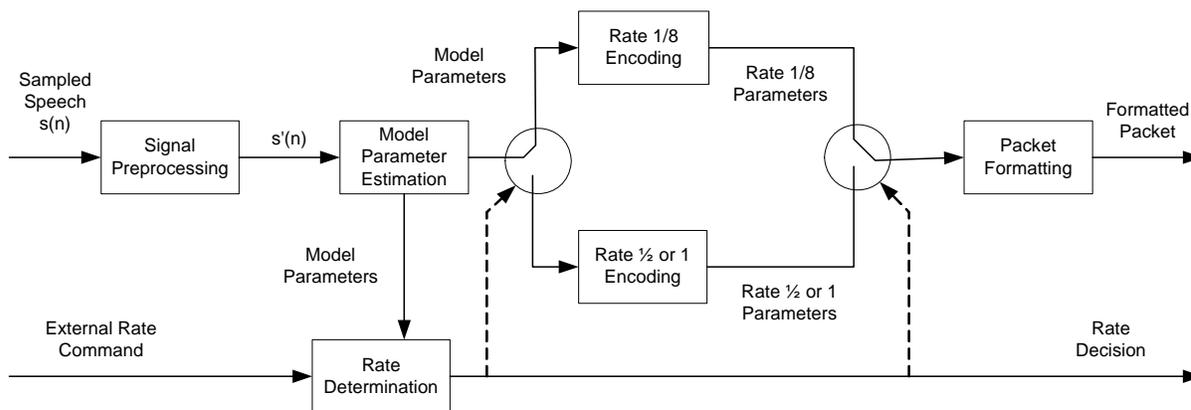
7

8 **Figure 4.1-1 Speech Encoder Top-Level Diagram**

9 The input speech vector, $\{s(n)\}$, is presented to the signal pre-processing module (see 4.1), which performs high-pass

10 and adaptive noise suppression filtering. The pre-processed speech vector, $\{s'(n)\}$, is then presented to the model

11 parameter estimation module. The model parameter estimation module (see 4.2) performs LPC analysis to determine a

12 set of linear prediction coefficients (LPCs) and the optimal pitch delay (τ). It also converts the LPCs to line spectral

13 pairs (LSPs) and calculates the long and short-term prediction gains. The rate determination module (see 4.3) applies a

14 voice activity detection (VAD) algorithm and rate selection logic in order to determine the type of packet to generate.

15 After estimating the model parameters, the encoder will characterize the excitation signal and quantize parameters in a

16 way appropriate to the selected rate (see 4.4 through 4.6). If Rate 1/8 is selected, the encoder will not attempt to

17 characterize any periodicity in the speech residual, but will instead just characterize its energy contour. At Rates 1/2 and

18 1 the encoder will apply the RCELP algorithm to match a time-warped version of the original speech residual.

19 A number of parameters are included at Rate 1 to provide enhanced performance in poor channel conditions. These

20 include the spectral transition indicator and the delay difference (Δ_τ). The packet formatting module (see 4.8) accepts all

21 of the parameters calculated and quantized in the rate-specific encoding modules, and formats a packet appropriate to the

22 selected rate. The formatted packet is then presented to the multiplex sub-layer. The rate decision is also presented to

23 the multiplex sub-layer. Sections 4.1 through 4.8 describe each of the encoder modules in detail.

4.2 Service Option 68

25 The EVRC-B coder uses ~~an~~ a capacity operating point and speech-type dependent coding scheme selection. Primarily it

26 consists of Code-Excited-Linear-Prediction (CELP), Prototype-Pitch-Period (PPP), Noise-Excited-Linear-Prediction

27 (NELP) coding methods. CELP is used for Full-Rate and Half-Rate, PPP is used for Full-Rate and Quarter-Rate, NELP

28 is used for Quarter-Rate, and Eighth-Rate.

1 Table 4.2-1 lists where the different coding schemes are used. Full-Rate CELP (FCELP) is used to code transitions,
2 poorly voiced frames and certain normal voiced frames as well. Half-Rate CELP (HCELP) is used to code ends of words
3 mostly and dimmed frames. Full-Rate PPP (FPPP) and Quarter-Rate PPP (QPPP) are used to code strongly voiced
4 frames. Quarter-Rate NELP (QNELP) is used for unvoiced frames. Silence is coded by a simple single-gain shaped
5 random noise waveform as in SO 3.

6 **Table 4.2-1 EVRC-B Coding Schemes**

Coding scheme	Type of speech
FCELP (Full-Rate CELP)	Transient, bump-up and some voiced frames
HCELP (Rate 1/2 CELP)	Ends of words, Dim-and-burst signaling
Special HCELP	Packet level dimming of FCELP
FPPP/QPPP (Full/Quarter-Rate Prototype Pitch Period)	Voiced frames
Special HPPP	Packet level dimming of FPPP
QNELP (Rate 1/4 Noise Excited LP)	Unvoiced frames
Special HNELP	Half-Rate frame containing NELP frame for use in situations not allowing Quarter-Rate (e.g., RATE_REDUC = '111')
Rate 1/8	Silence frames

7

8 The encoder uses the 4 primary traffic packet types permitted by [N3] and [N4] Multiplex Option 1: Rate 1 (171
9 bits/packet), Rate 1/2 (80 bits/packet), Rate 1/4 (40 bits/packet), and Rate 1/8 (16 bits/packet). Upon command, the
10 encoder will produce a blank packet (which contains no bits) or other than a Rate 1 packet, i.e., Rate 1/2 maximum;
11 otherwise, the encoder makes its own determination about what type of packet to generate. Bit allocations for each
12 packet type are given in Table 4.2-2.

13 **Table 4.2-2 Bit Allocations by Packet Type**

Parameter	Full-Rate CELP	Half-Rate CELP	Special Half-Rate CELP	Full-Rate PPP	Special Half-Rate PPP	Quarter-Rate PPP	Special Half-Rate NELP	Quarter-Rate NELP	Silence encoder
Mode Bit	1		1	1	1	1		1	
Special Half-Rate ID			2		2		2		
LSP	28	22	28	28	28	16	16	16	8
Lag/Special Packet ID	7	7	7	7	7	4	7		
Additional Lag			7		7				
ACB Gain	9	9	9						
FCB Index	105	30							
FCB Gain	15	12							
Delta lag	5								
Amplitude				28	28	18			
Global Alignment				7	7	0			

Parameter	Full-Rate CELP	Half-Rate CELP	Special Half-Rate CELP	Full-Rate PPP	Special Half-Rate PPP	Quarter-Rate PPP	Special Half-Rate NELP	Quarter-Rate NELP	Silence encoder
Band Alignments				99					
Energy VQ							17	17	6
Filter Shape							2	2	
Reserved	1		26	1		1	36	4	2
Total	171	80	80	171	80	40	80	40	16

1

2 All the reserved bits shall be set to 0. The algorithms in the sections that follow are presented in terms of block diagrams,
3 mathematical equations where appropriate, and in pseudo-code where equations would be cumbersome. Inputs, outputs,
4 and processing are defined for each processing block or module. Internal variables for each module are defined when
5 used. Modules are decomposed into sub-modules at increasing levels of detail until a defining equation or block of
6 pseudo-code describing the lowest level of processing is reached. The algorithm descriptions are designed to maximize
7 clarity, not to illustrate the most efficient implementation. For example, no attempt is made to illustrate memory
8 conserving techniques such as reuse of scratch variables or buffers. Operations such as Discrete Fourier Transforms
9 (DFTs) or Finite Impulse Response (FIR) filters are described in terms of a defining equation rather than in terms of one
10 of the fast algorithms available to carry them out.

11 Figure 4.2-1 is a high-level view of the EVRC-B speech encoder showing all major modules. Inputs to the encoder are
12 the speech signal vector, {s(n)}, Capacity Operating Point, and an external rate command signal. The external rate
13 command may direct the encoder to produce a blank packet or a packet other than Rate 1. If an external rate command is
14 received, it will supersede the encoder's internal rate selection mechanism.

15
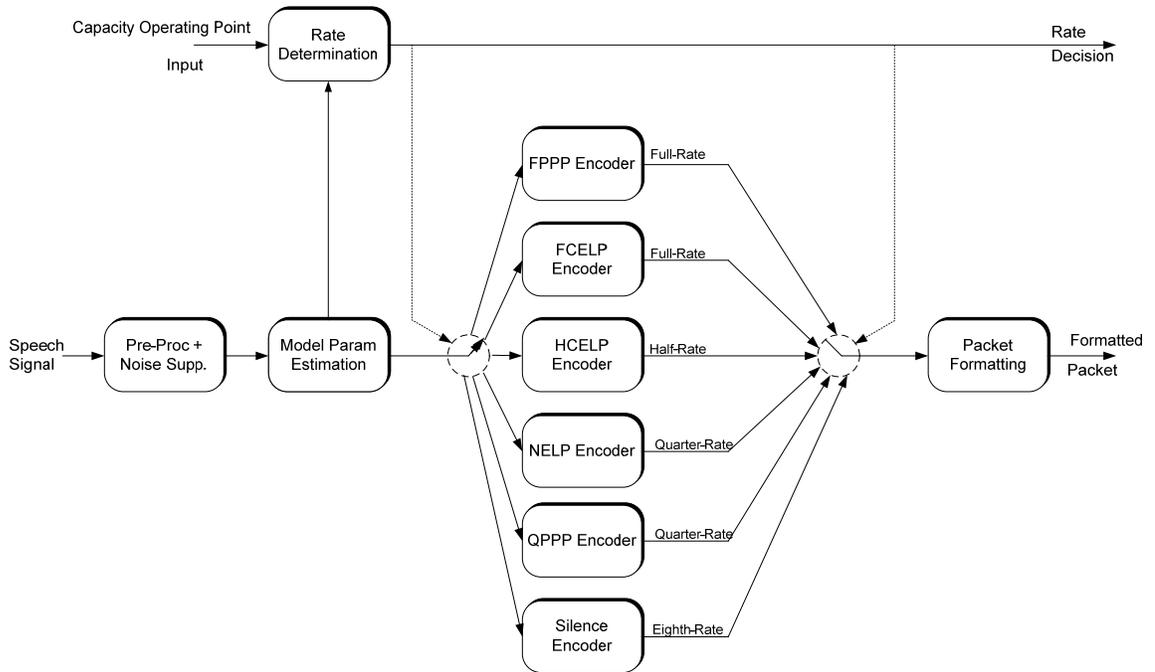
Figure 4.2-1 Speech Encoder Top-Level Diagram

Figure 4.2-2 is a speech decoder top-level diagram.
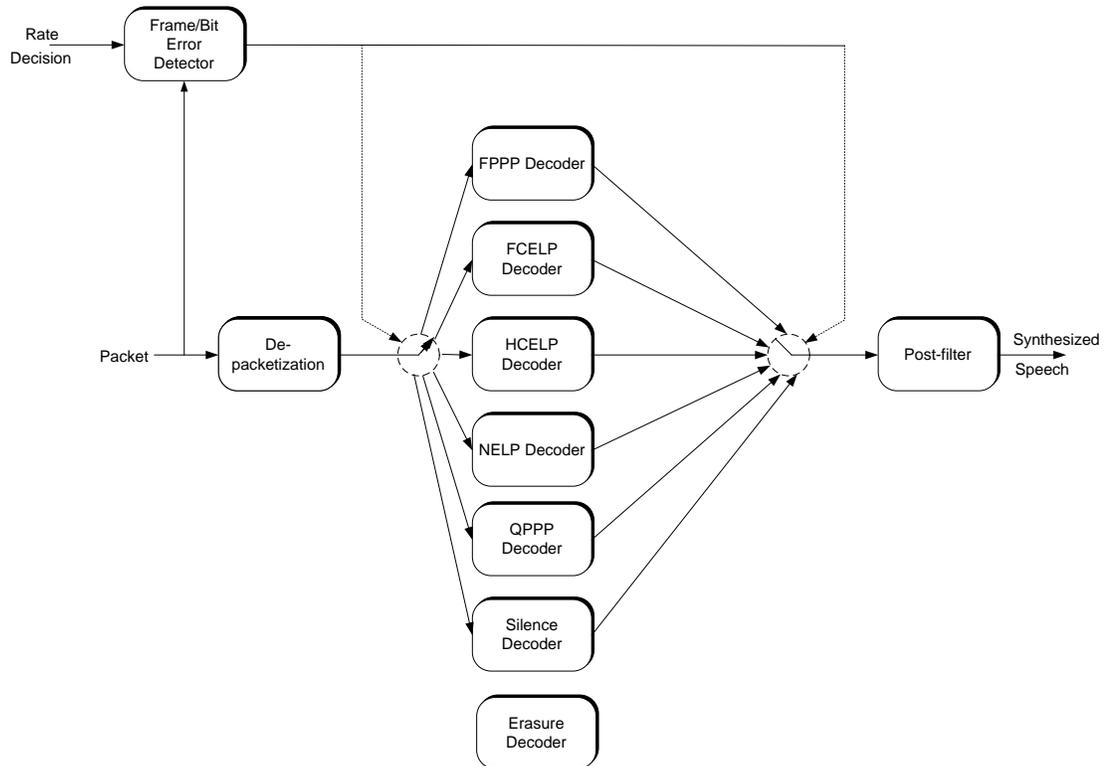
Figure 4.2-2 Speech Decoder Top-Level Diagram

After usual front-end pre-processing and voiced activity decision, the input speech and residual are analyzed to compute open-loop-pitch information as well as periodicity information. This information along with other features is used to categorize speech into transition, voiced, unvoiced, end-of-word, silence.

Then there is a further categorization of frames into one of the above coding schemes and rates which is <u>capacity</u> operating point-dependent.

The linear-prediction-coefficients (LPCs) are quantized according to the rate and coding scheme. The residual is computed using these quantized LPCs and modified to get a target waveform matching as perfectly as possible to a pitch contour without losing perceptual quality, as in SO 3. After this encoding of either this target waveform or a representative period of it or an energy contour of it is done with appropriate quantization and corresponding unquantization at the decoder, synthesis and update of memories. A perceptual quality enhancing adaptive postfilter is applied to the output of the decoder.

4.2.1 Pre-Processing

The input speech is processed through a high-pass-filter and noise-suppressor, after which certain parameters such as LPC, Normalized-Auto-Correlation-Function (NACF) are extracted. Voice-activity decision is made before any further processing. Pitch lag and NACF for the current frame are estimated based on the residual of the current frame, previous frame and part of the next frame. Some additional features are also estimated for use in rate/coding scheme decision which follows.

The High-Pass-Filter and Noise-Suppressor are similar to SO 3. The LPC analysis, pitch lag estimation, LPC to LSP conversion, and residual computation are similar to SO 3. New parameters computed NACF and certain other rate decision specific features. Voice-activity decision is similar to that in SO 3.

4.2.2 Rate/Coding Scheme Decision

The rate/coding scheme decision is of two kinds, open-loop and closed-loop, the former being invoked prior to any quantization being done (except for LPC in case of some open-loop decisions), the latter after. The open-loop rate decision is based mostly on parameters and features of input speech, both current and previous. The closed-loop rate decision is based on Quarter-Rate PPP quantizer performance.

The input <u>capacity</u> operating point controls EVRC-B to operate at an arbitrarily desired average-rate. This is first mapped onto one of three Anchor Operating Points, called AOP 0, 1, and 2 respectively. These anchor operating points control the basic behavior of EVRC-B rate-selection while the final rate-selection enables the average rate output of EVRC-B to match the input <u>capacity</u> operating point.

4.2.2.1 Open-Loop Decision

First of all the input signal is classified broadly into speech, end-of-speech and silence. If the anchor operating point chosen is anchor operating point-0, these classifications are correspondingly encoded with Full-Rate CELP, Half-Rate CELP, and silence-encoder. Otherwise, the non-silence part of the input is further classified using multi-level decision logic into transition, voiced, unvoiced and end-of-word. Additional features computed are SNR of previous and current frames, ZCR, high-band and low-band energies of current and previous frame. Each voiced Segment is made sure to be preceded by a transition frame to get good initial memories. Transitions are encoded with Full-Rate CELP, end-of-word is encoded with Half-Rate CELP, unvoiced is encoded with Quarter-Rate NELP, and voiced is further mapped into pattern sets of three frames. If the anchor operating point chosen is anchor operating point-1 the repetitive three-frame pattern set is encoded successively with Full-Rate CELP, Quarter-Rate PPP, and Full-Rate CELP. If the anchor operating

point chosen is anchor operating point-2, this three-frame pattern set is encoded successively with Quarter-Rate PPP, Quarter-Rate PPP and Full-Rate PPP.

Further, if inside the PPP scheme some waveform changes are extreme, the coding scheme is changed either from Full-Rate PPP to Full-Rate CELP or Quarter-Rate PPP to Full-Rate PPP/CELP.

4.2.2.2 Closed-Loop Decision

If the Quarter-Rate PPP encoding fails either in its phase representation or amplitude quantization, the Closed-Loop Rate Decision is used to change the rate to Full-Rate PPP (if anchor operating point-2) or Full-Rate CELP (if anchor operating point-1) and the voiced frame rate pattern is reset.

4.2.2.3 Capacity Operating Point-Switching and Dimming

To handle capacity operating point-switching and dimming, the pattern of coding voiced frames is reset in certain conditions. In anchor operating point-1 or anchor operating point-2, if a Full-Rate frame is dimmed to Half-Rate, the voiced frame pattern begins afresh with Full-Rate (either PPP or CELP depending on anchor operating point-2 or anchor operating point-1 respectively). This holds true in the case of capacity operating point-switching as well where the voiced frame pattern is reset to the usual pattern of choice for the new capacity operating point.

4.2.2.4 Arbitrary Capacity Operating Point Operation

EVRC-B has the capability to operate at an arbitrary average rate. This in turn means that the CDMA network using EVRC-B can achieve an arbitrary capacity operating point (COP) in the forward link as shown in Figure 4.2.2.4-1. The open-loop and the closed-loop rate decision thresholds are dynamically adjusted in order to achieve this arbitrary average rate, as shown in Figure 4.2.2.4-1.

Continuous Range of COPs

Higher Quality Lower Capacity — EVRC-B — **Lower Quality Higher Capacity**
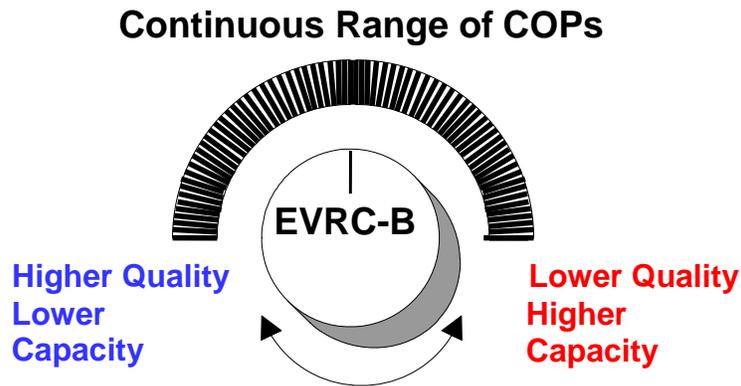
Figure 4.2.2.4-1 Continuous Range of COPs

The channel average rate for active speech, T, is computed as follows,

$$T = \frac{9600 N_{full} + 4800 N_{half} + 2400 N_{quarter}}{N_{full} + N_{half} + N_{quarter}}$$

Given a target average rate for active speech, in channel rate bits-per-second, say $T_{ActiveAvgChRate}$, EVRC-B encoder first determines an anchor operating point according to the following algorithm:

If $(T_{ActiveAvgChRate}$ > 7500) $OP_{anchor} = 0$

1   else if $(T_{ActiveAvgChRate} > 6600)$ $OP_{anchor} = 1$

2   else $OP_{anchor} = 2$

3 Once the anchor operating point has been determined, EVRC-B attempts to meet the average rate-target by adjusting the
4 fraction of Quarter-Rate frames to meet the target average rate. This adjustment is done dynamically by comparing the
5 actual average rate in a window of past 600 active speech frames to the target average rate, and computing an appropriate
6 fraction of frames that would be sent from Quarter-Rate to Full-Rate coding.

7 A fraction, p, of frames that would be sent from Quarter-Rate to Full-Rate is given by,

$$p = \begin{cases} \dfrac{(T_{next} - 7.5)}{(9.0 - 7.5)}; & OP_{anchor} = 0 \\[2mm] \dfrac{(T_{next} - 6.6)}{(7.5 - 6.6)}; & OP_{anchor} = 1 \\[2mm] \dfrac{(T_{next} - 5.75)}{(6.6 - 5.75)}; & OP_{anchor} = 2 \end{cases}$$

8

9 where T_{next} is the dynamic target active speech channel average rate for the next *RATEWIN* active frames given by,

$$T_{next} = 1.98 T_{ActiveAvgChRate} - T_{prev}$$

10

11 Where T_{prev} is the active speech channel average rate for the previous *RATEWIN* active frames.

12 Once the fraction, p, has been determined, the following algorithm is used to send the fraction, p, Quarter-Rate frames to
13 Full-Rate in an almost uniformly distributed manner.

14 For every active frame,

15

```
16      If (OPanchor   =    0) {
17          If   (open loop mode-decision EQUAL to UNVOICED){
18              Patterncount  =    Patterncount   +    1000 x p
19              If  (patterncount >    1000)     {
20      Encode unvoiced frame using full-rate
21      Patterncount =    patterncount   -    1000
22          }
23      }
24      If (OPanchor   =    1) {
25          If   (voiced frame is determined using the open-loop and closed-loop
26      decision to be coded using QPPP){
27              Patterncount  =    Patterncount   +    1000 x p
28              If  (patterncount >    1000)     {
29      Encode QPPP frame using full-rate CELP
30      Patterncount =    patterncount   -    1000
31          }
32      }
33      If (OPanchor   =    2) {
```

```
        If   (voiced frame is determined using the open-loop and closed-loop
    decision to be coded using QPPP and if previous frame was a QPPP frame)    {
            Patterncount  =  Patterncount  +   1000 x p
            If  (patterncount >  1000)    {
    Encode QPPP frame using full-rate CELP
    Patterncount =   patterncount  -   1000
       }
       }
```

4.2.3 Full-Rate CELP and Half-Rate CELP

The CELP algorithms are similar to those in SO 3 except for the following exceptions.

The fixed codebook in case of Full-Rate is more efficient and is with one pulse fewer per subframe. This is achieved by exploring and covering all possible combinations of 7 pulses and two signs on each possible location in a subframe. Packing and unpacking is slightly more complex than in SO 3. The fixed codebook of Half-Rate (used for end-of-word only) also is made somewhat more efficient by discarding undesirable pulse combinations from SO 3 and adding some new pulse combinations.

The Full-Rate CELP decoder is different in the way it handles the frame after erasure or the frame after a Quarter-Rate PPP which in turn would have followed an erasure. In the latter case, the previous Quarter-Rate PPP frame is re-synthesized with the proper lag and this is used for the adaptive codebook memories. In either case, after reconstruction of the residual of the current CELP frame, it could be replaced with a waveform-interpolated version of the representative prototypes of the current and previous reconstructed prototypes to eliminate clicks after erasure.

The special Half-Rate CELP frame is used when a Full-Rate CELP frame is converted by a packet-level signaling interworking function to a Half-Rate frame. The packet-level conversion is done by discarding the FCB index and gain bits in the Full-Rate, and repacking the remaining bits into a special Half-Rate frame. To indicate to the EVRC-B decoder that it is a special Half-Rate CELP frame, the 7 lag bits of the HCELP frame are set to 0x6E—a value that is never produced at the EVRC-B encoder.

4.2.4 Full-Rate PPP and Quarter-Rate PPP and Special Half-Rate PPP

4.2.4.1 Background

It was the perceptual importance of the periodicity in voiced speech that motivated the development of the (Prototype Pitch Period) PPP coding technique. PPP exploits the fact that pitch-cycle waveforms in a voiced Segment do not change quickly in a frame. This suggests that we do not have to transmit every pitch-cycle to the decoder; instead, we could transmit just a representative prototype pitch period. At the decoder, the nontransmitted pitch-cycle waveforms could then be derived by means of interpolation. In this way, very low-bit-rate coding can be achieved while maintaining high quality reconstructed voiced speech. Figure 4.2.4.1-1 illustrates an example of how the pitch-cycle waveforms are extracted and interpolated. We will refer these pitch-cycle waveforms as Prototype Pitch Periods (abbreviated as PPPs).
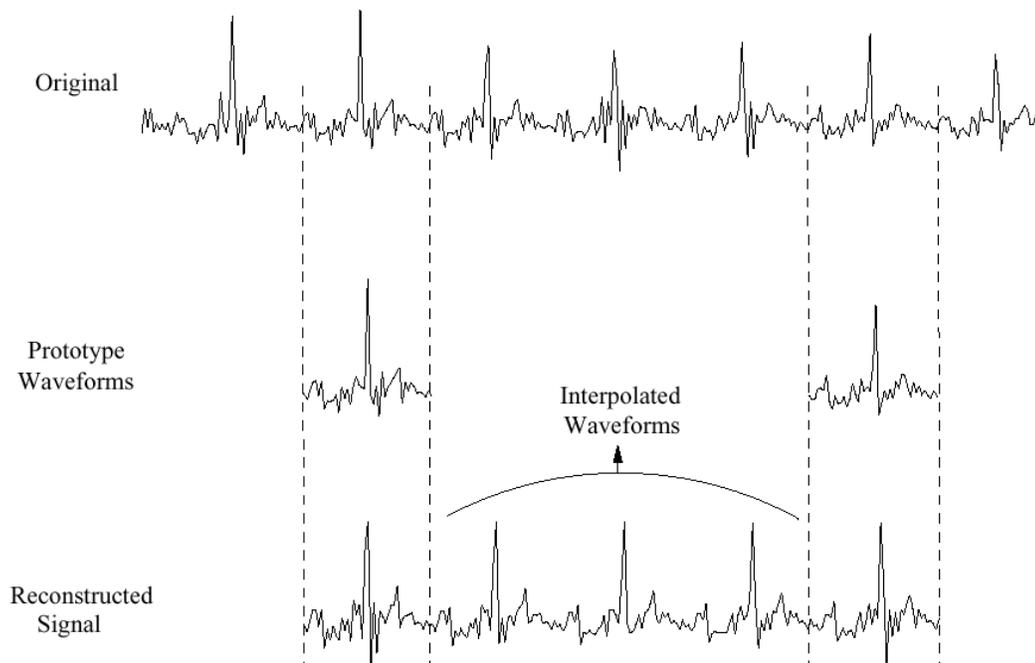
Figure 4.2.4.1-1 Illustration of the Principles of PPP Coding

In EVRC-B, PPP operates at two rates – Full-Rate (FPPP) and Quarter-Rate (QPPP). They both share the same fundamental concepts but differ in their bit-allocation schemes. In the context of this implementation, the quantization of PPP is carried out in frequency domain. Hence the time-domain signal is converted to a discrete-Fourier series (DFS), whose amplitudes and phases are independently quantized. The description of the PPP model will be first given.

4.2.4.2 PPP Model

Figure 4.2.4.2-1 presents a high-level schematic diagram of PPP coding scheme. Front-end processing including LPC analysis is same for this scheme. However, the LSP quantization for the QPPP is a Moving-Average predictive quantizer which exploits the stationarity of voiced speech.
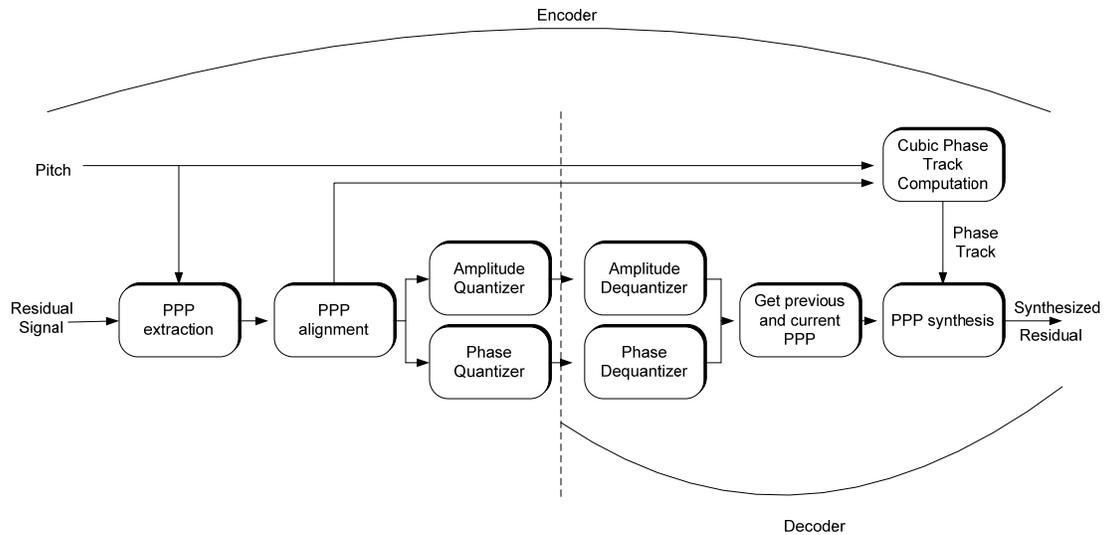
Figure 4.2.4.2-1 Block Diagram of PPP Coding Scheme

4.2.4.3 Further Processing of PPP Frames

Then, the corresponding residual is obtained which in turn is modified the same way as in CELP to follow a pitch contour similar to RCELP in SO 3.

Next, a PPP is extracted from the modified residual signal on a frame-wise basis. The length of the PPP is determined by the lag estimate supplied by the pitch estimator. Special attention needs to be paid to the boundaries of the PPP during the extraction process. Since each PPP will undergo circular rotation in the alignment process (as will be seen later), the energy around the PPP boundaries needs to be minimized to prevent discontinuities after circular rotation (where the left side of the PPP meets the right side). Such minimization can be accomplished by slightly jittering the location of the extraction.

After the extraction, the PPP of the current frame needs to be time-aligned with the PPP extracted from the previous frame. Specifically, the current PPP is circularly shifted until it has the maximum cross-correlation with the previous PPP. The alignment process serves two purposes: (1) facilitates PPP quantization especially in low-bit-rate predictive quantization schemes (2) facilitates the construction of the whole frame of excitation in the synthesis procedure which will be discussed next.

The aligned PPPs are then quantized and transmitted to the decoder. To create a full-frame of excitation from the current and previous PPPs, we need to compute an instantaneous phase track which is designed carefully so as to achieve maximum time-synchrony with the original residual signal. For this purpose, a cubic phase track can be employed along with four boundary conditions: (1) the initial lag value, (2) the initial phase offset, (3) the final lag value and (4) the final phase offset. At low bit rates, the final phase offset can be computed by the RCELP pitch contour.

Having created the entire frame of quantized residual, the decoder concludes its operation with LPC synthesis filter and memory updates.

1 ## 4.2.4.4 Amplitude Quantization

2 Figure 4.2.4.4-1 shows the block diagram of the amplitude quantizer used in EVRC-B.

3
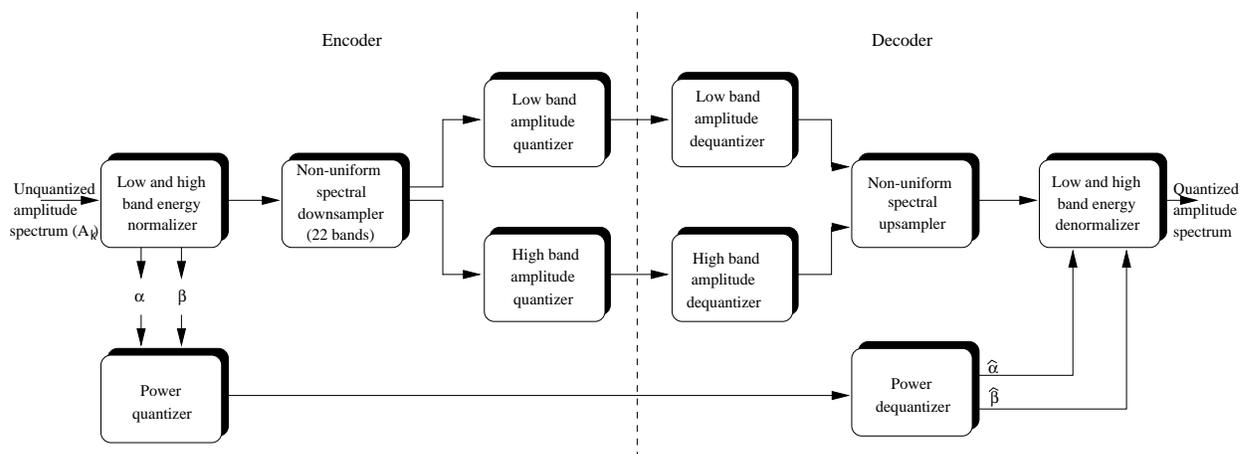
4 **Figure 4.2.4.4-1 A Block Diagram of Amplitude Quantization and Dequantization Modules**

5

6 As a first step, an amplitude spectrum is first normalized by two scaling factors at the encoder – one for the low band and
7 one for the high band. The two resulting scaling factors are vector-quantized in the logarithmic domain and transmitted
8 to the decoder. Next, the normalized spectrum is then non-uniformly downsampled to 22 spectral values according to the
9 bin partition shown in Figure 4.2.4.4-2.

10
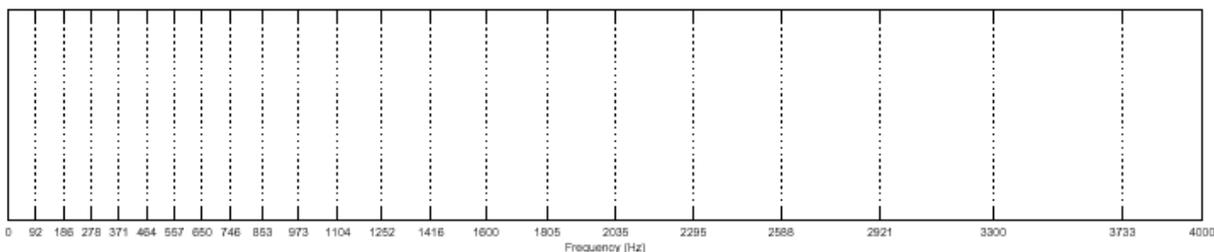
11 **Figure 4.2.4.4-2 An Illustrative Non-Uniform Band Partition Used PPP Quantization**

12 Specifically, each harmonic is first associated with a bin. Then, an average magnitude of the harmonics falling in each
13 bin is computed and hence, a vector of 22 values is produced. Figure 4.2.4.4-3 demonstrates an example of such non-
14 uniform downsampling process.
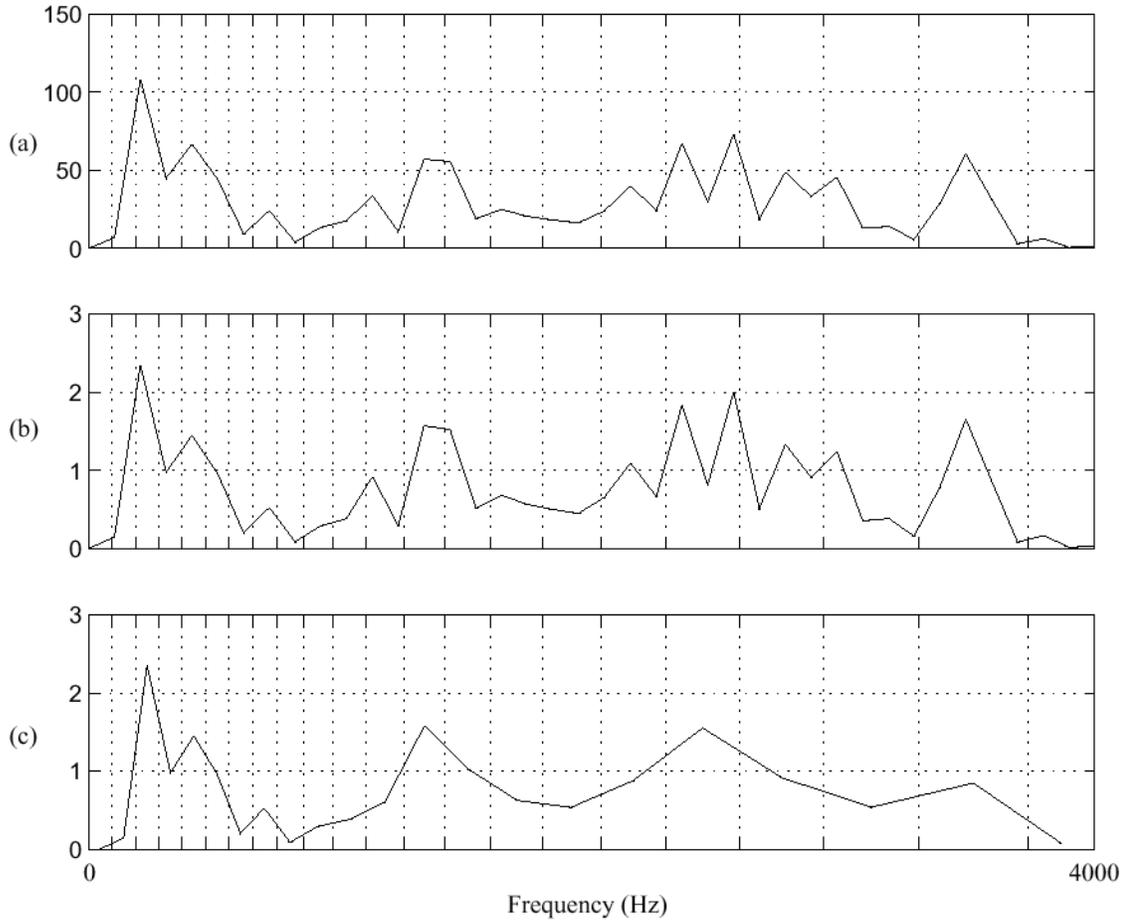
Figure 4.2.4.4-3 Illustration of the Effect of Spectral Downsampling and Upsampling

The vertical grid lines are based on the non-uniform band partition shown in Figure 4.2.4.4-3 – (a): the original amplitude spectrum of a PPP; (b): the normalized version of the spectrum; and (c): the non-uniformly-downsampled version of the spectrum.

The significance of this downsampling/averaging process is to transform a variable dimension vector to a fixed dimension vector. It also takes advantage of the frequency-dependent frequency resolution of the human auditory system and removes perceptually irrelevant information existed in the spectra. Finally, the downsampled spectrum is split into two spectra (the low and high bands) and each of which is separately quantized. At the decoder, the low and the high band spectra are first reconstructed from the bit-stream transmitted from the encoder. The two spectra are then combined and sent to a non-uniform spectral upsampler. Afterwards, the scaling factors recovered from the bit-stream are used to denormalize the upsampled spectrum to reconstruct the quantized spectrum.

In QPPP, both the scaling factors and the spectra are quantized differentially whereas in FPPP, they are done so in a nondifferential manner.

4.2.4.5 Phase Quantization

For QPPP, no bit is spent on phase quantization. The phase spectrum of the current PPP can be readily derived from combining the phase spectrum of the previous PPP and the contribution from the RCELP pitch contour.

1   For FPPP, we adopted a novel multi-band alignment approach to code phase spectrum. As demonstrated in Figure
2   4.2.4.5-1, the effect of applying one linear phase-shift to a PPP is equivalent to its circular rotation in time-domain.

$$+ k\theta$$

$$=$$
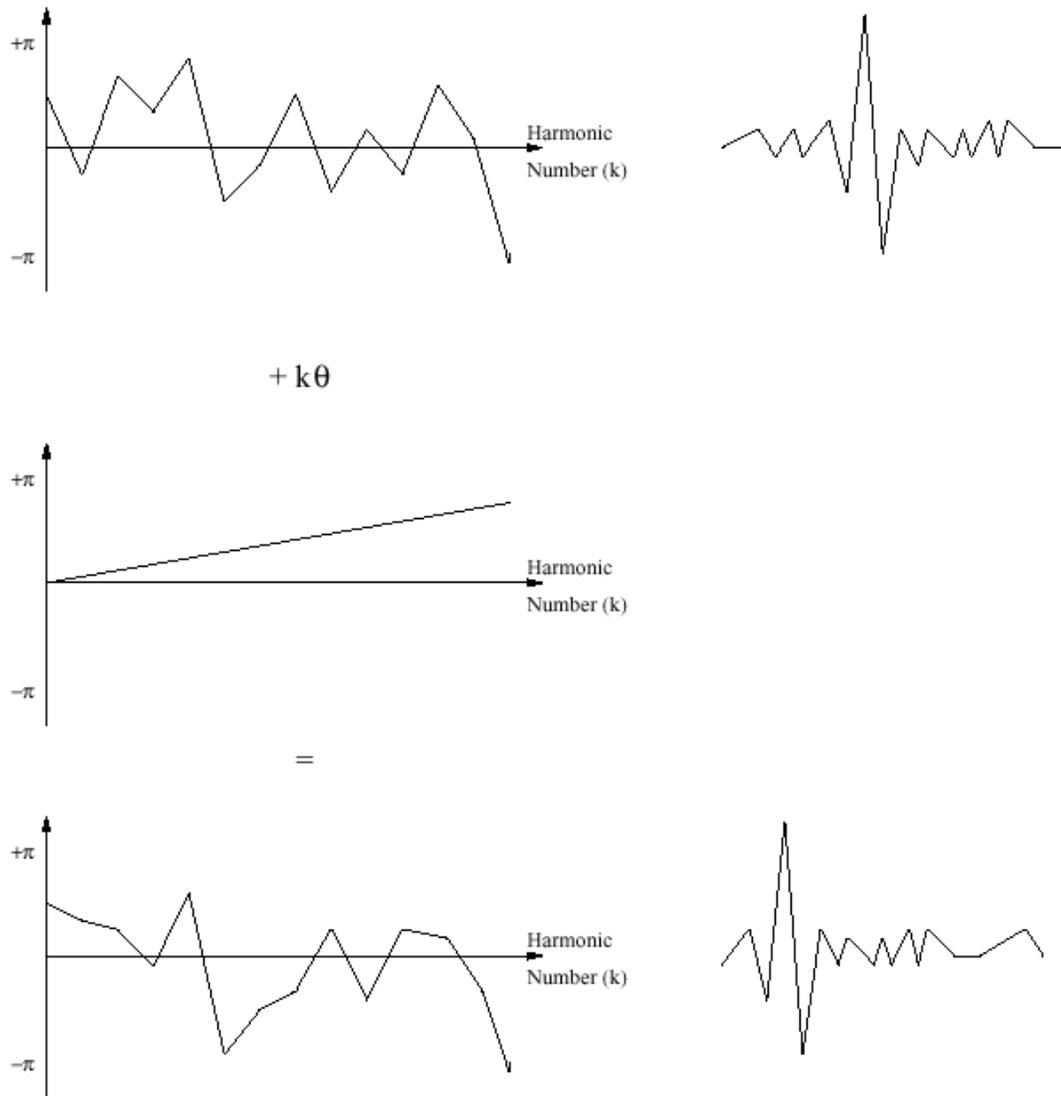
3

4   **Figure 4.2.4.5-1 Demonstration of the Effect of a Single-Band Alignment on a Single PPP**

5   By extending the same idea to multiple sub-bands, one can begin controlling the "shape" details of a PPP by applying
6   different phase-shifts to different sub-bands. An example of which is illustrated in Figure 4.2.4.5-2 for the case N=6 (six
7   sub-bands).

$$+k\theta_0 \ +k\theta_1 \ +k\theta_2 \ +k\theta_3 \ +k\theta_4 \ +k\theta_5$$
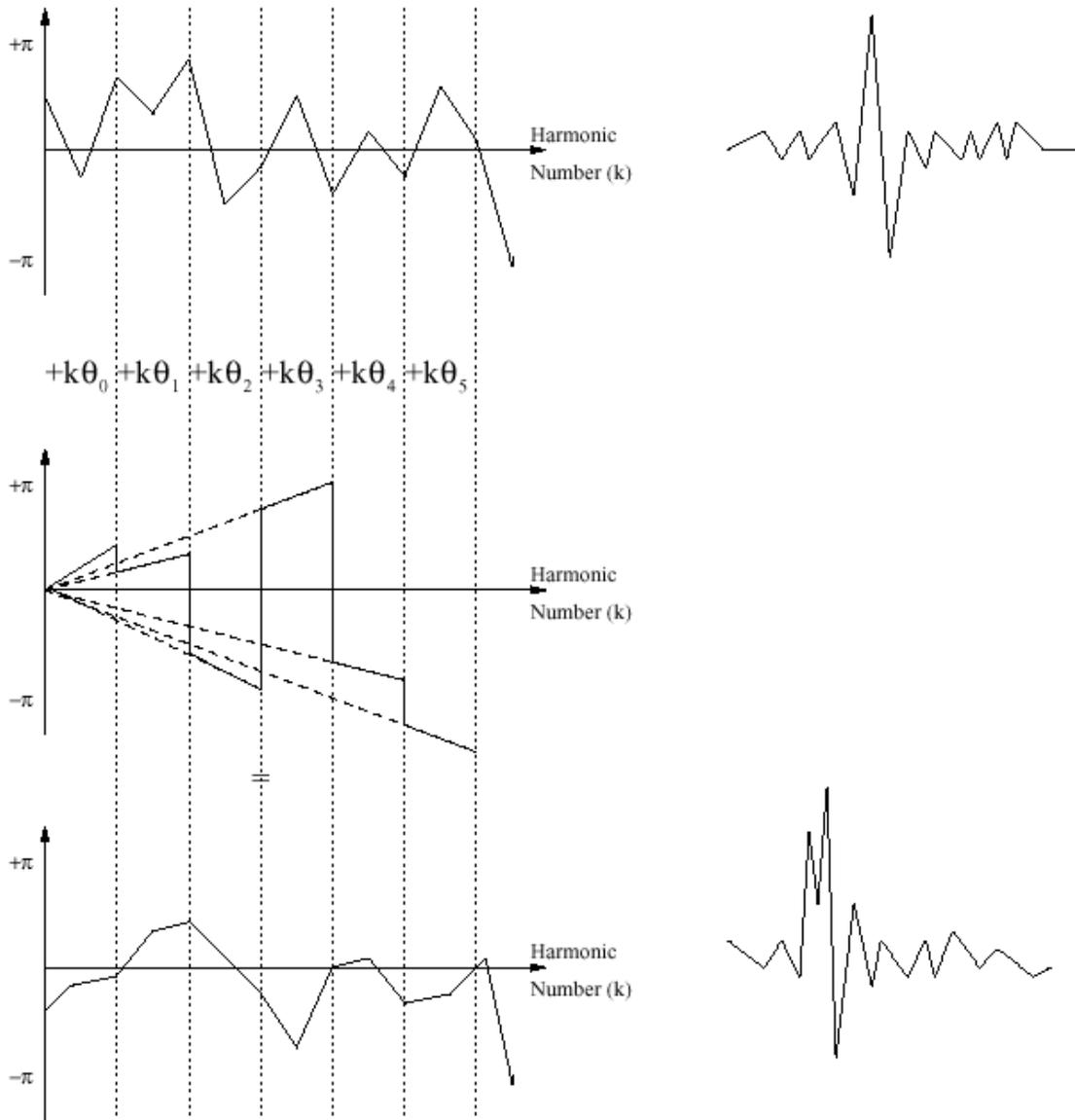
Figure 4.2.4.5-2 Demonstration of the Effect of Multi-Band Alignments on a Single PPP

Hence, the problem of phase quantization has been transformed into quantization of a series of linear phase shifts. Similar to the conventional alignment process, the search of these linear-phase shifts can be performed in the time-domain. The resulting linear-phase shifts are quantized in a scalar manner. The resolution and the range of each phase shift are chosen such that it achieves the target bit rate.

4.2.4.6 Special Half-Rate PPP Frame

The special Half-Rate PPP frame is used when a Full-Rate PPP frame is converted by a packet-level signaling interworking function to a Half-Rate frame. The packet-level conversion is done by discarding the band alignment bits in the Full-Rate, and repacking the remaining bits into a special Half-Rate frame. To indicate to the EVRC-B decoder that it

is a special Half-Rate PPP frame, the 7 lag bits of the HPPP frame are set to 0x6E—a value that is never produced at the EVRC-B encoder.

4.2.4.7 Detection of Outliers

One peculiarity of the PPP scheme is the detection and treatment of unsuitable frames. If the shape/length of the current PPP is deemed to change too much from the PPP of the previous frame (not enough periodicity), PPP is abandoned in favor of FCELP. As briefly mentioned before, a separate closed-loop measure is also devised to capture the outliers of the amplitude and phase quantizers in QPPP. When an outlier is detected in a particular frame, the rate of that frame is then increased to Full-Rate (FCELP or FPPP).

4.2.5 Frame Erasure Handling

Whenever a frame is lost due to erasure, relevant parameters from the previous frame are used and a simple frame repeat mechanism is performed on the lines of the previous frame's decoding mechanism.

4.2.6 Bad Rate/Packet Handling

If incorrect rate is received, checks which are always performed on certain parameters would fail and this would be the detection of bad rate/packet and an erasure would be flagged and this packet would be treated as lost and a frame erasure handling mechanism replaces the regular decoder.

4.3 Service Options 70 and 73

Traditional telephony systems use narrowband speech signals, with a frequency range of 300 Hz to 3400 Hz. A significant improvement in perceived speech quality and intelligibility is obtained with wideband speech, covering the larger frequency band of 50 Hz to 7 kHz. EVRC-WB and EVRC-NW or Service Options 70 and 73 leverages many of the techniques used in EVRC-B and builds on it for coding wideband speech signal. Service Options 70 and 73 also contain a generic audio coding mode that could be used for handling non-speech signals like music-on-hold, and ringback tones. Service Option 70 also contains two capacity operating points (see Table 2.5.1.2-1, SO 70 capacity operating points '100' and '111') that are interoperable with SO 68 (same as capacity operating points '000', and '111' respectively of SO 68). In other words, Service Option 70 supports a couple of capacity operating points that use the exact same Full-Rate CELP, Half-Rate CELP, and Eighth-Rate CELP algorithm and bit-stream as those used by SO 68. SO 73 capacity operating point '000' is interoperable with SO 70 (same as capacity operating point '000' of SO 70). Service Option 73 also contains seven capacity operating points (see Table 2.6.1.2-1, SO 73 capacity operating points '001' - '111') that are interoperable with SO 68 (same as capacity operating points '000', '002' - '111' respectively of SO 68). In other words, Service Option 73 supports seven capacity operating points that use the exact same Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate CELP algorithm and bit-stream as those used by SO 68. The following figure illustrates this design of EVRC-WB. The same concept applies to EVRC-NW.
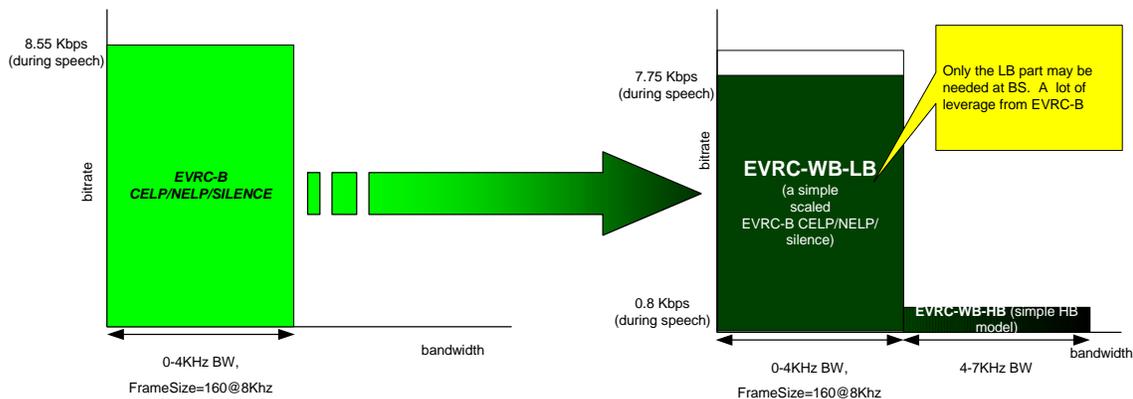
Figure 4.3-1 EVRC-WB Conceptual Overview

As shown in Figure 4.3-1, the EVRC-WB consists of two parts:

1. A low-band coder, referred to in this document as EVRC-WB-LB, which is a slightly modified version of the algorithms used in EVRC-B (i.e., SO 68.). This is an 8 kHz coder like the traditional coders (EVRC-A, EVRC-B etc). Note: Only the lowband portion (EVRC-WB-LB) needs to be implemented if the input and ouput PCM are sampled at 8 kHz (for example at the Base-station for mobile-to-PSTN calls). Further more, when the encoder input PCM is sampled at 8 kHz, an SO 70 encoder could use Capacity Operating Points '100' and '111' (see Table 2.5.1.2-1) where the encoder can use the same Full-Rate CELP, Half-Rate CELP, and Eighth-Rate algorithms and bitstreams used in SO 68 (interoperability/leverage between EVRC-WB and EVRC-B). Similarly, an SO 73 encoder could use Capacity Operating Points '001' - '111' (see Table 2.6.1.2-1) where the encoder can use the same Full-Rate CELP, Full-Rate PPP, Half-Rate CELP, Quarter-Rate PPP, Quarter-Rate NELP, and Eighth-Rate algorithms and bitstreams used in SO 68 (interoperability/leverage between EVRC-NW and EVRC-B).

2. A high-band coder, referred to in this document as EVRC-WB-HB, for coding the higher-frequency band.

In the rest of the document,

- EVRC-WB-LB refers to the low-band parts of EVRC-WB and EVRC-NW

- EVRC-WB-HB refers to the high-band parts of EVRC-WB and EVRC-NW

EVRC-WB refers to a combination of LB and HB portions of EVRC-WB, and similarly EVRC-NW refers to a combination of LB and HB portions of EVRC-NW.

Figure 4.3-2 shows the wideband encoder. The input signal consists of wideband speech sampled at 16 kHz, which is inputed to a 50 Hz cutoff HPF followed by wideband noise suppression (WB NS) preprocessing modules. An analysis filterbank operates on the WB NS output to generate a lowband signal sampled at 8 kHz, and a high-band signal which is sampled at 7 kHz. This high-band signal represents the band from 3.5 to 7 kHz in the input signal, and the final reconstructed wideband signal will be limited in bandwidth to 7 kHz. Note that there is a 500 Hz overlap between the low and high bands, allowing for a more gradual transition between the bands.

The lowband signal is encoded using a modified version of the algorithms used in EVRC-B speech coder, which employs CELP and NELP coding schemes (Note: EVRC-B also uses PPP coding schemes) with a frame size of 20 milliseconds. Reusing the existing EVRC-B codec modules facilitates implementation and saves program and ROM storage. An additional advantage of the split-band coding described in this document is that it enables obtaining a coded narrowband-only signal without having to go through the steps of wideband decoding and narrowband encoding. More

precisely, the narrowband bit stream is found by simply stripping the high-band side information portion off the wideband bit stream. This may be useful in scenarios where wideband playback is not possible outside of the knowledge of the encoder. For example, in the Base Station encoder when being used in the context of a mobile-to-land voice call where the PCM input/output is sampled at 8 kHz.

Several signals from the lowband coder are used by the high-band analysis and synthesis; these are: (1) the excitation (i.e., quantized residual) signal from the lowband coder; (2) the quantized LPCs to give first reflection coefficient (as an indicator of the spectral tilt of the narrowband signal); (3) the quantized adaptive codebook gain; and (4) the quantized pitch lag.

The high-band signal is pre-processed by a click-remover, which removes short energy bursts naturally occurring in speech. Unattenuated, these bursts interfere with the parametric high-band coder and cause artifacts in the reconstructed signal. The pre-processed high-band signal is then analyzed to obtain a parametric representation of the temporal and spectral envelopes. Next, a multiplexer (MUX) combines the coded parameter information with the lowband bit stream into a wideband bit stream, which is transmitted to the decoder.
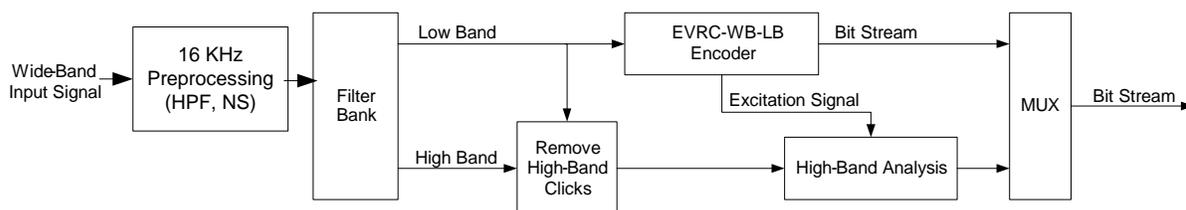
Figure 4.3-2 Wideband Encoder

The wideband decoder is shown in Figure 4.3-3. First, a demultiplexer (DEMUX) separates the bit stream into a lowband and a high-band part. The lowband decoder reconstructs the low-band signal, and passes on the excitation signal, first reflection coefficient, adaptive codebook gain and pitch lag to the high-band synthesis module. This module synthesizes the high-band signal using the side information from the encoder and the signals from the narrowband decoder. The low-band and high-band signals are combined in a synthesis filterbank, which outputs the reconstructed wideband speech signal, sampled at 16 kHz.
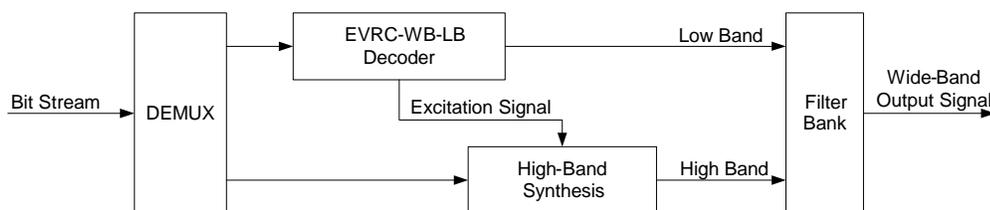
Figure 4.3-3 Wideband Decoder

A more detailed description of the high-band codec can be found in Section 4.3.3.

4.3.1 Leverage from EVRC-B to Obtain EVRC-WB-LB

Several changes were made to the EVRC-B codec to derive the lowband part of the wideband system. A summary of the most significant modifications is given as:

- Preprocessing (Noise Suppression and High-pass filter) operate on 16 kHz

- Perceptual Weighting, including Harmonic Noise Weighting

- The number of fixed codebook pulses per subframe has been reduced from 7 to 6 or 5 pulses.

- Closed loop delay contour adjustment

- ACB gain and FCB gains in CELP loop have joint optimization.

- Pitch pre-processing of excitation in decoder

- The delta lag (the difference between current and previous pitch lags, used after a frame erasure) is no longer transmitted.

- The Prototype Pitch Period (PPP) coding mode is not used.

- Noise Excited Linear Prediction (NELP) is used to send a payload at Half-Rate, which includes the description for the high band. NELP uses the 28-bit LSPVQ (same as Full-Rate). There are also some minor filtering changes.

- The original Half-Rate frame type (Half-Rate CELP) is used for Dim & Burst only.

- Eighth-Rate has new LSPVQ and Energy Quantization, but is very similar to that in EVRC-B

- Minor modifications to VAD and mode-decision

4.3.2 Bit Allocation

Table 4.3.2-1 outlines how payload bits are divided between lowband and high-band codec.

Table 4.3.2-1 Bit Allocation

Rate Set 1 (bits)	Low band	High band	Total
Full-Rate	155	16	171
Half-Rate	53	27	75
Eighth-Rate	16		16

4.3.3 Coding the High-Frequency Band of Wideband Speech

4.3.3.1 Click Remover

In wideband speech occasional bursts of energy (audible as clicks) occur in the upper part of the spectrum. These energy bursts are short in duration, typically just a few milliseconds, and span up to several kHz in frequency.

These clicks can have a negative impact on the high-band coding method, in two ways. Firstly, they cause the energy envelope over time to be much less smooth, introducing a sharp peak at the time of the energy burst. This means that either the time envelope needs to be modeled with high temporal resolution, increasing the amount of side information to the decoder, or that the energy burst will be smeared out over time during the reconstruction. A second problem with clicks is that they have a dominating influence on the spectral envelope, as modeled once per frame by the LP parameters. Consequently, the frame containing the click will be synthesized with a spectral envelope that is different from the preceding and following frames, which leads to a perceptually objectionable discontinuity.

1 Instead of trying to model and code these clicks, it is feasible to reduce these clicks in the input signal by means of a pre-
2 processing, without noticeably degrading the quality of the uncoded signal. Subsequently coding the pre-processed signal
3 avoids the aforementioned problems and produces fewer artifacts in the reconstructed signal.

4 The basic idea of the click-removing algorithm is to detect energy bursts that exist in the high band but not in the low
5 band, and then briefly reduce the level of the high-band signal during each of these energy bursts. A detailed description
6 follows below.

7 4.3.3.2 High-Band Analysis

8 Figure 4.3.3.2-1 shows the high-band analysis algorithm. The most important module is the one that generates the high-
9 band excitation signal, and is described in detail at the end of this section. The 7 kHz-sampled high-band input signal is
10 analyzed to produce parameters that, together with signals from the narrowband decoder, are used in the decoder to
11 reconstruct the high band. The parameters represent the spectral and temporal envelope of the high band, and the ratio of
12 harmonic to non-harmonic energy.

13 Six spectral parameters are obtained by a traditional linear prediction (LPC) analysis. For efficient coding, these
14 prediction coefficients are converted to line spectral frequencies (LSFs) and quantized using a vector quantizer. The
15 quantization index is transmitted to the decoder.

16 Five temporal gain parameters each represent the scale factor of a 4 millisecond subframe. The gain parameters are
17 found by comparing the energies in each subframe of the input high-band signal with the unscaled, spectrally shaped
18 high-band excitation signal. The gain parameters are also quantized with a vector quantizer, and the quantization index is
19 transmitted to the decoder.

20 A multiplexer (MUX) combines the two quantization indices representing the spectral and temporal envelopes into a
21 single bit stream. The total bit rate for the high band is 800 bits/second.
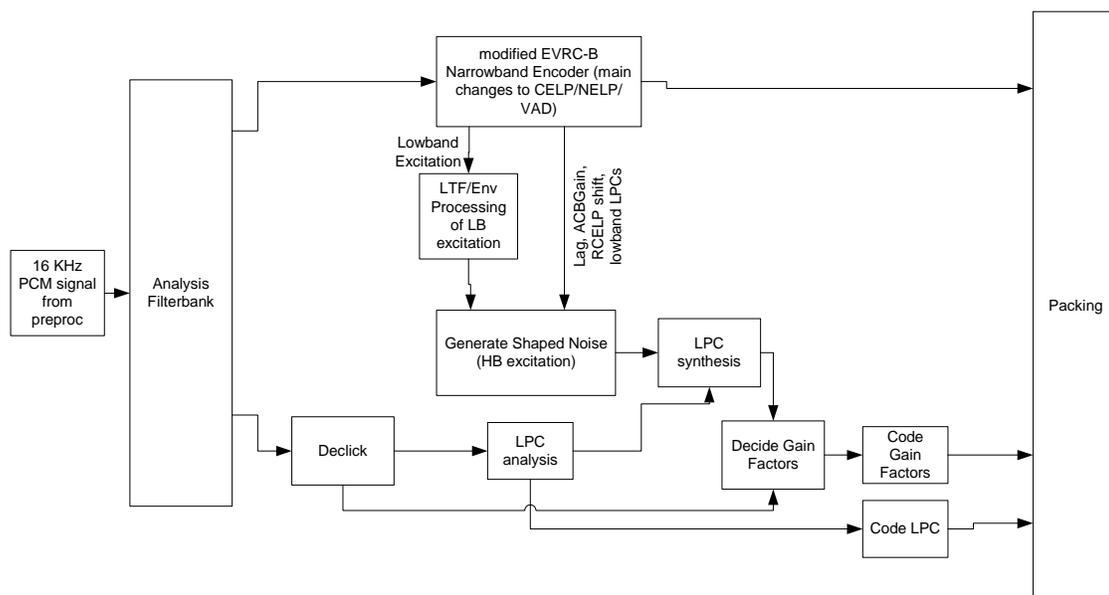
22

23 **Figure 4.3.3.2-1 High-Band Analysis**

4.3.3.3 High-Band Excitation Generation

The high-band excitation signal is derived from the low-band excitation

Figure 4.3.3.3-1 depicts the process that generates the high-band excitation from the narrowband excitation signal. The excitation signal is first run through an all-pass filter of the form

$$H(z) = \frac{-0.7 + z^{-4}}{1 - 0.7z^{-4}} \cdot \frac{0.6 + z^{-6}}{1 + 0.6z^{-6}} \cdot \frac{0.5 + z^{-8}}{1 + 0.5z^{-8}} .$$

This filter reduces the sparseness that results from encoding the low-band signal with a sparse codebook, and is intended to be used during unvoiced speech. The decision to use the output of the filter or the original input signal is derived from the quantized narrowband adaptive codebook gain and first reflection coefficient: unvoiced signals are characterized by a low adaptive codebook gain and a first reflection coefficient that is close to zero or positive (indicating a flat or upwards-tilted spectrum).

Next, a nonlinear function is applied to the above signal. This module up-samples the signal to 16 kHz, takes the absolute value. From here, a 7 kHz-sampled signal is produced using the same high-band analysis filter that was used to split the input signal in a low and a high band. The resulting signal is spectrally flattened with an adaptive 4th order linear prediction filter, to create the harmonically-extended excitation signal.

A modulated noise signal is generated by multiplying a unit-variance white noise signal with the envelope of the harmonically-extended excitation signal. This envelope is obtained by taking the squared value of each sample, smoothing with a first order IIR low-pass filter and taking the square-root of each smoothed sample.

The modulated noise and harmonically-extended excitation signals are now mixed together to create a signal with the right amount of harmonic and noise contents. The gains of the noise and harmonics parts are set adaptively to match the ratio of harmonic to noise energy in the high band of the original speech signal. Typically, the noise gain is between 0.1 and 0.7, and the harmonics gain is derived from the noise gain as $G_{harm} = \sqrt{1 - G_{noise}^2}$, so that the energy of the output signal is not affected by changing the gains.

Note that the high-band excitation is generated in both encoder and decoder. The encoder determines the side information for the gain factors by comparing the synthesized high band with the original high band, and the decoder then uses these gains to properly scale the synthesized high band. For the decoding process to be consistent with the encoding process, it is important that identical high-band excitations be generated in encoder and decoder. This is achieved by (1) using the *quantized* narrowband excitation, which is available to both encoder and decoder, and (2) having the same state for the random white noise generator in encoder and decoder at all times. The latter can be assured, for instance, by making the state of the random generator a deterministic function of information coded earlier within the same frame, such as the narrowband bit stream.
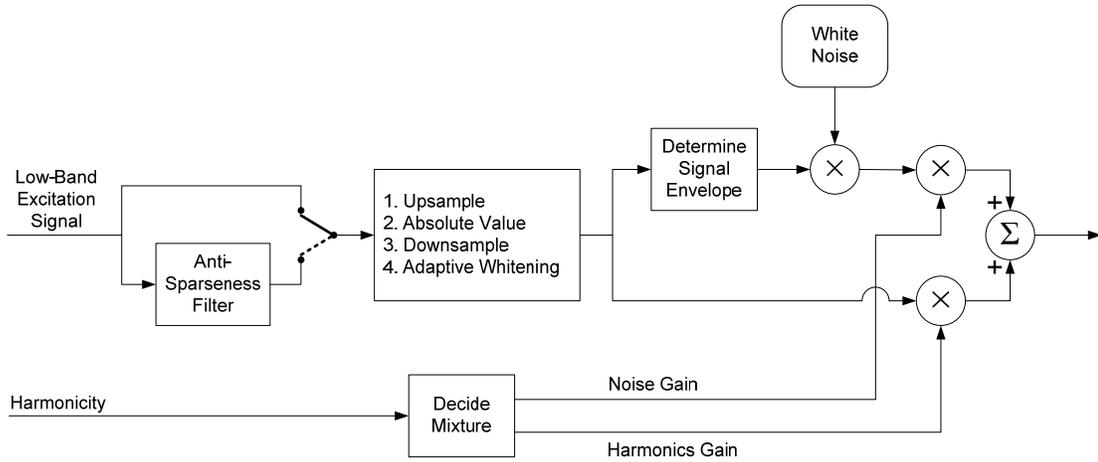
Figure 4.3.3.3-1 Generation of the High-Band Excitation Signal

4.3.4 Silence Encoder

Silence frames are basically coded as narrowband and the upper band parameters are derived from the preceding NELP. Basically, you can think of using DTX in upperband as the model for silence coding.

4.3.5 Decoder Details

Figure 4.3.5-1 shows the high-band decoder. It uses the side information transmitted from the encoder together with signals from the narrowband decoder to generate a high-band signal.

The high-band excitation signal is generated in the same way as in the encoder, using the decoded harmonicity parameter. Then, the excitation signal is spectrally shaped using the dequantized LPC coefficients. In the last step, the dequantized gain factors are used to scale each of the subframes, and the scaled subframes are combined in an overlap-add fashion to avoid discontinuities.
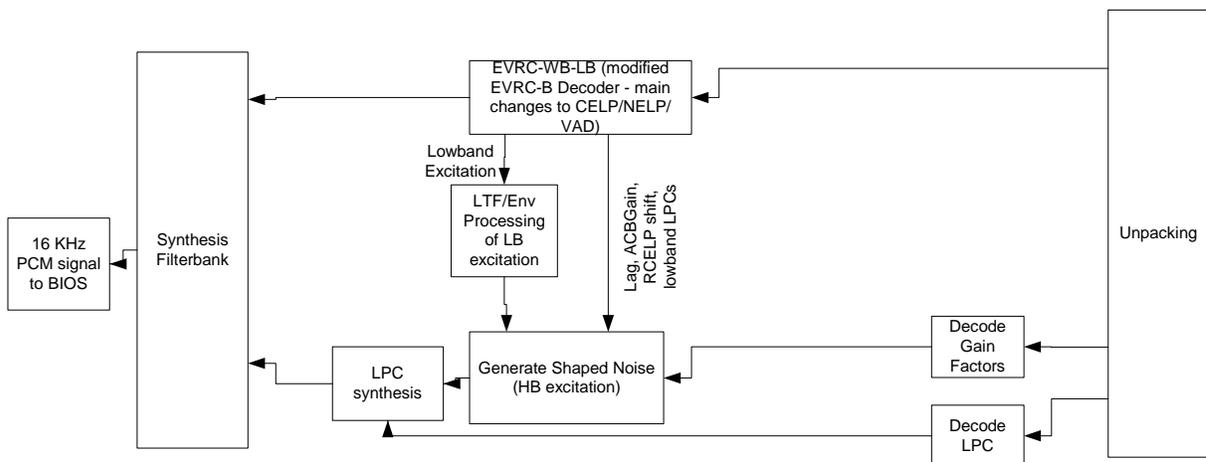
Figure 4.3.5-1 High-Band Synthesis

4.4 Input Signal Preprocessing

Input preprocessing is needed to condition the input signal against excessive low frequency and other background noises that can degrade the codec voice quality. Refer to Section 1.1 for guidelines concerning potential variations of the Input Signal Preprocessing functions.

4.4.1 Fast Automatic Gain Control for Wideband Inputs

For 16 kHz sampled inputs (SO 70 and SO 73 in WB mode), the input sampled speech shall be passed through a fast AGC as described below. The purpose of the AGC is to provide a soft clipping of high level input speech, such that the synthesized speech on the decoder side is less susceptible to the distortions associated with numeric saturation and hard clipping.

Inputs: The input sampled speech signal, $\{s(n)\}$, ranging from -32768 to 32767

Outputs: The AGC'ed speech signal which replaces the input speech, $\{s(n)\} \leftarrow \{s_{agc}(n)\}$

Initialization:

- The filter memory is set to all zeros at initialization.

- The AGC factor: $\gamma_{agc} = 1.0$

- The smoothed AGC factor: $\widetilde{\gamma}_{agc} = 1.0$

Processing:

First compute the peak and RMS values for the current frame of input speech:

$$E_{peak} = 10\log_{10}\left(\max\left\{s^2(n)\right\}\right); \quad 0 \le n < 320 , \tag{4.4.1-1}$$

$$E_{rms} = 10\log_{10}\left(\frac{1}{320}\sum_{i=0}^{319} s^2(n)\right). \tag{4.4.1-2}$$

Then calculate the coarse AGC factor as:

if ($E_{peak} > 87$ and $E_{rms} > 82$) {

$\gamma_{agc} = 0.707$

} else {

$\gamma_{agc} = \min\{1.0, 1.1 \times \gamma_{agc}\}$

}

and finally smooth and apply the AGC factor to the input speech:

for ($i = 0$; $i < 320$; i++) {

$\widetilde{\gamma}_{agc} = 0.999\widetilde{\gamma}_{agc} + 0.001\gamma_{agc}$

$s(i) = \widetilde{\gamma}_{agc}s(i)$

}

1 ## 4.4.2 High-Pass Filter

2 The input sampled speech shall be high-pass filtered as described below:

3 **Inputs:** The input sampled speech signal, $\{s(n)\}$

4 **Outputs:** The high-pass filtered speech signal, $\{s_{hp}(n)\}$

5 **Initialization:** The filter memory is set to all zeros at initialization.

6 **Processing:** The High-Pass Filter for Service Option 3 is a sixth order Butterworth filter implemented as three cascaded
7 biquadratic sections. The cutoff frequency of the filter is 120 Hz. The transfer function of the filter is:

$$H_{hpf} = \prod_{j=1}^{3} H_j(z)$$

8 (4.4.2-1)

9 where each section, $H_j(z)$, is given by:

$$H_j(z) = \frac{a_{j0} + a_{j1}z^{-1} + a_{j2}z^{-2}}{1 + b_{j1}z^{-1} + b_{j2}z^{-2}}.$$

10 (4.4.2-2)

11 The filter coefficients are given as:

12 $a_{10} = 1.0,$ $a_{11} = -2.000125721,$ $a_{12} = 1.000125737,$

13                              $b_{11} = -1.943779252,$ $b_{12} = 0.952444269,$

14 $a_{20} = 1.0,$ $a_{21} = -1.999873569,$ $a_{22} = 0.999873585,$

15                              $b_{21} = -1.866892280,$ $b_{22} = 0.875214548,$

16 $a_{30} = 0.833469450,$ $a_{31} = -1.666939491,$ $a_{32} = 0.833470028,$

17                              $b_{31} = -1.825209384,$ $b_{32} = 0.833345838.$

18

19 The High-Pass Filter for SO 68, SO 70, and SO 73 with 8 kHz sampled inputs is a fourth order filter. The cutoff
20 frequency of the filter is 80 Hz. The transfer function of the filter is:

$$H_{hpf} = \frac{\left(\begin{array}{l} (0.9329833984375) + (-2.7984619140625)z^{-1} + (2.7984619140625)z^{-2} \\ + (-0.9329833984375)z^{-3} \end{array} \right)}{\left(\begin{array}{l} 1 + (2.86181640625)z^{-1} + (-2.7315673828125)z^{-2} \\ + (0.869384765625)z^{-3} \end{array} \right)}$$

21
22 (4.4.2-3)
23

24 The High-Pass Filter for use with 16 kHz input to SO 70 and SO 73 is a fourth order filter. The cutoff frequency of the
25 filter is 50 Hz. The transfer function of the filter is:

26 $$H_{hpf}(z) = \frac{\sum_{i=0}^{4} b_i z^{-i}}{\sum_{i=0}^{4} a_i z^{-i}},$$ (4.4.2-4)

27 where the 32 bit quantified filter coefficients are given as:

28   **b** = {   0.966807253658771514892578125,   -3.867167595773935317993164 0625,

$$5.80072068423032760620117187 5, \qquad -3.86716759577393531799316406 25,$$

$$0.96680725365877151489257812 5 \ \}$$

$$\mathbf{a} = \{ \quad 1.0, \qquad\qquad\qquad\qquad -3.95542053133249282836914062 5,$$

$$5.86737525090575218200683593 75, \qquad -3.86847395822405815124511718 75,$$

$$0.95651939883828163146972656 25 \ \}$$

4.4.3 Noise Suppression

Noise Suppression is used to improve the signal quality that is presented to the Model Parameter Estimator. The procedures by which the Noise Suppression shall be implemented are described in this section. SO 70 and SO 73 with 16 kHz sampled input uses a wideband Noise Suppression algorithm, which is described below in the context of "Wideband NS", or "WB NS". When using SO 70 and SO 73 in narrowband mode (i.e., 8 kHz sampled inputs), the SO 3/SO 68 Noise Suppression (or "NB NS") algorithm shall be used.

Input:

- The output of the High-Pass Filter, $\{s_{hp}(n)\}$

- The open-loop long-term prediction gain from the previous frame β (WB NS only).

Output:

- The primary signal output of the Noise Suppressor is designated as $\{s'(n)\}$

- The zero-delay auxiliary output for use in signal analysis $\{s_{aux}(n)\}$ (WB NS only)

- The SNR band estimates for 0-2 kHz and 2-4 kHz bands, curr_ns_snr[0] and curr_ns_snr[1], respectively

Initialization:

- The following variables shall be set to zero at initialization (frame $m = 0$):

  o The overlapped portion of the input frame buffer, $\{d(m)\}$

  o The pre-emphasis and de-emphasis memories

  o The overlap-and-add buffer, $h(n)$

  o The output buffer history, $s'(n)$; $0 \le n < 160$

- The following shall be initialized to a startup value other than zero:

  o The channel energy estimate, $E_{ch}(m)$, (see 4.4.3.2)

  o The long-term power spectral estimate, $\overline{\mathbf{E}}_{dB}(m)$, (see 4.4.3.6)

  o The channel noise estimate, $E_n(m)$, (see 4.4.3.13)

Processing: Although the frame size of the speech codec is 20 ms, the Noise Suppressor frame size is 10 ms. Therefore, the following procedures shall be executed two times per 20 ms speech frame and the current 10 ms frame shall be denoted m. Figure 4.4.3-1 depicts the Noise Suppression system that is described in the following sections. The signals/blocks added for use with WB NS are designated by dashed lines.

Figure 4.4.3-1 Noise Suppression Block Diagram

4.4.3.1 Frequency Domain Conversion

The input signal is windowed using a smoothed trapezoid window, in which the first D samples of the input frame buffer $\{d(m)\}$ are overlapped from the last D samples of the previous frame. This overlap is described as:

$$d(m,n) = d(m-1, L+n); \qquad 0 \le n < D, \tag{4.4.3-1}$$

where m is the current frame, n is the sample index to the buffer $\{d(m)\}$, $L = 80$ is the frame length, and

$D = 24$ is the overlap (or delay) in samples. For WB NS, $L = 160$, and $D = 160$.

The remaining samples of the input buffer are then pre-emphasized according to the following:

$$d(m, D+n) = s_{hp}(n) + \zeta_p s_{hp}(n-1); \qquad 0 \le n < L, \tag{4.4.3-2}$$

where $\zeta_p = -0.8$ is the pre-emphasis factor. This results in the input buffer containing $L + D = 104$ samples in which the first D samples are the pre-emphasized overlap from the previous frame, and the following L samples are pre-emphasized input from the current frame. For WB NS, $L + D = 320$ samples.

Next, a smoothed trapezoidal window is applied to the input buffer to form the DFT data buffer, $\{g(n)\}$, defined as:

1

$$g(n) = \begin{cases} d(m,n)\sin^2(\pi(n+0.5)/2D) & ; 0 \le n < D, \\ d(m,n) & ; D \le n < L, \\ d(m,n)\sin^2(\pi(n-L+D+0.5)/2D) & ; L \le n < D+L, \\ 0 & ; D+L \le n < M, \end{cases} \tag{4.4.3-3}$$

2 where $M = 128$ is the DFT sequence length and all other terms are previously defined. For WB_NS, the window is a
3 sine window, and the signal $\{g(n)\}$ is defined as:

4

$$g(n) = \begin{cases} d(m,n)\sin(\pi(n+0.5)/(D+L) & ; 0 \le n < D+L \\ 0 & ; D+L \le n < M \end{cases}, \tag{4.4.3-4}$$

5 where $M = 512$ is the DFT sequence length.

6 The transformation of $g(n)$ to the frequency domain is performed using the Discrete Fourier Transform (DFT) defined[†]
7 as:

8

$$G(k) = \frac{2}{M}\sum_{n=0}^{M-1} g(n)e^{-j2\pi nk/M} \ ; \qquad 0 \le k < M, \tag{4.4.3-5}$$

9 where $e^{j\omega}$ is a unit amplitude complex phasor with instantaneous radial position ω.

4.4.3.2 Channel Energy Estimator

11 Calculate the channel energy estimate $E_{ch}(m)$ for the current frame, m, as:

12

$$E_{ch}(m,i) = \max\left\{E_{\min}, \alpha_{ch}(m)E_{ch}(m-1,i) + (1-\alpha_{ch}(m))\frac{1}{f_H(i)-f_L(i)+1}\sum_{k=f_L(i)}^{f_H(i)} |G(k)|^2\right\} \ ; 0 \le i < N_c, \tag{4.4.3-6}$$

13 where $E_{\min} = 0.0625$ is the minimum allowable channel energy, $\alpha_{ch}(m)$ is the channel energy smoothing factor (defined
14 below), $N_c = 16$ ($N_c = 61$ for WB NS) is the number of combined channels, and $f_L(i)$ and $f_H(i)$ are the i-th elements of the
15 respective low and high channel combining tables, which are defined for $N_c = 16$ as:

16 $\quad f_L = \{ 2, 4, 6, 8, 10, 12, 14, 17, 20, 23, 27, 31, 36, 42, 49, 56 \}$,

17 $\quad f_H = \{ 3, 5, 7, 9, 11, 13, 16, 19, 22, 26, 30, 35, 41, 48, 55, 63 \}$,

18 and defined for $N_c = 61$ as:

19 $\quad f_L = \{ 1, 2, 3, 4, 5, 6, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55,
20 $\quad$ 57, 59, 62, 65, 68, 71, 74, 77, 80, 84, 88, 92, 96, 100, 105, 110, 115, 121, 127, 133, 141, 149, 157, 166, 176,
21 $\quad$ 186, 197, 209, 221, 235 \}$,

22 $\quad f_H = \{ 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54,
23 $\quad$ 56, 58, 61, 64, 67, 70, 73, 76, 79, 83, 87, 91, 95, 99, 104, 109, 114, 120, 126, 132, 140, 148, 156, 165, 175, 185,
24 $\quad$ 196, 208, 220, 234, 248 \}$,

25 The channel energy smoothing factor, $\alpha_{ch}(m)$, is defined as:

[†] This atypical definition is used to exploit the efficiencies of the complex Fast Fourier Transform (FFT). The $2/M$ scale factor results from preconditioning the M point real sequence to form an $M/2$ point complex sequence that is transformed using an $M/2$ point complex FFT. Details on this technique can be found in Proakis, J. G. and Manolakis, D. G., *Introduction to Digital Signal Processing*, New York, Macmillan, 1988, pp. 721-722.

$$\alpha_{ch}(m) = \begin{cases} 0.00 & ; m \le 1 \\ 0.45 & ; m > 1 \text{ and NB NS} \\ 0.40 & ; m > 1 \text{ and WB NS} \end{cases} \qquad (4.4.3\text{-}7)$$

So, this means that $\alpha_{ch}(m)$ assumes a value of zero for the first frame ($m = 1$) and a value of 0.45 for all subsequent frames of NB NS, and 0.40 for WB NS. This allows the channel energy estimate to be initialized to the unfiltered channel energy of the first frame.

4.4.3.3 Channel SNR Estimator

Estimate the quantized channel SNR indices as:

$$\sigma_q(i) = \max\left\{0, \min\left\{89, \text{round}\left\{10\log_{10}\left(\frac{E_{ch}(m,i)}{E_n(m,i)}\right)/0.375\right\}\right\}\right\}; \quad 0 \le i < N_c, \qquad (4.4.3\text{-}8)$$

where $\mathbf{E}_n(m)$ is the current channel noise energy estimate (see 4.4.3.13), and the values of $\{\sigma_q\}$ are constrained to be between 0 and 89, inclusive. In addition, the WB NS requires the unquantized SNR values, expressed by:

$$\sigma_{unq}(i) = 10\log_{10}\left(\frac{E_{ch}(m,i)}{E_n(m,i)}\right); \quad 0 \le i < N_c, \qquad (4.4.3\text{-}9)$$

which can be saved as an intermediate step in Equation 4.4.3-8 with no computational overhead.

4.4.3.4 Voice Metric Calculation

Next, calculate the sum of voice metrics as:

$$v(m) = \sum_{i=0}^{N_c-1} V\left(\sigma_q(i)\right), \qquad (4.4.3\text{-}10)$$

where $V(k)$ is the kth value of the 90 element voice metric table $\mathbf{V}$, that is defined as:

$\mathbf{V}$ = { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 7, 7, 7, 8, 8, 9, 9, 10, 10, 11, 12, 12, 13, 13, 14, 15, 15, 16, 17, 17, 18, 19, 20, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50 }.

4.4.3.5 Band SNRs for Rate Determination Algorithm

The SNR estimates curr_ns_snr[0] and curr_ns_snr[1] are required for the Speech Mode Decision Algorithm in 4.8.3 for SO 68, SO 70, and SO 73. These estimates are given as:

$$\text{curr\_ns\_snr}[0] = 10\log_{10}\left(\frac{\sum_{i=0}^{K_1-1} E_{ch}(m,i)}{\sum_{i=0}^{K_1-1} E_n(m,i)}\right), \qquad (4.4.3\text{-}11)$$

and

$$\text{curr\_ns\_snr}[1] = 10\log_{10}\left(\frac{\sum_{i=K_1}^{K_2-1} E_{ch}(m,i)}{\sum_{i=K_1}^{K_2-1} E_n(m,i)}\right), \qquad (4.4.3\text{-}12)$$

where $K_1 = 11$, $K_2 = N_c$ for NB NS, and $K_1 = 33$, $K_2 = 50$ for WB NS.

4.4.3.6 Spectral Deviation Estimator

Calculate the estimated log power spectrum as:

$$E_{dB}(m,i) = 10 \log_{10}\left(E_{ch}(m,i)\right) \; ; \qquad 0 \le i < N_c \; . \tag{4.4.3-13}$$

Then, calculate the estimated spectral deviation between the current power spectrum and the average long-term power spectral estimate:

$$\Delta_E(m) = \sum_{i=0}^{N_c-1} \left| E_{dB}(m,i) - \bar{E}_{dB}(m,i) \right|, \tag{4.4.3-14}$$

where $\bar{\mathbf{E}}_{dB}(m)$ is the average long-term power spectral estimate calculated during the previous frame, as defined in Equation 4.4.3-19. The initial value of $\bar{E}_{dB}(m)$, however, is defined to be the estimated log power spectrum of frame 1, or:

$$\bar{E}_{dB}(m) = E_{dB}(m); \qquad \text{m=1}. \tag{4.4.3-15}$$

Calculate the total channel energy estimate, $E_{tot}(m)$, for the current frame, m, according to the following:

$$E_{tot}(m) = 10 \log_{10}\left(\sum_{i=0}^{N_c-1} E_{ch}(m,i) \right). \tag{4.4.3-16}$$

Calculate the exponential windowing factor, $\alpha(m)$, as a function of total channel energy, $E_{tot}(m)$, as:

$$\alpha(m) = \alpha_H - \left(\frac{\alpha_H - \alpha_L}{E_H - E_L} \right)\left(E_H - E_{tot}(m) \right), \tag{4.4.3-17}$$

and then limit the result to be between α_H and α_L by

$$\alpha(m) = \max\left\{ \alpha_L, \min\left\{ \alpha_H, \alpha(m) \right\} \right\}, \tag{4.4.3-18}$$

where E_H and E_L are the energy endpoints (in dB) for the linear interpolation of $E_{tot}(m)$, that is transformed to $\alpha(m)$ which has the limits $\alpha_L \le \alpha(m) \le \alpha_H$. The values of these constants are defined as: $E_H = 50$, $E_L = 30$, $\alpha_H = 0.99$, $\alpha_L = 0.50$. As an example, a signal with relative energy of 40 dB would use an exponential windowing factor of $\alpha(m) = 0.745$ for the following calculation.

Update the average long-term power spectral estimate for the next frame by:

$$\bar{E}_{dB}(m+1,i) = \alpha(m)\bar{E}_{dB}(m,i) + \left(1 - \alpha(m)\right)E_{dB}(m,i); \qquad 0 \le i < N_c, \tag{4.4.3-19}$$

where all the variables are previously defined.

4.4.3.7 Peak-to-Average Ratio

For the WB NS only, the Peak-to-Average Ratio calculation is used to detect tonal input signals above 220 Hz, and then inhibit the update of the background noise estimate to avoid attenuating these types of inputs. The Peak-to-Average ratio is given as:

$$\phi(m) = 10 \log_{10}\left(\max\{E_{ch}(m,i)\} \right) - E_{tot}; \qquad 6 \le i < N_c, \tag{4.4.3-20}$$

1 ## **4.4.3.8 Background Noise Update Decision**

2 The following logic, as shown in pseudo-code, demonstrates how the noise estimate update decision is ultimately made:

3   /* Normal update logic */

4   *update_flag* = FALSE

5   if ($v(m) \leq$ UPDATE_THLD) {

6   *update_flag* = TRUE

7   *update_cnt* = 0

8   }

9   /* Forced update logic */

10   else if (($E_{tot}(m) >$ NOISE_FLOOR_DB) and ($\Delta_E(m) <$ DEV_THLD)) {

11   if (WB_NS) {

12   /* Increment update_cnt only if low long-term prediction gain and no sine waves */

13   if ($\beta < 0.4$ and $\phi(m) < 10$)

14   *update_cnt* = *update_cnt* + 1

15   } else

16   *update_cnt* = *update_cnt* + 1

17   if (*update_cnt* $\geq$ UPDATE_CNT_THLD)

18   *update_flag* = TRUE

19   }

20   /* "Hysteresis" logic to prevent long-term creeping of update_cnt */

21   if (*update_cnt* == *last_update_cnt*)

22   *hyster_cnt* = *hyster_cnt* + 1

23   else

24   *hyster_cnt* = 0

25   *last_update_cnt* = *update_cnt*

26   if (*hyster_cnt* > HYSTER_CNT_THLD)

27   *update_cnt* = 0

28 The values of the previously used constants are UPDATE_THLD = 35, NOISE_FLOOR_DB = $10\log_{10}(E_{floor})$ (see
29 4.1.2.8), DEV_THLD = $28N_c/16$, UPDATE_CNT_THLD = 50, HYSTER_CNT_THLD = 6.

30 ## **4.4.3.9 SNR Estimate Modification**

31 Next, determine whether the channel SNR modification should take place, then proceed to modify the appropriate SNR
32 indices:

```
1          /* Set or reset modify flag */

2          index_cnt = 0

3          for ( i = NM  to  Nc - 1 step 1 ) {

4               if ( σq(i) ≥ INDEX_THLD )

5                    index_cnt = index_cnt + 1

6               }

7          if ( index_cnt < INDEX_CNT_THLD )

8               modify_flag = TRUE

9          else

10              modify_flag = FALSE

11
```

12 /* Modify the SNR indices to get $\left\{\sigma'_q\right\}$ */

```
13         if ( modify_flag == TRUE )

14              for ( i = 0 to  Nc - 1 step 1 )

15                   if (( v(m) ≤ METRIC_THLD ) or (σq(i) ≤ SETBACK_THLD ))
```

16 $$\sigma'_q(i) = 1$$

```
17                   else
```

18 $$\sigma'_q(i) = \sigma_q(i)$$

```
19         else
```

20 $$\left\{\sigma'_q\right\} = \left\{\sigma_q\right\}$$

21 /* Limit $\left\{\sigma''_q(i)\right\}$ to SNR threshold σ_{th} */

22 for ($i = 0$ to N_c - 1 step 1)

23 if $\left(\sigma'_q(i) < \sigma_{th}\right)$

24 $$\sigma''_q(i) = \sigma_{th}$$

```
25         else
```

26 $$\sigma''_q(i) = \sigma'_q(i)$$

27

28 For NB NS, the previous constants and thresholds are given to be: $N_M = 5$, INDEX_THLD = 12, INDEX_CNT_THLD =

29 5, METRIC_THLD = 45, SETBACK_THLD = 12, $\sigma_{th} = 6$. For WB NS, these are given as: $N_M = 12$, INDEX_THLD =

30 12, INDEX_CNT_THLD = 20, METRIC_THLD = 172, SETBACK_THLD = 12, $\sigma_{th} = 6$.

4.4.3.10 SNR Variability Estimator

For WB NS only, the SNR Variability Estimator works in conjunction with the Channel Gain Computation block to increase the SNR gain threshold for non-stationary background noises such as "office babble" and some kinds of street noise. The first step in this process is to estimate the instantaneous average channel SNR from the unquantized channel SNRs:

$$SNR_{inst} = \frac{1}{N_c - 5} \sum_{i=5}^{N_c - 1} \sigma_{unq}(i) .$$
(4.4.3-21)

Then, the negative SNR variability is computed based on negative values of the instantaneous SNR. It is presumed that a negative SNR can only occur as a result of fluctuating background noise, and not from the presence of voice. Therefore, the SNR threshold bias factor σ_{bias} is derived by calculating the negative SNR variability $\psi(m)$ as:

$$\psi(m) = \begin{cases} 0.99\psi(m-1) + 0.01 SNR_{inst}^2 & ; SNR_{inst} < 0 \\ \psi(m-1) & ; otherwise \end{cases} .$$
(4.4.3-22)

The variability factor is initially set to 0.375 when the frame count is less than or equal to four ($m \leq 4$). The SNR threshold bias factor σ_{bias} is then calculated as:

$$\sigma_{bias} = \min\{40(\psi(m) - 0.75), 16\} .$$
(4.4.3-23)

4.4.3.11 Channel Gain Computation

Compute the overall gain factor for the current frame as:

$$\gamma_n = \max\left\{\gamma_{min}, -10\log_{10}\left\{\frac{1}{E_{floor}} \sum_{i=0}^{N_c - 1} E_n(m, i)\right\}\right\},$$
(4.4.3-24)

where $\gamma_{min} = -13$ is the minimum overall gain, $E_{floor} = 1$ is the noise floor energy, and $E_n(m)$ is the estimated noise spectrum (see 4.4.3.13) calculated during the previous frame.

Next, calculate channel gains (in dB) as:

$$\gamma_{dB}(i) = \mu_g\left(\sigma_q''(i) - \sigma_{th}\right) + \gamma_n; \qquad 0 \leq i < N_c,$$
(4.4.3-25)

where μ_g is the gain slope, and σ_{th} is the SNR gain threshold. For NB NS, $\sigma_{th} = 6$ and $\mu_g = 0.39$, and for WB NS, $\mu_g = 0.50$ and the value of the SNR gain threshold is based on the SNR variability estimate as:

$$\sigma_{th} = 6 + \sigma_{bias} .$$
(4.4.3-26)

Then convert to linear channel gains using the following expression:

$$\gamma_{ch}(i) = \min\left\{1, 10^{\gamma_{dB}(i)/20}\right\}; \qquad 0 \leq i < N_c,$$
(4.4.3-27)

4.4.3.12 Frequency Domain Filtering

Now, apply the channel gains to the transformed input signal $G(k)$:

$$H(k) = \begin{cases} \gamma_{ch}(i)G(k) & ; f_L(i) \leq k \leq f_H(i), \ 0 \leq i < N_c, \\ G(k) & ; 0 \leq k < f_L(0), \ f_H(N_c - 1) < k \leq M/2, \end{cases}$$
(4.4.3-28)

where the bottom part of the equation represents the frequencies that are not altered by the channel gains. For the WB NS, the channel gains are applied in such a manner that the lowest and highest frequencies are modified by the closest adjacent frequency gain, or:

$$H(k) = \begin{cases} \gamma_{ch}(i)G(k) & ; f_L(i) \le k \le f_H(i), 0 \le i < N_c \\ \gamma_{ch}(0)G(k) & ; 0 \le k < f_L(0) \\ \gamma_{ch}(N_c - 1)G(k) & ; f_H(N_c - 1) < k \le M/2 \end{cases}$$
(4.4.3-29)

.

It is also required that the magnitude of $H(k)$ be even, and the phase be odd, so that the following condition is also imposed:

$$H(M-k) = H^*(k); \quad 0 < k < M/2,$$
(4.4.3-30)

where $*$ denotes complex conjugate. This guarantees that the imaginary part of the inverse DFT of $H(k)$ will be zero (in Equation 4.4.3-34).

4.4.3.13 Background Noise Estimate Update

If (and only if) the update flag is set (*update_flag* == TRUE), then update the channel noise estimate for the next frame by:

$$E_n(m+1,i) = \max\left\{ E_{\min}, \alpha_n E_n(m,i) + (1-\alpha_n)E_{ch}(m,i) \right\}; \quad 0 \le i < N_c,$$
(4.4.3-31)

where $E_{\min} = 0.0625$ is the minimum allowable channel energy, and $\alpha_n = 0.9$ is the channel noise smoothing factor. For the NB NS, the channel noise estimate shall be initialized for each of the first four frames to the estimated channel energy, i.e.:

$$E_n(m,i) = \max\left\{ E_{init}, E_{ch}(m,i) \right\} \quad ; 1 \le m \le 4, 0 \le i < N_c,$$
(4.4.3-32)

where $E_{init} = 16$ is the minimum allowable channel noise initialization energy. For the WB NS, the channel noise estimate shall be initialized as:

$$E_n(m,i) = \begin{cases} 0.5 \max\{E_{init}, E_{ch}(m,i)\} & ; m = 1 \\ 0.5 E_n(m-1,i) + 0.5 \max\{E_{init}, E_{ch}(m,i)\} & ; 1 < m \le 16 \end{cases},$$
(4.4.3-33)

where $E_{init} = 0.0625$ is the minimum allowable channel noise initialization energy.

4.4.3.14 Time Domain Signal Reconstruction

Convert the filtered signal to the time domain using the inverse DFT:

$$h(m,n) = \frac{1}{2}\sum_{k=0}^{M-1} H(k)e^{j2\pi nk/M}; \quad 0 \le n < M \quad.$$
(4.4.3-34)

Complete the frequency domain filtering process by applying overlap-and-add. For NB NS:

$$h'(n) = \begin{cases} h(m,n) + h(m-1,n+L) & ; \quad 0 \le n < M - L, \\ h(m,n) & ; \quad M - L \le n < L. \end{cases}$$
(4.4.3-35)

and for WB NS ($L = D = 160$):

$$h'(n) = h(m,n)\sin\left(\pi(n+0.5)/2L\right) + h(m-1,n+L)\sin\left(\pi(n+L+0.5)/2L\right); \; 0 \le n < L$$
(4.4.3-36)

1 where $h(m-1, n+L)$ are the samples from the last part of the previous frame.

2 Finally, apply signal de-emphasis by:

3
$$s'(n+N_{offs}) = h'(n) + \zeta_d s'(n+N_{offs}-1); \quad 0 \le n < L \tag{4.4.3-37}$$

4 where $\zeta_d = 0.8$ is the de-emphasis factor, $\{s'(n)\}$ is the output buffer, and N_{offs} is the relative offset into the output buffer.
5 Since the 10 ms per frame Noise Suppression is performed twice per 20 ms speech frame, the output for NB NS
6 presented to Model Parameter Estimation comprises $s'(n)$; $160 \le n < 320$. Therefore, $N_{offs} = 160$ for the first 10 ms NS
7 frame and $N_{offs} = 240$ for the second 10 ms NS frame. The data in the buffer from the previous frame shall be time
8 shifted accordingly.

9 For the WB NS, the output comprises $s'(n)$; $160 \le n < 480$. Therefore, $N_{offs} = 160$ for the first 10 ms frame and
10 $N_{offs} = 320$ for the second 10 ms frame.

11 ## 4.4.3.15 Zero Delay Overlap/Add

12 For the second WB NS frame only, the overlapped portion of the filtered frame $h(m, n)$ is combined with a gain scaled
13 version of the overlapped portion of the input signal $d(m, n)$ to form an auxiliary signal $h''(n)$, given as:

14
$$h''(n) = h(m,n+L)\sin^2\left(\pi(n+0.5)/2L\right) + \gamma_{max}d(m,n+L)\cos^2\left(\pi(n+0.5)/2L\right); \quad 0 \le n < L \tag{4.4.3-38}$$

15 where γ_{max} is the maximum noise suppression gain applied to the spectrum of the current frame in Equation 4.4.3-27.,
16 or:

17
$$\gamma_{max} = \max\{\gamma_{ch}(i)\}; \quad 0 \le i < N_c. \tag{4.4.3-39}$$

18 The auxiliary signal $s_{aux}(n)$ is then generated by applying signal deemphasis according to:

19
$$s_{aux}(n) = h''(n) + \zeta_d s_{aux}(n-1); \quad 0 \le n < L \tag{4.4.3-40}$$

20 where the deemphasis memory $s_{aux}(-1)$ is initialized to the last sample of the primary NS output
21 $s_{aux}(-1) = s'(N_{offs}+L-1)$.

22 This process effectively produces a temporary auxiliary signal that can be used to extend the noise suppressed frame into
23 the future for the purposes of spectral and pitch analysis. That is, the auxiliary signal is used by the Model Parameter
24 Estimation (sec 4.6) as the "lookahead" part of the analysis frame, and is discarded on the following frame as the proper
25 noise suppressed frame becomes available. The purpose of this is twofold. First, the net effect on end-to-end delay of
26 the NS algorithm is zero, and second, an increased overlap/add of 10 ms (coincidentally equal to the lookahead delay)
27 allows for better spectral estimation and frequency domain filtering operations within the NS algorithm.

1 ## 4.5 Analysis Filterbank

2
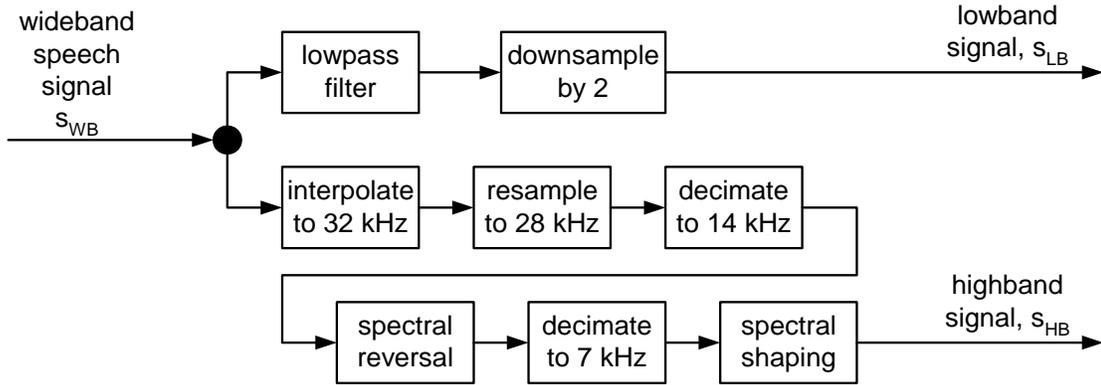
3 **Figure 4.5-1 Analysis Filterbank**

4 ### 4.5.1 Generation of the Lowband Signal, s_{LB}

5 The lowpass filter and the downsample-by-2 can be realized as a polyphase implementation. The wideband speech signal
6 s_{WB} is first separate into odd time indexed and even time indexed subsequences. These subsequences are filtered by two
7 13th order FIR filters and the outputs of these filters are added to derive the lowband signal, s_{LB}.

8 ### 4.5.2 Generation of the Highband Signal, s_{HB}

9 ### 4.5.2.1 Interpolation to 32 kHz

10 The 16 kHz signal s_{WB} is first upsampled to 32 kHz, $s_{32}[n]$, through the use of a 2-section polyphase implementation.

11 The even components of $s_{32}[n]$, for n=0, 2, 4, 6 etc are obtained by filtering $s_{WB}[n/2]$ through an allpass filter whose
12 transfer function is given by

$$H_{up2,0} = \left(\frac{a_{up2,0,0} + z^{-1}}{1 + a_{up2,0,0}z^{-1}} \right)\left(\frac{a_{up2,0,1} + z^{-1}}{1 + a_{up2,0,1}z^{-1}} \right)$$

13

14 And the odd components of $s_{32}[n]$, for n=1, 3, 5, 7 etc are obtained by filtering $s_{WB}[n/2]$ through an allpass filter whose
15 transfer function is given by

$$H_{up2,1} = \left(\frac{a_{up2,1,0} + z^{-1}}{1 + a_{up2,1,0}z^{-1}} \right)\left(\frac{a_{up2,1,1} + z^{-1}}{1 + a_{up2,1,1}z^{-1}} \right)$$

16

17 ### 4.5.2.2 Resampling to 28 kHz

18 $s_{32}[n]$ is resampled by a factor of 7/8 to obtain a 28 kHz signal, $s_{28}[n]$. This is obtained using a polyphase interpolation
19 according to

$$s_{28}(7n+j) = \sum_{k=0}^{9} h_{32to28}(j,k)s_{32}(8n+j); \qquad n = 0,1,2...,320/8-1, j = 0,1,2,...,6$$

20

1 ## 4.5.2.3 Decimation to 14 kHz

2 The 28 kHz signal s_{28} is decimated to 14 kHz, $s_{14}[n]$, through the use of a 3-section polyphase implementation.

3 The even components of $s_{28}[n]$, for n=0, 2, 4, 6 etc are filtered through an allpass filter whose transfer function is given

4 by

$$H_{down2,0} = \left(\frac{a_{down2,0,0} + z^{-1}}{1 + a_{down2,0,0}z^{-1}} \right) \left(\frac{a_{down2,0,1} + z^{-1}}{1 + a_{down2,0,1}z^{-1}} \right) \left(\frac{a_{down2,0,2} + z^{-1}}{1 + a_{down2,0,2}z^{-1}} \right)$$

5

6 And the odd components of $s_{28}[n]$, for n=1, 3, 5, 7 etc are filtered through an allpass filter whose transfer function is

7 given by

$$H_{down2,1} = \left(\frac{a_{down2,1,0} + z^{-1}}{1 + a_{down2,1,0}z^{-1}} \right) \left(\frac{a_{down2,1,1} + z^{-1}}{1 + a_{down2,1,1}z^{-1}} \right) \left(\frac{a_{down2,1,2} + z^{-1}}{1 + a_{down2,1,2}z^{-1}} \right)$$

8

9 The outputs of these two polyphase components are averaged to yield the 14 kHz sampled signal, s_{14}.

10 ## 4.5.2.4 Spectral Reversal

11 The spectrum of $\hat{s}_{14}$ is reversed by multiplying the odd components by -1 to yield a signal s_{14}.

12 ## 4.5.2.5 Decimation to 7 kHz

13 The 14 kHz spectrally reversed signal s_{14} is decimated to 7 kHz, $s_7[n]$, through the use of a 3-section polyphase

14 implementation.

15 The even components of $s_{14}[n]$, for n=0, 2, 4, 6 etc are filtered through an allpass filter whose transfer function is given

16 by

$$H_{down2,0} = \left(\frac{a_{down2,0,0} + z^{-1}}{1 + a_{down2,0,0}z^{-1}} \right) \left(\frac{a_{down2,0,1} + z^{-1}}{1 + a_{down2,0,1}z^{-1}} \right) \left(\frac{a_{down2,0,2} + z^{-1}}{1 + a_{down2,0,2}z^{-1}} \right)$$

17

18 And the odd components of $s_{14}[n]$, for n=1, 3, 5, 7 etc are filtered through an allpass filter whose transfer function is

19 given by

$$H_{down2,1} = \left(\frac{a_{down2,1,0} + z^{-1}}{1 + a_{down2,1,0}z^{-1}} \right) \left(\frac{a_{down2,1,1} + z^{-1}}{1 + a_{down2,1,1}z^{-1}} \right) \left(\frac{a_{down2,1,2} + z^{-1}}{1 + a_{down2,1,2}z^{-1}} \right)$$

20

21 The outputs of these two polyphase components are added to yield the 7 kHz sampled signal, s_7.

22 ## 4.5.2.6 Spectral Shaping

23 The 7 kHz signal, s_7, is the shaped using a 1st order IIR shaping filter of the form

$$H_{shaping}(z) = 0.95 \frac{1 + z^{-1}}{1 - 0.9z^{-1}}$$

24

25 to obtain the highband signal, s_{HB}.

4.6 Model Parameter Estimation

Model Parameter Estimation comprises the linear predictive analysis, residual calculation and long-term prediction that must be performed for each frame, independent of the rate decision.

Inputs: The inputs to model parameter estimation are:

- The current 20 ms frame number, m

- The pre-processed speech output vector, $\{s'(n)\}$

- The unquantized line spectral pairs, $\Omega(m-1)$, calculated in the previous frame

- The LPC prediction gain, $\gamma_{lpc}(m-1)$, calculated in the previous frame

The pre-processed speech buffer, $\{s'(n)\}$, contains 320 pre-processed speech samples, i.e., $0 \le n < 320$. Samples 0 through 79 are "lookback" from the previous frame, samples 80 through 239 are the current frame, and samples 240 through 319 are "lookahead" to the next frame. The last 160 samples from the preprocessing modules, then, constitute 80 samples for the last half of the "current" frame and 80 samples "lookahead" to the next frame.

Outputs: The outputs of model parameter estimation are:

- The unquantized linear predictive coefficients for the current frame, $\{a\}$

- The unquantized LSPs for the current frame, $\Omega(m)$

- The LPC prediction gain, $\gamma_{lpc}(m)$

- The prediction residual, $\{\varepsilon(n)\}$

- The long-term pitch delay estimate, τ

- The long-term prediction gain, β

- The spectral transition indicator, *LPCFLAG*

- The bandwidth expanded correlation coefficients, $\boldsymbol{R}_w$

Processing: The model parameter estimation module performs three major functions:

- Calculation of the formant filter parameters and prediction gain

- Generation of the short-term prediction residual signal

- Calculation of the delay estimate and long-term prediction gain

In addition to these functions, this module also generates a spectral transition indicator, which is used to improve channel impairment performance. The following sections describe each of the major model parameter estimation functions in detail.

4.6.1 Formant Filter Parameter Calculation

Inputs: The inputs to the formant filter parameter calculation are:

- The pre-processed speech input signal, $\{s'(n)\}$

- The LPC prediction gain from the previous frame, $\gamma_{lpc}(m-1)$

Outputs: The outputs of the formant filter parameter calculation are:

- The unquantized LPCs for the current frame $\{a\}$

- The unquantized LSPs for the current frame, $\Omega(m)$

- The LPC prediction gain, $\gamma_{lpc}(m)$

- The impulse response of the formant filter, $h(n)$

- The spectral transition indicator, *LPCFLAG*

Initialization: The LPC prediction gain, $\gamma_{lpc}(m) = 1$, for $m = 0$

Processing: The formant filter parameters are calculated from the pre-processed speech buffer, $\{s'(n)\}$, using the autocorrelation method and Durbin's Recursion. The details of the computation shall be executed as described in 4.6.1.1 through 4.6.1.3.

4.6.1.1 Direct Form LPC Parameter Calculation

For Service Option 3, the pre-processed speech buffer is windowed using a Hamming window centered at the end of the current frame, i.e., on sample 239. The equation for the Hamming window is:

$$W_H(k) = 0.54 - 0.46 \cos\left(\frac{2\pi}{160}(k - 160)\right); \quad 160 \le k < 320, \tag{4.6.1-1}$$

and the expression for the windowed speech is:

$$s'_H(k - 160) = W_H(k)s'(k); \quad 160 \le k < 320. \tag{4.6.1-2}$$

The first 17 terms of the autocorrelation function of $\{s_H'(n)\}$, **R**, are calculated directly using:

$$R(k) = \sum_{i=0}^{159-k} s'_H(i)s'_H(i + k); \quad 0 \le k < 16. \tag{4.6.1-3}$$

For Service Options 68, 70, and 73, the pre-processed speech buffer is windowed using the assymetric window given by

$$W_H(k) = \frac{\left(0.5 - 0.5\cos\left(\frac{2\pi}{320}k\right)\right)e^{\frac{8.4k}{320}}}{286.5}; \quad 0 \le k < 320 \tag{4.6.1-4}$$

and the expression for the windowed speech is:

$$s'_H(k) = W_H(k)s'(k); \quad 0 \le k < 320 \tag{4.6.1-5}$$

The first 17 terms of the autocorrelation function of $\{s_H'(n)\}$, **R**, are calculated directly using:

$$R(k) = \sum_{i=0}^{320-k} s'_H(k)s'_H(i + k); \quad 0 \le k < 16 \tag{4.6.1-6}$$

The first 11 are used for LPC analysis, while all 17 terms are used by the Rate Determination Algorithm (see 4.3). The autocorrelation terms are then spectrally smoothed by windowing as follows:

$$R_w(k) = \begin{cases} 1.00003 R(k); & k = 0, \\ \exp\left[-\frac{1}{2}\left(\frac{40\pi k}{8000}\right)^2\right]R(k); & 1 \le k \le 16. \end{cases} \tag{4.6.1-7}$$

The unquantized, unweighted LPCs, {α} are then calculated from $\mathbf{R}_W$ using Durbin's Recursion[†], as follows:

{

$$E^{(0)} = R_w(0)$$

$$i = 1$$

while (i ≤ 10)

{

$$k_i = \frac{1}{E^{(i-1)}}\left[R_w(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R_w(i-j) \right]$$

$$\alpha_i^{(i)} = k_w$$

$$j = 1$$

while (j < i)

{

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}$$

$$j = j + 1$$

}

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}$$

$$i = i + 1$$

}

}

The LPC coefficients are $\alpha_j^{(10)}$, for $1 \le j \le 10$, or {α} = { α_1, α_2, ..., α_{10} }.

In SO 68, two additional parameters called "stoporder15", and "stoporder30" are monitored during the Levinson Durbin recursion. These correspond to the iteration numbers where the prediction gains first cross 15 dB and 30 dB respectively, and are defined by.

Stoporder15 = iteration index when $\dfrac{E^i}{E^0} > 10^{\frac{15}{10}}$, and

Stoporder30 = iteration index when $\dfrac{E^i}{E^0} > 10^{\frac{30}{10}}$.

If Stoporder30 ≤ 4, the frame is encoded using Full-Rate CELP.

[†] See Rabiner, L. R. and Schafer, R. W., *Digital Processing of Speech Signals,* (New Jersey: Prentice-Hall Inc, 1978), pp. 411-412. The superscripts in parentheses represent the stage of Durbin's recursion. For example $\alpha_j^{(i)}$ refers to α_j at the *i*th stage.

If Stoporder15 ≤ 4, and the frame is chosen for Half-Rate CELP coding by the Voice Activity Detection function, then the frame is encoded using Full-Rate CELP.

4.6.1.2 Generation of Spectral Transition Indicator (*LPCFLAG*)

For SO 3 only, the impulse response, $\{h_{raw}\}$, of the unweighted, unquantized formant filter is calculated to 54 terms. The unweighted, unquantized formant filter is given by:

$$\frac{1}{A_{raw}(z)} = \frac{1}{1 - \sum_{k=1}^{10} \alpha_k z^{-k}}, \tag{4.6.1-8}$$

where k is the LPC index. The energy of the raw impulse response is then calculated as an estimate of the LPC prediction gain. This is given by:

$$\gamma_{lpc}(m) = \sum_{k=0}^{53} h_{raw}^2(k). \tag{4.6.1-9}$$

If the ratio $\gamma_{lpc}(m)/\gamma_{lpc}(m\text{-}1) > 10$ (where $\gamma_{lpc}(m\text{-}1)$ is the LPC prediction gain from the previous frame), then the spectral transition indicator, *LPCFLAG*, is set to 1, indicating that a large spectral transition has occurred. Otherwise, *LPCFLAG* is set to 0.

4.6.1.3 Direct Form LPC to LSP Conversion

The raw LPCs are bandwidth expanded using an exponential window:

$$\alpha_k = (0.994)^k \alpha_k \quad ; 0 < k \le 10 \tag{4.6.1-10}$$

The unquantized, bandwidth expanded LPCs are then converted to LSPs. Here, $A(z)$ is given by:

$$A(z) = 1 - \sum_{k=1}^{10} \alpha_k z^{-k}, \tag{4.6.1-11}$$

where $\{a\}$ are the bandwidth expanded LPC coefficients.

Next, define $P_A(z)$ and $Q_A(z)$ as follows:

$$P_A(z) = A(z) + z^{-11} A(z^{-1}) = 1 + \sum_{i=1}^{5} p_i z^{-i} + \sum_{i=6}^{10} p_{11-i} z^{-i} + z^{-11} \tag{4.6.1-12}$$

$$Q_A(z) = A(z) - z^{-11} A(z^{-1}) = 1 + \sum_{i=1}^{5} q_i z^{-i} - \sum_{i=6}^{10} q_{11-i} z^{-i} - z^{-11}, \tag{4.6.1-13}$$

where $p_i = -a_i - a_{11-i}$; $1 \le i \le 5$, and $q_i = -a_i + a_{11-i}$; $1 \le i \le 5$. The LSP frequencies are the ten roots which exist between $\omega = 0$ and $\omega = 0.5$ in the following two equations:

$$P'(\omega) = \cos 5(2\pi\omega) + p'_1 \cos 4(2\pi\omega) + \ldots + p'_4 \cos (2\pi\omega) + p'_5/2, \tag{4.6.1-14}$$

$$Q'(\omega) = \cos 5(2\pi\omega) + q'_1 \cos 4(2\pi\omega) + \ldots + q'_4 \cos (2\pi\omega) + q'_5/2, \tag{4.6.1-15}$$

where the p' and q' values are computed recursively as follows from the p and q values:

$$p'_0 = q'_0 = 1, \tag{4.6.1-16}$$

$$p'_i = p_i - p'_{i-1} \qquad 1 \le i \le 5, \tag{4.6.1-17}$$

1          $q'_i = q_i + q'_{i-1}$ $1 \leq i \leq 5$.   (4.6.1-18)

2 Since the formant synthesis (LPC) filter is stable, the roots of the two functions alternate; the smallest root, ω_1 is the
3 lowest root of $P'(\omega)$, the next smallest root, ω_2 is the lowest root of $Q'(\omega)$, etc. Thus, ω_1, ω_3, ω_5, ω_7, and ω_9 are the roots
4 of $P'(\omega)$, and ω_2, ω_4, ω_6, ω_8, and ω_{10} are the roots of $Q'(\omega)$. The corresponding unquantized LSP parameter vector is
5 defined as:

6          $\Omega(m) = \{ \omega_1, \omega_2, ..., \omega_{10} \}$.   (4.6.1-19)

7 ## 4.6.2 Generation of the Short-Term Prediction Residual

8 **Inputs:** The inputs to the short-term prediction residual calculation are:

9 • The unquantized LSPs from the previous frame, $\Omega(m\text{-}1)$

10 • The unquantized LSPs calculated for the current frame, $\Omega(m)$

11 • The pre-processed speech input vector, $\{s'(n)\}$

12 **Output:** The output of short-term prediction residual calculation is:

13 • The residual vector, $\{\varepsilon(n)\}$

14 This vector contains 320 elements, partitioned in the same way as the pre-processed speech input vector. The first 80
15 samples represent the last 10 ms of the previous frame; the next 160 samples represent the current 20 ms frame, and the
16 last 80 samples represent the 10 ms look-ahead.

17 **Initialization:** The unquantized LSPs, $\Omega(m,i) = 0.048i$; $m = 0$, $1 \leq i \leq 10$.

18 **Processing:** The short-term prediction residual signal, $\{\varepsilon(n)\}$, is generated by passing $\{s'(n)\}$ through the inverse filter
19 created by interpolating between $\Omega(m\text{-}1)$ and $\Omega(m)$. The $\{s'(n)\}$ vector is divided into five segments, and a different set
20 of interpolated LSPs is computed for each corresponding segment. The interpolated LSPs are converted to LPCs, and
21 the appropriate segment of $\{s'(n)\}$ is convolved with the resulting filter to generate the corresponding samples of $\{\varepsilon(n)\}$.
22 The initial state of the inverse filter shall be zero for each frame, m, i.e., this is a zero state filter. Also, throughout the
23 following discussion, variables with a "dot" are implied to be interpolated, e.g., $\dot{\Omega}(m,k)$.

24 ## 4.6.2.1 LSP Interpolation

25 For each segment k, the unquantized, interpolated LSP vector is:

26          $\dot{\Omega}(m,k) = (1 - \mu_k)\Omega(m-1) + \mu_k\Omega(m)$ $; 0 \leq k \leq 4$ (4.6.2-1)

27 where the interpolator constants, $\{\mu\}$, and their corresponding sets of sample indices for each segment of $\{s'(n)\}$ are
28 given in Table 4.6.2.1-1.

29

30 **Table 4.6.2.1-1 LSP Interpolation Constants**

k	Segment start sample	Segment end sample	μ_k
0	0	79	0.0
1	80	132	0.1667
2	133	185	0.5

k	Segment start sample	Segment end sample	μ_k
3	186	239	0.8333
4	240	319	1.0

4.6.2.2 LSP to Direct Form LPC Conversion

For each Segment, $0 \le k \le 4$, the following process is executed. First, $\dot{P}_A(z)$ and $\dot{Q}_A(z)$ are computed from a set of interpolated LSP frequencies, $\dot{\Omega}(m,k)$, as follows:

$$\dot{P}_A(z) = \left(1+z^{-1}\right)\prod_{j=1}^{5}\left(1-2z^{-1}\cos\left(2\pi\,\dot{\omega}_{(2j-1)}\right)+z^{-2}\right), \tag{4.6.2-2}$$

$$\dot{Q}_A(z) = \left(1-z^{-1}\right)\prod_{j=1}^{5}\left(1-2z^{-1}\cos\left(2\pi\,\dot{\omega}_{(2j)}\right)+z^{-2}\right), \tag{4.6.2-3}$$

where $\left\{\dot{\omega}_1,\dot{\omega}_2,...,\dot{\omega}_{10}\right\} = \dot{\Omega}(m,k)$, and the k is implied. These equations can also be expressed as:

$$\dot{P}_A(z) = 1 + \sum_{i=1}^{5}\dot{p}_i\,z^{-i} + \sum_{i=6}^{10}\dot{p}_{11-i}\,z^{-i} + z^{-11}, \tag{4.6.2-4}$$

$$\dot{Q}_A(z) = 1 + \sum_{i=1}^{5}\dot{q}_i\,z^{-i} - \sum_{i=6}^{10}\dot{q}_{11-i}\,z^{-i} - z^{-11}. \tag{4.6.2-5}$$

Then the LPC coefficients are computed from the coefficients of $\dot{P}_A(z)$ and $\dot{Q}_A(z)$ as follows:

$$\dot{A}(z) = \frac{\dot{P}_A(z)+\dot{Q}_A(z)}{2}, \tag{4.6.2-6}$$

$$= 1 + \sum_{i=1}^{5}\frac{(\dot{p}_i+\dot{q}_i)}{2}z^{-i} + \sum_{i=6}^{10}\frac{(\dot{p}_{11-i}+\dot{q}_{11-i})}{2}z^{-i}, \tag{4.6.2-7}$$

$$\dot{A}(z) = 1 - \sum_{i=1}^{10}\dot{a}_i\,z^{-i}. \tag{4.6.2-8}$$

So, the interpolated LPC coefficients for Segment k of the current frame are given by:

$$\dot{a}_i(k) = \begin{cases} -\dfrac{\dot{p}_i+\dot{q}_i}{2} & ;\, 1\le i \le 5 \\[2em] -\dfrac{\dot{p}_{11-i}+\dot{q}_{11-i}}{2} & ;\, 6\le i \le 10 \end{cases} \tag{4.6.2-9}$$

4.6.2.3 Generation of Residual Samples

The short-term prediction residual, $\varepsilon(n)$, is generated by passing $s'(n)$ through the inverse filter using the appropriate LPCs:

1

$$\varepsilon(n) = s'(n) - \sum_{i=1}^{10} a_i(k)s'(n-i), \qquad\qquad (4.6.2\text{-}10)$$

2   where the value for k in $a_i(k)$ is determined by the segment in which n lies as determined by the starting and ending

3   sample numbers indicated in Table 4.6.2.1-1. All values of $s'(n)$ for $n < 0$ are zero.

4 ## 4.6.3 Calculation of the Delay Estimate and Long-Term Prediction Gain

5   **Inputs:** The input to the delay estimate and long-term prediction gain calculation is:

6   • The short-term residual vector, $\{\varepsilon(n)\}$

7   **Outputs:** The outputs of the delay estimate and long-term prediction gain calculation are:

8   • The pitch delay estimate, τ

9   • The long-term prediction gain, β

10   **Initialization:** The values of all local state variables and buffers are set to zero at start-up.

11   **Processing:** The pitch delay shall be calculated using the method described in this section. The pitch delay is the delay

12   that maximizes the autocorrelation function of the short-term prediction residual signal, subject to certain constraints.

13   This calculation is carried out independently over two estimation windows. The first of these comprises the entire current

14   frame; the second comprises the second half of the current frame and the look-ahead. Rules are then applied to combine

15   the delay estimates and gains for the two estimation windows. The constrained search for the optimal delay in each

16   window shall be carried out as follows:

17   The processes in 4.6.3.1 through 4.6.3.8 shall be performed once for each of the two estimation windows of each frame.

18 ## 4.6.3.1 Non-Exhaustive Open Loop Delay Search

19   The residual signal, $\{\varepsilon(n)\}$, is filtered and decimated by a factor of four to generate the decimated residual signal,

20    $\{\varepsilon_d(n)\}$, by applying:

21

$$x(n) = \varepsilon(n + n_{start}) + 2.2875x(n-1) - 1.956x(n-2) + 0.5959x(n-3); \qquad 0 \le n < 80, \qquad (4.6.3\text{-}1)$$

22   then:

23

$$\varepsilon_d(n') = \varepsilon_d(n' + 20) \quad , \qquad 0 \le n' < 20, \qquad\qquad (4.6.3\text{-}2)$$

24   and finally:

25

$$\varepsilon_d(n') = x(4n' - 77) - 0.312[x(4n' - 78) + x(4n' - 79)] + x(4n' - 80), \qquad 20 \le n' < 40 \qquad (4.6.3\text{-}3)$$

26   where $\{x(n)\}$ is the decimator filter memory, $\{\varepsilon_d(n)\}$ is the decimated residual buffer, and n_{start} is 160 for the first

27   estimation window and 240 for the second estimation window. The autocorrelation function of this decimated residual

28   signal is generated using:

29

$$r(d) = \sum_{k=0}^{40-d} \varepsilon_d(k)\varepsilon_d(k+d) \qquad ; 5 \le d \le 30, \qquad\qquad (4.6.3\text{-}4)$$

30   and the delay, d_{max}, corresponding to the maximum positive correlation, r_{max}, is found.

4.6.3.2 Refinement of Open Loop Delay Estimate for SO 3

The following operations are then performed to determine whether to use d_{max} as the decimated delay estimate, or to use an estimate in the neighborhood of the smoothed delay estimate, $\tilde{\tau}$, instead:

$$\text{if } (\tilde{\tau} \neq 0 \text{ AND } |\tilde{\tau} - 4d_{\max}| > 2) \ \{$$

$$\text{if}(r'_{max} > 0.835 \ r_{max}) \ \{$$

$$d_{max} = d'_{max}$$

$$\}$$

$$\}$$

where d'_{max} is calculated as the delay corresponding to the maximum positive value of $\{r(d)\}$ in the range: $\max\{$ $5, \lfloor \tilde{\tau}/4 \rfloor - 2 \} \leq d \leq \min\{ 30, 4 + \max\{ 5, \lfloor \tilde{\tau}/4 \rfloor - 2 \} \}$, and $\tilde{\tau}$ is updated per Equation 4.4.3-11.

The optimal delay estimate, D_{max}, is calculated as the index corresponding to R_{max}, which is the maximum positive value of:

$$R(D) = \sum_{n=0}^{159-D} \varepsilon(n + n_0)\varepsilon(n + n_0 + D) \ ; \max\{20, 4d_{\max} - 3\} \leq D \leq \min\{120, 4d_{\max} + 3\}, \tag{4.6.3-5}$$

where $n_0 = 80$ and 160 for the respective first and second estimation windows.

4.6.3.3 Refinement of Open Loop Delay Estimate for SO 68, SO 70, and SO 73

The original residual, $\varepsilon(n)$, is then used to compute additional correlation around dmax

$$rr(d) = \sum_{k=0}^{160-d} \varepsilon(k)\varepsilon(k + d); \qquad M1 \leq d \leq M2, \tag{4.6.3-6}$$

where, $M1 = \max(20, d_{\max} - 3)$, and $M2 = \min(120, d_{\max} + 3)$.

and the delay, $dd_{\max}$, corresponding to the maximum positive correlation, $rr_{\max}$, is found, and the normalized correlation is:

$$\beta = \min\left(1.0, \max\left(0.0, \frac{rr_{\max}}{sqrt\left(\left(\sum_{k=0}^{160-dd_{\max}-1}(\varepsilon(k))^2\right)\left(\sum_{k=dd_{\max}}^{160-1}(\varepsilon(k))^2\right)\right)} \right)\right). \tag{4.6.3-7}$$

The following operations are then performed to determine whether to use $dd_{\max}$ as the decimated delay estimate, or to use an estimate in the neighborhood of the smoothed delay estimate, $\hat{\tau}$, instead:

Similar to above, an estimate of the delay dpmax within 6 samples of smoothed delay estimate $\hat{\tau}$, the corresponding maximum positive correlation, rpmax, and the normalized correlation, βp are determined.

If $\tilde{\tau} \neq 0 \ AND |\tilde{\tau} - dd_{max}| > 6$, and if the normalized correlation around the smoothed delay estimate is greater than a weighted version of the normalized correlation of the current frame, then the delay estimate around the smoothed delay

is used for the current frame. That is, if $\beta p > w \cdot \beta$ (where w=0.6 if ddmax $> \hat{\tau} + 6$, w=1.2 if $dd_{\max} > \hat{\tau}$ -6, w=10 otherwise) then $\beta = \beta p$, and $dd_{\max} = dp_{\max}$.

A further check for pitch halving is performed to further refine the delay estimate if $dd_{\max} > 40$. Similar to above, an estimate of the delay dhmax within 2 samples of smoothed delay estimate ddmax/2, the corresponding maximum positive correlation, rhmax, and the normalized correlation, βh are determined.

If $\beta h > 0.8 \cdot \beta$, then $\beta = \beta h$, and $dd_{\max} = dh_{\max}$.

4.6.3.4 Long-Term Prediction Gain Calculation

The energy of the undecimated residual is calculated as follows:

$$R_{\varepsilon}(d) = \sqrt{\sum_{i=0}^{159-D_{\max}} \varepsilon^2(i+n_0) \sum_{j=0}^{159-D_{\max}} \varepsilon^2(j+n_0 + D_{\max})} , \qquad (4.6.3\text{-}8)$$

from which the long-term prediction gain can be derived by:

$$\beta = \max\left\{0, \min\left\{1, \frac{R_{\max}}{R_{\varepsilon}(0)}\right\}\right\} . \qquad (4.6.3\text{-}9)$$

4.6.3.5 Smoothed Delay Estimate and LTP Gain for SO3

The following operations are performed to determine whether to replace the values of β and D_{max} calculated in 4.6.3.1 and 4.6.3.2 by values obtained in the neighborhood of the smoothed delay estimate, $\tilde{\tau}$:

  if($\tilde{\tau} > 0$) {

    if($(D_{\max} > \min\{120, \tilde{\tau} + 6\}$ AND $\beta' > 0.6\beta$) OR

       $(D_{\max} < \min\{20, \tilde{\tau} - 6\}$ AND $(\beta' > 1.2\beta)$) {

         $\beta = \beta'$

         $D_{max} = D'_{max}$

    }

  }

where D'_{max} is the index of R'_{max}, the maximum positive value of:

$$R'(D) = \sum_{n=0}^{159-D} \varepsilon(n+n_0)\varepsilon(n+n_0+D) \quad ; \max\{20, \tilde{\tau} - 6\} \le D \le \min\{120, \tilde{\tau} + 6\} \qquad (4.6.3\text{-}10)$$

and:

$$\beta' = \max\left\{0, \min\left\{1, \frac{R'_{\max}}{\sqrt{\sum_{i=0}^{159-D'_{\max}} \varepsilon^2(i+n_0) \sum_{j=0}^{159-D'_{\max}} \varepsilon^2(j+n_0 + D'_{\max})}}\right\}\right\} . \qquad (4.6.3\text{-}11)$$

The smoothed delay and long-term prediction gain estimates are updated as follows:

$$\tilde{\beta} = \begin{cases} \beta & ; \beta > 0.4, \\ 0.75\tilde{\beta} & ; \text{otherwise,} \end{cases}$$

(4.6.3-12)

then:

$$\tilde{\tau} = \begin{cases} D_{max} & ; \beta > 0.4, \\ \tilde{\tau} & ; \beta \le 0.4 \text{ AND } \tilde{\beta} \ge 0.3, \\ 0 & ; \text{otherwise} \end{cases}$$

(4.6.3-13)

4.6.3.6 Smoothed Delay Estimate and LTP Gain for SO 68, SO 70, and SO 73

The smoothed delay and long-term prediction gain estimates are updated as follows:

$$\tilde{\beta} = \begin{cases} \beta & ; \beta > 0.4, \\ 0.75\tilde{\beta} & ; \text{otherwise,} \end{cases}$$

(4.6.3-14)

then:

$$\tilde{\tau} = \begin{cases} dd_{max}; & \beta > 0.4 \\ \tilde{\tau}; & \beta \le 0.4 \text{ AND } \tilde{\beta} \ge 0.3 \\ 0; & \text{otherwise} \end{cases}$$

(4.6.3-15)

4.6.3.7 Composite Delay and Gain Calculations for SO 3

The delay estimates and long-term prediction gains for the two estimation windows are combined as follows:

Let (D_0, β_0) be the optimal delay and gain found for the window comprising the current frame, and let (D_1, β_1) be the optimal delay and gain found for the window comprising the second half of the current frame and the look-ahead. Then:

if ($\beta_0 > \beta_1 + 0.4$) {

    if ($|D_0 - D_1| > 15$) {

        $\tau = D_0$

        $\beta = \beta_0$

    }

    else {

        $\tau = (D_0 + D_1) / 2$

        $\beta = (\beta_0 + \beta_1) / 2$

    }

}

else {

    $\tau = D_1$

    $\beta = \beta_1$

1   }

4.6.3.8 Composite Delay and Gain Calculations for SO 68, SO 70, and SO 73

The delay estimates and long-term prediction gains for the two estimation windows are combined as follows:

Let (D_0, β_0) be the optimal delay and gain found for the window comprising the current frame, and let (D_1, β_1) be the optimal delay and gain found for the window comprising the second half of the current frame and the look-ahead. Then:

```
            if ( β 0 > β1 + 0.4 ) {
                if ( │D0 – D1│ > 15 ) {
                    τ = D0
                    β = β0
                }
                else {
                    τ = (D0 + D1) / 2
                    β = (β0 + β1) / 2
                }
            }
            else if ( β 0 > β1 + 0.05 AND │D0 – last_delay<7│  ) {
                if ( │D0 – D1│ > 15 ) {
                    τ = D0
                    β = β0
                }
                else {
```

$$\tau = \arg \max_{d}\left(\sum_{k=40}^{160-d} \varepsilon(k)\varepsilon(k+d) \right);$$

$$\left(\frac{Do+D1}{2} -1 \right) \le d \le \left(\frac{Do+D1}{2} +1 \right)$$

```
                    β = (β0 + β1) / 2
                }
            }
            else {
                τ = D1
                β = β1
            }
```

4.7 Determining the Data Rate for SO 3 and Voice Activity Detection for SO 68, SO 70, and SO 73

For SO 3, the rate determination algorithm (RDA) is used to select one of three encoding rates: Rate 1, Rate 1/2, and Rate 1/8. Active speech is encoded at Rate 1 or Rate 1/2, and background noise is encoded at Rate 1/8.

For SO 68, SO 70, and SO 73, the RDA that is described here is essentially a Voice Activity Detector that first separates active speech from background noise. Mode-decision, and a final rate-determination described in Section 4.8, and Section 4.8.4 are used to determine the final coding rate, and type of speech coding.

The actual data rate used for encoding the input signal may be modified to be Rate 1/2 maximum for the purpose of inserting a signaling message (see 2.2.1). Refer to 1.1 for guidelines concerning potential variations of the RDA.

Inputs:

- The bandwidth expanded correlation coefficients, $\boldsymbol{R}_W$

- The long-term prediction gain, β

Outputs: The speech encoder rate for the current frame, $Rate(m)$

Initialization:

- The rate histories, $Rate(m\text{-}1)$ and $Rate(m\text{-}2)$, are set to Rate 1/8 frames.

- The band energy, $E^{sm}_{f(i)}(m)$, initialization is given in 4.7.2.1.

- The background noise estimate, $B_{f(i)}(m)$, initialization is given in 4.7.2.2.

- The signal energy estimate, $S_{f(i)}(m)$, initialization is given in 4.7.2.3.

Processing: The data rate shall be determined by the processing steps defined in 4.7.1 and 4.7.2.

4.7.1 Estimating the Data Rate Based on Current Signal Parameters

The encoding rate is determined by comparing the current frame energy in each of two frequency bands, $f(1)$ and $f(2)$, to background noise estimates in these respective bands. Thresholds above the background noise in each band are determined by an estimated signal-to-noise ratio in each band. These thresholds are set for Rate 1, Rate 1/2, and Rate 1/8 encoding. The highest rate calculated from the two frequency bands is then selected as the encoding rate for the current frame.

4.7.1.1 Computing Band Energy

The rate determination algorithm uses energy thresholds to determine the encoding rate for the current frame. The input speech is divided into two distinct bands: band $f(1)$ spans 0.3-2.0 kHz, band $f(2)$ spans 2.0-4.0 kHz.[†] The band energy for band $f(i)$, $BE_{f(i)}$, is calculated as

$$BE_{f(i)} = R_w(0)R_{f(i)}(0) + 2.0 \sum_{k=1}^{L_h-1} R_w(k)R_{f(i)}(k) \tag{4.7.1-1}$$

[†] Whenever a variable (or symbol or value) with a subscript $f(i)$ appears in any equation or pseudocode in 4.5, it refers to a variable (or symbol or value) associated with either band $f(1)$ or band $f(2)$.

where:

$$R_{f(i)}(k) = \sum_{n=0}^{L_h-1-k} h_i(n)h_i(n+k) \quad ; 0 \le k < L_h \tag{4.7.1-2}$$

and $h_i(n)$ is the impulse response of the band-pass filter i, where $i = 1, 2$. $R_w(k)$ is the autocorrelation sequence defined in Equation 4.4.1-7, and $L_h = 17$ is the length of the impulse response of the band-pass filters.

The band-pass filters used for both frequency bands are defined in Table 4.7.1.1-1.

Table 4.7.1.1-1 FIR Filter Coefficients Used for Band Energy Calculations

n	$h_1(n)$ (lower band)	n	$h_2(n)$ (upper band)
0	-5.557699E-02	0	-1.229538E-02
1	-7.216371E-02	1	4.376551E-02
2	-1.036934E-02	2	1.238467E-02
3	2.344730E-02	3	-6.243877E-02
4	-6.071820E-02	4	-1.244865E-02
5	-1.398958E-01	5	1.053678E-01
6	-1.225667E-02	6	1.248720E-02
7	2.799153E-01	7	-3.180645E-01
8	4.375000E-01	8	4.875000E-01
9	2.799153E-01	9	-3.180645E-01
10	-1.225667E-02	10	1.248720E-02
11	-1.398958E-01	11	1.053678E-01
12	-6.071820E-02	12	-1.244865E-02
13	2.344730E-02	13	-6.243877E-02
14	-1.036934E-02	14	1.238467E-02
15	-7.216371E-02	15	4.376551E-02
16	-5.557699E-02	16	-1.229538E-02

4.7.1.2 Calculating Rate Determination Thresholds

The rate determination thresholds for each frequency band $f(i)$ are a function of both the background noise estimate, $B_{f(i)}(m-1)$, and the signal energy estimate, $S_{f(i)}(m-1)$, of the previous or $(m-1)$th frame. Two thresholds for each band are computed as

$$T_1\left(B_{f(i)}(m-1), SNR_{f(i)}(m-1)\right) = k1\left(SNR_{f(i)}(m-1)\right)B_{f(i)}(m-1), \tag{4.7.1-3}$$

$$T_2\left(B_{f(i)}(m-1), SNR_{f(i)}(m-1)\right) = k2\left(SNR_{f(i)}(m-1)\right)B_{f(i)}(m-1), \tag{4.7.1-4}$$

where the integer $SNR_{f(i)}(m-1)$ is:

$$SNR_{f(i)}(m-1) = \begin{cases} 0 & ; QSNRU_{f(i)}(m-1) < 0, \\ QSNRU_{f(i)}(m-1) & ; 0 \le QSNRU_{f(i)}(m-1) \le 7 \\ 7 & ; QSNRU_{f(i)}(m-1) > 7 \end{cases} \quad (4.7.1\text{-}5)$$

and where:

$$QSNRU_{f(i)}(m-1) = round\left\{ \left(10\log_{10}\left(S_{f(i)}(m-1)/B_{f(i)}(m-1) \right) - 20 \right)/5 \right\}. \quad (4.7.1\text{-}6)$$

4.7.1.2.1 SO 3

The functions $k1(\bullet)$ and $k2(\bullet)$ are defined in Table 4.7.1.2.2-1, and $B_{f(i)}(m\text{-}1)$ and $S_{f(i)}(m\text{-}1)$ are defined in 4.7.2.2 and 4.7.2.3, respectively.

Table 4.7.1.2.1-1 Threshold Scale Factors as a Function of SNR

$SNR_{f(i)}(m\text{-}1)$	$k1(SNR_{f(i)}(m\text{-}1))$	$k2(SNR_{f(i)}(m\text{-}1))$
0	7.0	9.0
1	7.0	12.6
2	8.0	17.0
3	8.6	18.5
4	8.9	19.4
5	9.4	20.9
6	11.0	25.5
7	31.6	79.6

4.7.1.2.2 SO 68, SO 70, and SO 73

The functions $k1(\bullet)$ and $k2(\bullet)$ depend on the value of RATE_REDUC, and are defined in Table 4.7.1.2.2-1 to Table 4.7.1.2.2-3, and $B_{f(i)}(m\text{-}1)$ and $S_{f(i)}(m\text{-}1)$ are defined in Sections 4.7.2.2 and 4.7.2.3, respectively. In particular, Table 4.7.1.2.2-1 is for SO 68 RATE_REDUC values '000', '001', and '111', for SO 70 RATE-REDUC values '100' and '111', and for SO 73 RATE_REDUC values '001' and '111'; Table 4.7.1.2.2-2 is for SO 68 RATE_REDUC values '010' and '011', for SO 70 RATE-REDUC values '000', and for SO 73 RATE_REDUC values '000', '010', and '011'; and Table 4.7.1.2.2-3 is for SO 68 and SO 73 RATE_REDUC values '100' - '110'.

Table 4.7.1.2.2-1 Threshold Scale Factors as a Function of SNR

$SNR_{f(i)}(m\text{-}1)$	$k1(SNR_{f(i)}(m\text{-}1))$	$k2(SNR_{f(i)}(m\text{-}1))$
0	7.0	9.0
1	7.0	12.6
2	8.0	17.0
3	8.6	18.5
4	8.9	19.4
5	9.4	20.9
6	22.0	51.0
7	73.2	158.2

Table 4.7.1.2.2-2 Threshold scale Factors as a Function of SNR

$SNR_{f(i)}(m-1)$	$k1(SNR_{f(i)}(m-1))$	$k2(SNR_{f(i)}(m-1))$
0	7.0	9.0
1	7.0	12.6
2	8.0	17.0
3	8.6	18.5
4	8.9	19.4
5	11.3	25.1
6	33.0	76.5
7	126.4	318.4

Table 4.7.1.2.2-3 Threshold Scale Factors as a Function of SNR

$SNR_{f(i)}(m-1)$	$k1(SNR_{f(i)}(m-1))$	$k2(SNR_{f(i)}(m-1))$
0	7.0	9.0
1	8.7	15.8
2	9.0	18.5
3	10.0	20.0
4	10.6	22.8
5	13.8	31.6
6	50.2	158.5
7	200.0	631.0

The threshold scale factors are identical for the low- and high-frequency bands.

4.7.1.3 Comparing Thresholds

Band energy, $BE_{f(i)}$, is compared with two thresholds: $T_1(B_{f(i)}(m-1), SNR_{f(i)}(m-1))$ and $T_2(B_{f(i)}(m-1), SNR_{f(i)}(m-1))$. If $BE_{f(i)}$ is greater than both thresholds, Rate 1 is selected. If $BE_{f(i)}$ is greater than only one threshold, Rate 1/2 is selected. If $BE_{f(i)}$ is at or below both thresholds, Rate 1/8 is selected. This procedure is performed for both frequency bands and the higher of the two encoding rates selected from the individual bands is chosen as the preliminary data rate, *Rate*(*m*), of the current frame *m*.

4.7.1.4 Performing Hangover

If the preliminary data rate decision (from 4.7.1) transitions from at least two consecutive Rate 1 frames to a lower rate frame, then the next *M* frames are encoded as Rate 1 before allowing the encoding rate to drop to Rate 1/2 and finally to Rate 1/8. The number of hangover frames, *M*, is a function of the $SNR_{f(1)}(m-1)$ (the SNR in the lower frequency band) and is denoted as $Hangover(SNR_{f(1)}(m-1))$ in Table 4.7.1.4.1-1. $SNR_{f(1)}(m-1)$ is calculated as defined in Equation 4.5.1-5. The hangover algorithm is defined by the following pseudocode:

>   {

1   if (*Rate*(*m*) == *Rate* 1)

2   *count* = 0

3   if (*Rate*(*m*-1) == *Rate* 1 and *Rate*(*m*-2) == *Rate* 1 and *Rate*(*m*) != *Rate* 1) {

4   if (*count* == 0)

5   *M* = *Hangover*($SNR_{f(1)}$(*m*-1))

6   if (*count* < *M*) {

7   *Rate*(*m*) = *Rate* 1

8   *count* = *count* + 1

9   }

10   }

11   }

where *Rate*(*m*) is the rate of the current frame and *Rate*(*m*-1) and *Rate*(*m*-2) are the rates of the previous two frames, respectively. Also, the rates for the previous frames, as used here, are those generated by the RDA prior to override logic, and are, therefore, not subject to modification by external rate control as described in 4.7.1.5.

4.7.1.4.1 Hangover Table for SO 3

Table 4.7.1.4.1-1 Hangover Frames as a Function of SNR

$SNR_{f(1)}$(*m*-1)	Hangover ($SNR_{f(1)}$(*m*-1))
0	7
1	7
2	7
3	3
4	0
5	0
6	0
7	0

4.7.1.4.2 Hangover Table for SO 68, SO 70, and SO 73

For SO 68, SO 70, and SO 73, the hangover table is chosen to be one of Table 4.7.1.4.2-1 to Table 4.7.1.4.2-3 depending on the capacity operating point (viz, required aggressiveness of the VAD to meet the capacity operating point). In particular, Table 4.7.1.4.2-1 is for SO 68 RATE_REDUC values '000', '001', and '111', for SO 70 RATE-REDUC values '100' and '111', and for SO 73 RATE_REDUC values '001' and '111'; Table 4.7.1.4.2-2 is for SO 68 RATE_REDUC values '010' and '011', for SO 70 RATE-REDUC values '000', and for SO 73 RATE_REDUC values '000', '010', and '011'; and Table 4.7.1.4.2-3 is for SO 68 and SO 73 RATE_REDUC values '100' – '110'.

₁ **Table 4.7.1.4.2-1 Hangover Frames as a Function of SNR**

$SNR_{f(1)}(m\text{-}1)$	Hangover $SNR_{f(1)}(m\text{-}1)$
0	5
1	5
2	4
3	3
4	0
5	0
6	0
7	0

₂

₃ **Table 4.7.1.4.2-2 Hangover Frames as a Function of SNR**

$SNR_{f(1)}(m\text{-}1)$	Hangover $SNR_{f(1)}(m\text{-}1)$
0	4
1	4
2	2
3	2
4	0
5	0
6	0
7	0

₄

₅ **Table 4.7.1.4.2-3 Hangover Frames as a Function of SNR**

$SNR_{f(1)}(m\text{-}1)$	Hangover $SNR_{f(1)}(m\text{-}1)$
0	4
1	3
2	2
3	1
4	0
5	0
6	0
7	0

₆

₇ 4.7.1.5 Constraining Rate Selection

₈ The rate selected by the procedures described in 4.7.1.3 and 4.7.1.4 are used for the current frame except where it is
₉ modified by the following constraints.

₁₀ If the previous frame was selected as Rate 1 and the current frame is selected as Rate 1/8, then the encoding rate of the
₁₁ current frame should be modified to Rate 1/2. There are no other restrictions on encoding rate transitions.

If the speech codec has been commanded to generate a Rate 1/2 maximum packet and the rate determined is Rate 1, it generates a Rate 1/2 packet. If the speech codec has been requested to generate a Blank packet, it generates a packet based on the rate determined by the rate determination algorithm (RDA).

4.7.2 Updating RDA Parameters

After the RDA is complete, RDA parameters shall be updated as described in 4.7.2.1 through 4.7.2.3.

4.7.2.1 Updating the Smoothed Band Energy

The band energy, $BE_{f(i)}$, calculated in Equation 4.5.1-1 is smoothed and used to estimate both the background noise energy (see 4.7.2.2) and signal energy (see 4.7.2.3) in each band. The smoothed band energy, $E^{sm}_{f(i)}(m)$, is computed as:

$$E^{sm}_{f(i)}(m) = \alpha_E(m)E^{sm}_{f(i)}(m-1) + \left(1 - \alpha_E(m)\right)BE_{f(i)} \quad ; \ 1 \le i \le 2 \tag{4.7.2-1}$$

where m refers to the current frame, and

$$\alpha_E(m) = \begin{cases} 0 & ; m \le 1, \\ 0.6 & ; m > 1. \end{cases} \tag{4.7.2-2}$$

This allows the smoothed band energy to be initialized to the band energy of the first frame ($m = 1$).

4.7.2.2 Updating the Background Noise Estimate

An estimate of the background noise level, $B_{f(i)}(m)$, is computed for the current, or mth, frame using $B_{f(i)}(m-1)$, $E^{sm}_{f(i)}(m)$ (see 4.7.2.1) and $SNR_{f(i)}(m-1)$ (see 4.7.1.2). Pseudocode describing the background noise update for band $f(i)$ is given as

    {

        if ($\beta < 0.30$ for 8 or more consecutive frames)

            $B_{f(i)}(m)$= min{ $E^{sm}_{f(i)}(m)$, 80954304, max{ $1.03B_{f(i)}(m-1)$, $B_{f(i)}(m-1)+ 1$ } }

        else {

           if ($SNR_{f(i)}(m-1) > 3$)

                $B_{f(i)}(m)$= min{ $E^{sm}_{f(i)}(m)$, 80954304, max{ $1.00547B_{f(i)}(m-1)$, $B_{f(i)}(m-1)+ 1$ } }

           else

                $B_{f(i)}$ (m)= min{ Esmf(i)(m), 80954304, $B_{f(i)}$ (m-1) }

        }

        if ($B_{f(i)}$ (m) < lownoise(i))

           $B_{f(i)}$ (m) = lownoise(i)

    }

where β, $E^{sm}_{f(i)}(m)$, and $SNR_{f(i)}(m-1)$ are defined in 4.6.3 and Equations 4.5.2-1 and 4.5.1-5, respectively. *lownoise*(1) equals 160.0 and *lownoise*(2) equals 80.0.

At initialization, the background noise estimate for the first frame, $B_{f(i)}(0)$, is set to 80,954,304 for both frequency bands, and the consecutive frame counter for β is set to zero. Initialization also occurs if the audio input to the encoder is disabled and then enabled.[†]

4.7.2.3 Updating the Signal Energy Estimate

The signal energy, $S_{f(i)}(m)$, is computed as

$$\{$$

If ($\beta > 0.5$ for 5 or more consecutive frames)

$$S_{f(i)}(m) = \max\{ E^{sm}_{f(i)}(m), \min\{ 0.97 S_{f(i)}(m-1), S_{f(i)}(m-1) - 1 \} \}$$

else

$$S_{f(i)}(m) = \max\{ E^{sm}_{f(i)}(m), S_{f(i)}(m-1) \}$$

$$\}$$

where β and $E^{sm}_{f(i)}(m)$ are defined in Section 4.6.3 and Equation 4.5.2-1, respectively.

At initialization, the signal energy estimates for the first frame, $S_{f(1)}(0)$ and $S_{f(2)}(0)$, are set to 51,200,000 and 5,120,000, respectively, and the consecutive frame counter for β is set to zero. Initialization also occurs if the audio input to the encoder is disabled and then enabled.

4.8 Mode Decision and Rate Decision for SO 68, SO 70, and SO 73

While the RDA block mentioned above aims to classify the speech broadly into speech, and background noise, the Mode-Decision block further classifies the non-silence part of the input is further classified using a multi-level decision logic into transition, voiced, unvoiced and end-of-word, based on a variety of features and the desired <u>capacity</u> operating point.

4.8.1 Pitch-based NACF

An important feature that is used in the mode-decision is the pitch-based autocorrelation similar to the β that was computed in the Open-loop pitch section. The procedure for computing the pitch based NACF is shown in the flowcharts below.

Inputs:

- residual – formant residual of current frame and look-ahead frame computed with unquantized LPC

- D_0, D_1 – delay estimates for the first and second subframe of the current frame

Outputs:

- nacf[4] – NACF from 4-subframe NACF track searching, returned value

- nacf_at_pitch – Maximum NACF around pitch estimates for 3 subframes

---

[†] This prevents the silence before the audio is connected from being mistaken as unusually low background noise.

High-pass filter two frames' (current and lookahead) residual, 3dB cut-off frequency at about 100 HZ

↓

Low-pass filter the residual with smoothly rolling off 12th order FIR.

↓

Decimate the band-pass filtered residual by 2

↓

Compute pitch based NACF for the 2nd subframe of current frame and 1st subframe of the look-ahead frome using pitch estimate of the respective subframe. The search window is centered on one pitch epoch behind the subframe, the window size it the lesser of 32 and 0.4*pitch.

↓

Compute NACF for the 2nd subframe of the look-ahead frame with a full-range search.

↓

Compute NACF by searching 4 subframe full range NACF tracks based on the same bandpassed decimated residuals.

↓

Back up current frame decimated residuals.

1

2   **Figure 4.8.1-1 Pitch-Based NACF Computation**

3 ## 4.8.2 Pitch Based NACF Computation

4 The Open Loop Pitch Search provides the NACF, a key ingredient for mode.

5 ### 4.8.2.1 Decimation of Formant Residual (Current Frames and Look-Ahead Frames)

6 The formant residual of the current frame, and the look-ahead frame is first filtered through a 3rd order High-pass, with
7 3dB cut-off frequency at about 100 HZ.

8 It is then low-pass filtered with a FIR filter of length 13 and then decimated by a factor of 2. This decimated residual
9 signal is denoted by $r_d(n)$.

10 ### 4.8.2.2 NACF Computation for the Two Subframes of Current Frame

11 The normalized autocorrelation functions (NACFs) for two subframes of the current frame are computed as

$$nacf(k) = MAX\left(\frac{sign\left(\sum_{n=0}^{40-1}\left(r_d(40k+n)r_d(40k+n-lag(k)+i)\right)\right)\left(\sum_{n=0}^{40-1}\left(r_d(40k+n)r_d(40k+n-lag(k)+i)\right)\right)^2}{\left(\sum_{n=0}^{40-1}\left(r_d(40k+n)r_d(40k+n)\right)\right)\left(\sum_{n=0}^{40-1}\left(r_d(40k+n-lag(k)+i)r_d(40k+n-lag(k)+i)\right)\right)}\right)$$

(4.8.2-1)

for $k = 1,2$, and the maximization done over all integer i such that,

$$-\frac{1+MAX\left(6,MIN(0.2 \cdot pitchlag(k),16)\right)}{2} \le i \le \frac{1+MAX\left(6,MIN(0.2 \cdot pitchlag(k),16)\right)}{2}$$

4.8.2.3 NACF Computation for the Look-Ahead Frame

The normalized autocorrelation functions (NACFs) for the lookahead subframe is computed as

$$nacf(2) = MAX\left(\frac{sign\left(\sum_{n=0}^{80-1}\left(r_d(80+n)r_d(80+n-i)\right)\right)\left(\sum_{n=0}^{80-1}\left(r_d(80+n)r_d(80+n-i)\right)\right)^2}{\left(\sum_{n=0}^{80-1}\left(r_d(80+n)r_d(80+n)\right)\right)\left(\sum_{n=0}^{80-1}\left(r_d(80+n-i)r_d(80+n-i)\right)\right)}\right)$$

(4.8.2-2)

the maximization done over all integer i such that,

$$\frac{20}{2} \le i < \frac{120}{2}.$$

4.8.3 Speech Coding Mode Decision Algorithm

Definitions:

- Transient – voiced but less periodic speech, encoded with Full-Rate CELP

- Up-transient – first voiced frame in active speech, encoded the same way as transient

- Down-transient – low energy voiced speech typically at end of word, encoded with Half-Rate CELP

- Voiced – highly periodic voiced speech, mainly vowels. Will be further classified as Full-Rate or Quarter-Rate according to different patterns to meet ADR requirements

- Unvoiced – unvoiced speech, mainly consonants. Encoded with Quarter-Rate NELP

- Silence – inactive speech. Encoded with Eighth-Rate same as EVRC-A

Inputs:

- curr_va – current frame VAD, which comes from EVRC RDA

- prev_voiced – indicates the presence of stationary voiced speech in previous frame

- prev_mode – previous frame mode decision

1   ●   nacf_at_pitch – subframe NACFs based on pitch estimates

2   ●   curr_ns_snr – low and high band instantaneous SNRs from the noise suppressor

3   ●   curr_snr – low and high band instantaneous SNRs from the EVRC RDA (VAD)

4   ●   t_in – speech buffer, which has both the current frame and the look ahead frame speech

5   **Outputs:**

6   ●   Current frame mode decision

7 ## 4.8.3.1 Energy and Zero Crossing Rate

8 Energies in the 0 to 2 kHz band and 2 to 4 kHz band are computed as $EL = \sum_{n=0}^{159} s_L^2(n)$, and $EH = \sum_{n=0}^{159} s_H^2(n)$ respectively,

9 where $s_L(n)$, and $s_H(n)$ are the low-pass filtered (using a 12th order pole-zero LPF), and the high-pass filtered (using a

10 12th order pole-zero HPF) versions of the input speech respectively. The speech signal energy itself is $E = \sum_{n=0}^{159} s^2(n)$.

11 The zero crossing rate ZCR is computed as

12   If (s(n)s(n+1)<0) ZCR=ZCR+1, $0 \le n < 159$

13 Table 4.8.3.1-1 describes the features used in the mode decision.

14

15   **Table 4.8.3.1-1 Features Used in Mode Decision**

Feature name	Description
curr_ns_snr	This feature is extracted from the noise suppressor. It's the band SNRs derived from the FFT. Low band is from 0 to 2 kHZ, high band is from 2 to 4 kHZ.
curr_snr	This is the band SNRs derived from VAD, after noise suppression.
curr_snr_diff	Band SNR difference, defined as curr_snr[0]-curr_snr[1]
zcr	This is the zero crossing rate of the current frame, defined as the number of sign changes in speech.
E	Current frame energy
Enext	Look ahead frame energy
bER	Band energy ratio, defined as log2(EL/EH), where EL is the low band current frame energy from 0 to 2k, and EH is the high band current frame energy from 2k to 4k.
vEav	3-frame average voiced energy, updated only if current frame is voiced, or reset to 0.1 at unvoiced or inactive speech.
vEprev	Previous valid 3-frame average voiced energy
vER	Current energy to previous 3-frame average voiced energy ratio, defined as 10*log10(E/vEprev).
vER2	Defined as MIN(20,10*log10(E/vEav))
maxsfe_idx	Current frame is evenly divided into 10 subframes, r.m.s value is computed on each subframe. This is the index to the subframe that has the largest r.m.s value in the second half of the current frame.

16

17 The flow chart in Figure 4.8.3.1-1 illustrates the high level mode-decision scheme.

> Set SNR thresholds (VOICEDTH,LOWVOICEDTH, UNVOICEDTH) for clean or noisy condition based on curr_ns_snr[0], if curr_ns_snr[0] is greater than 25 dB, it's clean speech.

> Filter current frame speech, compute band energy and additional features such as zcr, bER, vER etc.

> Make mode decision in active speech

> Update voiced indicator, update vEav, vEprev if voiced. Backup some features.

Figure 4.8.3.1-1 High Level Diagram of Mode Decision

The various features are used in a multilevel decision logic to make the mode-decision. The following diagrams outline how the mode decision is made. Current mode decision is a transition between states based on the previous mode decision and features such as NACFs. The states are the different modes that correspond to different coding schemes.

nacf_at_pitch[2]>VOICEDTH – second subframe NACF of the current frame is high
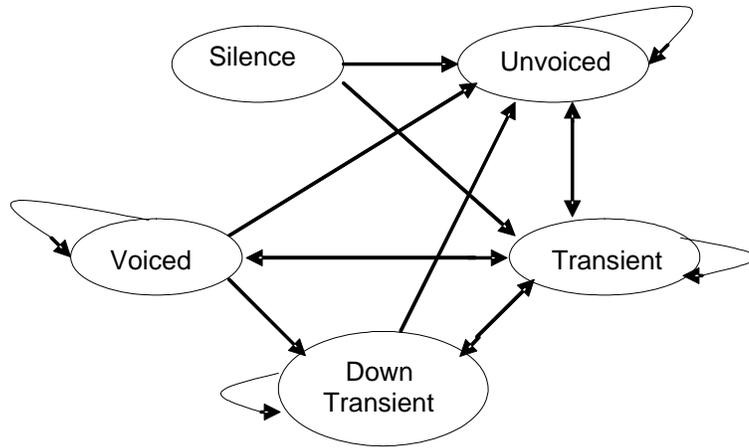
Figure 4.8.3.1-2 nacf_at_pitch[2]>VOICEDTH – Second Subframe NACF of the Current Frame Is High

The following pseudo-code shows the decision making process using the features mentioned above, and detailing the above state transition diagram.

```
if ( nacf_ap[2]>=VOICEDTH ) { // 2nd subframe NACF high
    switch (prev_mode) {

    case SILENCE:
```

```
1           m.speech_type=UP_TRANSIENT;
2           if (nacf_ap[3]<UNVOICEDTH && zcr>70 && bER<0)
3             m.speech_type=UNVOICED;
4           if (nacf_ap[3]<UNVOICEDTH && zcr>100 && vER<-20)
5             m.speech_type=UNVOICED;
6           if (vER<-25)
7             m.speech_type=UNVOICED;
8
9           break;
10
11      case UNVOICED:
12          m.speech_type=UP_TRANSIENT;
13          if (nacf_ap[3]<UNVOICEDTH && nacf_ap[4]<UNVOICEDTH && nacf<UNVOICEDTH
14    && zcr>70 && bER<0)
15              m.speech_type=UNVOICED;
16          if (nacf_ap[3]<UNVOICEDTH && zcr>100 && vER<-20 && E<Eprev)
17              m.speech_type=UNVOICED;
18          if (vER<-25)
19              m.speech_type=UNVOICED;
20
21          break;
22
23      case VOICED:
24          m.speech_type=VOICED;
25          /*
26          if ((nacf_ap[2]-nacf_ap[3]>0.3) && nacf_ap[3]<LOWVOICEDTH &&
27    E>=2*Eavg)
28              m.speech_type=TRANSIENT;
29           */
30          if ((nacf_ap[2]-nacf_ap[1]>0.3) && nacf_ap[1]<LOWVOICEDTH &&
31    E>0.5*Eprev)
32              m.speech_type=TRANSIENT;
33          if (vER<-18 && nacf_ap[3]<VOICEDTH)
34            m.speech_type=DOWN_TRANSIENT;
35          if (vER<-30 && E<Eprev)
36            m.speech_type=UNVOICED;
37          break;
38
39      case DOWN_TRANSIENT:
40          m.speech_type=DOWN_TRANSIENT;
41          //     if (E>0.2*vEav) m.speech_type=TRANSIENT;
42          if (vER<-30)
43            m.speech_type=UNVOICED;
44          if (E>Eprev) m.speech_type=TRANSIENT;
45          break;
```

```
1
2           default:  // previous is transient
3              m.speech_type=VOICED;
4              if ((nacf_ap[2]-nacf_ap[1]>0.45) || nacf_ap[1]<UNVOICEDTH )
5                m.speech_type=TRANSIENT;
6              if (nacf_ap[1]-nacf_ap[0]>0.3 && nacf_ap[1]<VOICEDTH && prev_mode
7        !=TRANSIENT)
8                m.speech_type=TRANSIENT;
9
10             if (((nacf_ap[1]-nacf_ap[0]>0.3)||(nacf_ap[2]-nacf_ap[1]>0.3)) &&
11       nacf_ap[1]<LOWVOICEDTH && nacf_ap[0]<UNVOICEDTH)
12               m.speech_type=TRANSIENT;
13
14             if (E<0.05*vEav && nacf_ap[3]<VOICEDTH)
15               m.speech_type=DOWN_TRANSIENT;
16             if (vER<-30 && E<Eprev)
17               m.speech_type=UNVOICED;
18             break;
19           }
20         }
21
```

22 nacf_at_pitch[2]<UNVOICEDTH – second subframe NACF of the current frame is low
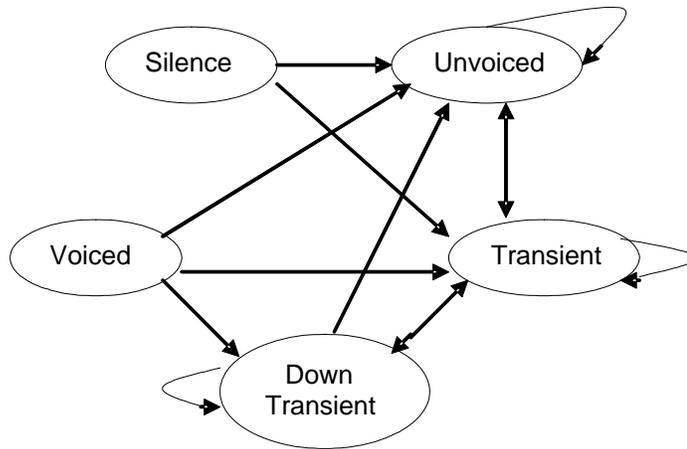
23 **Figure 4.8.3.1-3 nacf_at_pitch[2]<UNVOICEDTH – Second Subframe NACF of the Current Frame Is**
24 **Low**

25 The following pseudo-code shows the decision making process using the features mentioned above, and detailing the
26 above state transition diagram.

```
27
28             if ( nacf_ap[2]<=UNVOICEDTH ) {
29               switch (prev_mode) {
30
```

```
1            case SILENCE:
2                m.speech_type=UNVOICED;
3                if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
4        nacf_ap[4]>LOWVOICEDTH && zcr<110 && vER>-21 && (nacf_ap[3]>UNVOICEDTH ||
5        bER>-2))
6                    m.speech_type=UP_TRANSIENT;
7                if (zcr<60 && bER>6 && vER>-13 && curr_snr_diff>13.5463)
8                    m.speech_type=UP_TRANSIENT;
9                if ((nacf_ap[3]>LOWVOICEDTH || nacf_ap[4]>LOWVOICEDTH) && bER>3 &&
10       zcr<60 && vER>0)
11                   m.speech_type=UP_TRANSIENT;
12               if (nacf_ap[3]>LOWVOICEDTH && nacf_ap[4]>LOWVOICEDTH && bER>3 &&
13       zcr<40 && vER>-15)
14                   m.speech_type=UP_TRANSIENT;
15               break;
16
17           case UNVOICED:
18               m.speech_type=UNVOICED;
19               if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
20       nacf_ap[4]>LOWVOICEDTH && zcr<110 && vER>-21 && (nacf_ap[3]>UNVOICEDTH ||
21       bER>-2))
22                   m.speech_type=UP_TRANSIENT;
23               if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] && E>5*Eprev &&
24       bER>3 && vER>-16)
25                   m.speech_type=UP_TRANSIENT;
26               if (nacf_ap[4]>VOICEDTH && bER>3 && zcr<80 && E>2*Eprev && vER>-20)
27                   m.speech_type=UP_TRANSIENT;
28               if (nacf_ap[3]>VOICEDTH && bER>0 && zcr<80 && E>2*Eprev && vER>-16)
29                   m.speech_type=UP_TRANSIENT;
30               if ((nacf_ap[3]>LOWVOICEDTH || nacf_ap[4]>LOWVOICEDTH) && bER>0 &&
31       zcr<80 && maxsfe_idx==SFNUM && vER>-16)
32                   m.speech_type=UP_TRANSIENT;
33               if (nacf_ap[3]>UNVOICEDTH && nacf_ap[4]>UNVOICEDTH && zcr<80 &&
34       bER>4 && vER>-20 && curr_snr_diff>0 && curr_ns_snr[0]>=SNR_THLD)  // Clean
35       condition
36                   m.speech_type=UP_TRANSIENT;
37
38               if (nacf_ap[3]>LOWVOICEDTH && nacf_ap[4]>LOWVOICEDTH && zcr<80 &&
39       E>Eprev && vER>-20)
40                   m.speech_type=UP_TRANSIENT;
41               if (nacf_ap[3]>VOICEDTH && nacf_ap[4]>VOICEDTH && vER>-20)
42                   m.speech_type=UP_TRANSIENT;
43               if (bER>0 && zcr<80 && E>10*Eprev && vER>-10)
44                   m.speech_type=UP_TRANSIENT;
45
```

```
1            if (((bER>5 && vER>-10) || (bER>3 && vER>-6) || (bER>0 && vER>-3))
2    && zcr<80 && curr_snr_diff>0 && curr_ns_snr[0]>=SNR_THLD) // Clean condition
3              m.speech_type=UP_TRANSIENT;
4
5            if (bER>-3 && vER>-14 && zcr<80 && curr_ns_snr[0]<SNR_THLD) //Noisy
6    condition
7              m.speech_type=UP_TRANSIENT;
8            if (nacf>UNVOICEDTH && zcr<60 && bER>2 && vER>-5 &&
9    curr_ns_snr[0]<SNR_THLD) //Noisy condition
10              m.speech_type=UP_TRANSIENT;
11            if (bER>-5 && vER>-4 && nacf_ap[3]>UNVOICEDTH &&
12    nacf_ap[4]>nacf_ap[3] && zcr<100 && curr_ns_snr[0]<SNR_THLD) // Noisy
13    condition
14              m.speech_type=UP_TRANSIENT;
15
16          break;
17
18        case DOWN_TRANSIENT:  // following down_transient
19          m.speech_type=UNVOICED;
20          if (bER>0) {
21            if (vER>-20 && zcr<70)
22              m.speech_type=DOWN_TRANSIENT;
23            if ((nacf_ap[3]>LOWVOICEDTH || nacf_ap[4]>LOWVOICEDTH) &&
24    E>=2*Eprev && vER>-15)
25                m.speech_type=TRANSIENT;
26            if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
27    nacf_ap[4]>VOICEDTH && vER>-15 && E>Eprev)
28                m.speech_type=TRANSIENT;
29          }
30          else {
31            if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
32    nacf_ap[4]>VOICEDTH && vER>-15 && E>2*Eprev)
33                m.speech_type=TRANSIENT;
34          }
35          break;
36        case VOICED:
37        default:
38          if (bER>0) {
39            m.speech_type=TRANSIENT;
40            if (vER2<-15 && zcr<90 && E<Eprev && nacf_ap[3]<VOICEDTH)
41              m.speech_type=DOWN_TRANSIENT;
42            if (vER<-25 && E<Eprev)
43              m.speech_type=UNVOICED;
44          }
45          else {
```

```
1              m.speech_type=UNVOICED;
2              if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
3      nacf_ap[4]>UNVOICEDTH && zcr<100 && vER>-20 && curr_ns_snr[0]<SNR_THLD) //
4      Noisy condition
5                  m.speech_type=TRANSIENT;
6              if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
7      nacf_ap[4]>LOWVOICEDTH && zcr<100 && vER>-20 && curr_ns_snr[0]>=SNR_THLD) //
8      Clean condition
9                  m.speech_type=TRANSIENT;
10
11              if (((bER>-1 && vER>-12 && zcr<80) || (bER>-3 && vER>-10 &&
12      zcr<90) || (bER>-7 && vER>-3 && zcr<105)) && curr_ns_snr[0]<SNR_THLD) // Noisy
13      condition
14                  m.speech_type=TRANSIENT;
15              if (bER>-3 && vER>-10 && zcr<90 && curr_ns_snr[0]>=SNR_THLD) //
16      Clean condition
17                  m.speech_type=TRANSIENT;
18            }
19          break;
20
21        }
22      }
```

23 nacf_at_pitch[2]>=UNVOICEDTH && nacf_at_pitch[2]<=VOICEDTH – major gray area
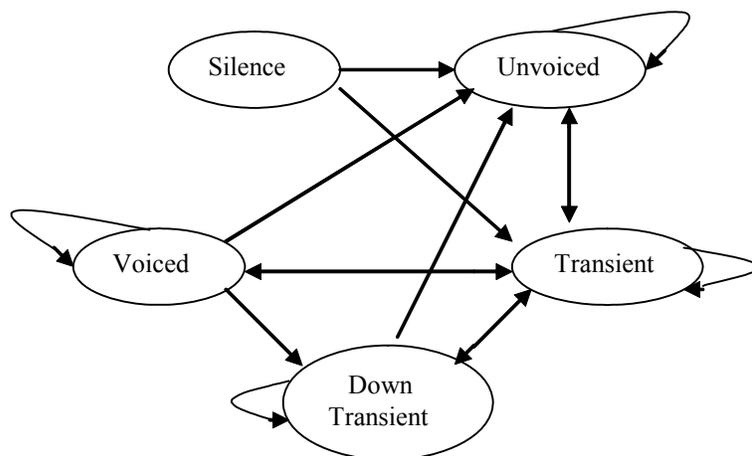
24

25   **Figure 4.8.3.1-4 nacf_at_pitch[2]>=UNVOICEDTH && nacf_at_pitch[2]<=VOICEDTH**

26

27 The following pseudo-code shows the decision making process using the features mentioned above, and detailing the
28 above state transition diagram.

29

```
30              switch (prev_mode) {
```

31

```
1          case SILENCE:
2              m.speech_type=UNVOICED;
3              if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
4          nacf_ap[4]>LOWVOICEDTH && zcr<110 && vER>-22)
5                  m.speech_type=UP_TRANSIENT;
6
7              if (nacf_ap[2]>LOWVOICEDTH && curr_snr_diff>0 && E>2*Eprev && zcr<70
8          && vER>-20)
9                  m.speech_type=UP_TRANSIENT;
10             if (nacf_ap[2]>LOWVOICEDTH && nacf_ap[3]>LOWVOICEDTH && zcr<70 &&
11         vER>-20)
12                 m.speech_type=UP_TRANSIENT;
13             if ((nacf_ap[4]>VOICEDTH || nacf_ap[3]>VOICEDTH) &&
14         nacf_ap[3]>UNVOICEDTH && zcr<70 && vER>-20)
15                 m.speech_type=UP_TRANSIENT;
16             if ((nacf_ap[3]-nacf_ap[2]>0.3) && E>2*Eprev && zcr<70 && bER>0 &&
17         vER>-20)
18                 m.speech_type=UP_TRANSIENT;
19             if (nacf_ap[4]>VOICEDTH && (nacf_ap[2]>LOWVOICEDTH ||
20         nacf_ap[3]>LOWVOICEDTH) && E>2*Eprev && curr_snr_diff>0 && vER>-20)
21                 m.speech_type=UP_TRANSIENT;
22             if (nacf_ap[3]>LOWVOICEDTH && nacf_ap[4]>LOWVOICEDTH && bER>3 &&
23         zcr<40 && vER>-15)
24                 m.speech_type=UP_TRANSIENT;
25             break;
26
27         case UNVOICED:
28             m.speech_type=UNVOICED;
29             if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
30         nacf_ap[4]>LOWVOICEDTH && zcr<110 && vER>-22)
31                 m.speech_type=UP_TRANSIENT;
32             if ((nacf_ap[4]>LOWVOICEDTH || nacf_ap[3]>LOWVOICEDTH) && zcr<100 &&
33         E>Eprev && bER>0 && vER>-20)
34                 m.speech_type=UP_TRANSIENT;
35             if (nacf_ap[4]>VOICEDTH && nacf>UNVOICEDTH && zcr<100 && E>Eprev &&
36         bER>1 && vER>-16)
37                 m.speech_type=UP_TRANSIENT;
38             if (zcr<60 && bER>3 && E>5*Eprev && curr_snr_diff>9.0309 && vER>-20)
39                 m.speech_type=UP_TRANSIENT;
40             if ((nacf_ap[3]>UNVOICEDTH || nacf_ap[4]>UNVOICEDTH) && bER>1 &&
41         vER>-16 && zcr<80 && curr_snr_diff>-6.0206)
42                 m.speech_type=UP_TRANSIENT;
43
44             if (nacf_ap[2]>LOWVOICEDTH && curr_snr_diff>0 && E>2*Eprev && zcr<70
45         && vER>-20)
46                 m.speech_type=UP_TRANSIENT;
```

```
            if (nacf_ap[2]>LOWVOICEDTH && nacf_ap[3]>LOWVOICEDTH && zcr<100 &&
vER>-20)
                m.speech_type=UP_TRANSIENT;
            if ((nacf_ap[4]>VOICEDTH || nacf_ap[3]>VOICEDTH) &&
nacf_ap[3]>UNVOICEDTH && zcr<80 && vER>-20)
                m.speech_type=UP_TRANSIENT;
            if ((nacf_ap[3]>LOWVOICEDTH || nacf_ap[4]>LOWVOICEDTH) && bER>0 &&
zcr<100 && maxsfe_idx==SFNUM && vER>-20)
                m.speech_type=UP_TRANSIENT;
            if (nacf_ap[3]>UNVOICEDTH && nacf_ap[4]>UNVOICEDTH && zcr<100 &&
vER>-3 && bER>-5 && curr_ns_snr[0]<SNR_THLD)  // Noisy condition
                m.speech_type=UP_TRANSIENT;

            if (bER>4 && zcr<100 && maxsfe_idx==SFNUM && vER>-20)
                m.speech_type=UP_TRANSIENT;

            if ((nacf_ap[3]-nacf_ap[2]>0.3) && E>2*Eprev && zcr<70 && bER>0 &&
vER>-20)
                m.speech_type=UP_TRANSIENT;
            if (nacf_ap[4]>VOICEDTH && (nacf_ap[2]>LOWVOICEDTH ||
nacf_ap[3]>LOWVOICEDTH) && E>2*Eprev && curr_snr_diff>0 && vER>-20)
                m.speech_type=UP_TRANSIENT;

            if (((bER>5 && vER>-10) || (bER>3 && vER>-6) || (bER>0 && vER>-3))
&& zcr<80 && curr_ns_snr[0]>=SNR_THLD ) // Clean condition
                m.speech_type=UP_TRANSIENT;

            if (bER>-3 && vER>-14 && zcr<80 && curr_ns_snr[0]<SNR_THLD) // Noisy
condition
                m.speech_type=UP_TRANSIENT;
          break;
        case DOWN_TRANSIENT:
          m.speech_type=UNVOICED;
          if (bER>0) {
            if (vER>-20 && zcr<70)
              m.speech_type=DOWN_TRANSIENT;
            if ((nacf_ap[3]>LOWVOICEDTH || nacf_ap[4]>LOWVOICEDTH) &&
E>=2*Eprev && vER>-15)
                m.speech_type=TRANSIENT;
            if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
nacf_ap[4]>VOICEDTH && vER>-15)
                m.speech_type=TRANSIENT;

          }
          else {
```

```
            if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] &&
nacf_ap[4]>VOICEDTH && vER>-15 && E>2*Eprev)
                m.speech_type=TRANSIENT;
            }
          break;
        case VOICED:
        default:
          if (nacf_ap[2]>=LOWVOICEDTH) {
            m.speech_type=VOICED;
            if (nacf_ap[1]<LOWVOICEDTH || (nacf_ap[1]<VOICEDTH && nacf_ap[1]-
nacf_ap[0]>0.3) || bER<0)
                m.speech_type=TRANSIENT;
            if (vER<-15 && nacf_ap[3]<LOWVOICEDTH && E<Eprev)
              m.speech_type=DOWN_TRANSIENT;
            if (bER<0 && vER<-20 && Enext<E && nacf_ap[3]<UNVOICEDTH &&
nacf_ap[4]<UNVOICEDTH)
                m.speech_type=UNVOICED;
            if (bER<-30 && E<Eprev)
              m.speech_type=UNVOICED;

            if ((nacf_ap[2]-nacf_ap[3]>0.3) && nacf_ap[3]<0.3 &&
nacf_ap[4]<0.25)
                m.speech_type=TRANSIENT;
          }
          else {
            if (bER>0) {
              m.speech_type=TRANSIENT;
              if (vER2<-15 && zcr<90 && E<Eprev && nacf_ap[3]<VOICEDTH)
                m.speech_type=DOWN_TRANSIENT;
              if (vER<-25 && E<Eprev)
                m.speech_type=UNVOICED;
            }
            else {
              m.speech_type=UNVOICED;
              if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] && zcr<100 &&
vER>-20 && curr_ns_snr[0]>=SNR_THLD) // Clean condition
                  m.speech_type=TRANSIENT;
              if (nacf_ap[3]>nacf_ap[2] && nacf_ap[4]>nacf_ap[3] && zcr<110 &&
vER>-15 && curr_ns_snr[0]<SNR_THLD) // Noisy condition
                  m.speech_type=TRANSIENT;
              if (bER>-1 && vER>-12 && zcr<80)
                m.speech_type=TRANSIENT;
              if (bER>-3 && vER>-10 && nacf>UNVOICEDTH && zcr<90)
                  m.speech_type=TRANSIENT;
```

```
1                           if (bER>-7 && vER>-3 && zcr<105 && curr_ns_snr[0]<SNR_THLD) //
2       Noisy condition
3                               m.speech_type=TRANSIENT;
4                   }
5
6               }
7
8               break;
9             }
10          }
11
```

4.8.4 Rate-Decision for SO 68, SO 70, and SO 73

Depending on Input Capacity Operating Point, the mode-decision is first mapped to the different modes of the encoder at different rates as following according to the general rules shown in the following table.

Table 4.8.4-1 Mapping of Speech Mode Decision to Encoding Rate and Mode

Open loop mode decision	Mode and rate used to encode the frame in SO 68, SO 70, and SO 73 narrowband mode	Mode and rate used to encode the frame in SO 70 and SO 73 wideband mode
Transient	Full-Rate CELP	Full-Rate CELP
Down Transient	Half-Rate CELP or Full-Rate CELP	Full-Rate CELP
Uptransient	Full-Rate CELP	Full-Rate CELP
Voiced	FPPP, or QPPP, or FCELP processing Depending on the Capacity Operating Point, and a pre-determined pattern of interleaving QPPP with either FCELP or FPPP. A closed loop decision is later used in QPPP processing to achieve the input capacity operating point of EVRC-B.	Full-Rate CELP.
Unvoiced	Quarter-Rate NELP processing	Half-Rate NELP processing
Silence	Eighth-Rate processing	Eighth-Rate processing

4.8.4.1 Constraining Rate Selection

If the speech codec has been commanded to generate a Rate 1/2 maximum packet and the rate determined is Rate 1, it generates a Rate 1/2 CELP packet. If the speech codec has been requested to generate a Blank packet, it generates a packet based on the rate determined by the rate selection algorithm.

4.9 Quantization of LSP Parameters for SO 3

Inputs: The inputs to the LSP quantizer are:

- The unquantized LSPs for the current frame, $\Omega(m)$

- The rate of the current frame, $Rate(m)$

Outputs: The outputs from the LSP quantizer are:

1 • The quantized LSPs for the current frame, $\Omega_q(m)$

2 **Processing:** The quantizer takes the form of a weighted split vector LSP quantizer. The splits for the quantizer for a

3 given *Rate* are given in parameter Table 4.9-1. All of the sub-matrices are coded using the number of bits found in this

4 table. The codebooks $q_{rate}(k)$, that are used to quantize each set of LSPs can be found in Table 8-1 through Table 8-9.

5 Each LSP set has one codebook with dimensions $n_{sub}(k)$ by $n_{size}(k)$. $n_{sub}(k)$ is the number of LSP parameters in the set

6 and $n_{size}(k)$ is the codebook size.

7

8 **Table 4.9-1 LSP Parameter Splits**

Codebook number k	LSP parameters in set $\Omega(m)$	LSP split starting parameter index $B(k)$	Number of LSP parameters $n_{sub}(k)$	Code book size $n_{size}(k)$	Number of bits allocated
Rate 1					
1	$\Omega(m, i); i \in (1, 2)$	1	2	64	6
2	$\Omega(m, i); i \in (3, 4)$	3	2	64	6
3	$\Omega(m, i); i \in (5, 6, 7)$	5	3	512	9
4	$\Omega(m, i); i \in (8, 9, 10)$	8	3	128	7
Rate 1/2					
1	$\Omega(m, i); i \in (1, 2, 3)$	1	3	128	7
2	$\Omega(m, i); i \in (4, 5, 6)$	4	3	128	7
3	$\Omega(m, i); i \in (7, ..., 10)$	7	4	256	8
Rate 1/8					
1	$\Omega(m, i); i \in (1, ..., 5)$	1	5	16	4
2	$\Omega(m, i); i \in (6, ..., 10)$	6	5	16	4

9

10 The number of codebooks, k_{num}, for each rate is shown in Table 4.9-2.

11 **Table 4.9-2 Number of LSP Codebooks**

Rate	Number of codebooks k_{num}
1	4
1/2	3
1/8	2

12

4.9.1 Computation of Weights

14 The weights, $w(i)$, for the LSP frequencies are defined as follows:

$$\omega(i) = \begin{cases} \dfrac{50}{2\pi}+1 & ; \Delta_\Omega(m,i) = 0, \\ \dfrac{0.5}{2\pi\Delta_\Omega(m,i)}+1 & ; \text{otherwise}, \end{cases} \quad 1 \le i \le 10,$$

(4.9.1-1)

where:

$$\Delta_\Omega(m,i) = \begin{cases} \Omega(m,i+1) - \Omega(m,i) & ; i=1, \\ \min\big((\Omega(m,i) - \Omega(m,i-1)),(\Omega(m,i+1) - \Omega(m,i))\big) & ; 2 \le i \le 9, \\ \Omega(m,i) - \Omega(m,i-1) & ; i=10. \end{cases}$$

(4.9.1-2)

4.9.2 Error Matrix Computation

Compute the error matrix, $e(k)$, for each set of LSP frequencies k:

$$e(k,j) = \sum_{i=1}^{nsub(k)} \omega\big(B(k)+i-1\big)\big(\Omega\big(m,B(k)+i-1\big) - q_{rate}(k,i,j)\big)^2 \quad ;1 \le j \le n_{size}(k), \ 1 \le k \le k_{num},$$

(4.9.2-1)

where i is the sub-index of each LSP parameter set, and j is the index of the codebook entries. All the other parameters are defined in Table 4.9-1 and Table 4.9-2.

4.9.3 Adjustment of Quantization Error

The error associated between codebooks is adjusted to prevent excessively small (or negative) gaps between LSPs at codebook seams. If the difference between a particular $q(k, 1, j)$ and $\Omega_q(m, B(k)\text{-}1)$ is less than δ, $e(k, j)$ is assigned a value that will make it invalid for the search. The error matrix is adjusted by:

$$e(k,j) = \begin{cases} MAXFLOAT & ;\big(q_{rate}(k,l,j) - \Omega_q(m,B(k)-1)\big) \le \delta, \\ e(k,j) & ;\text{otherwise}, \end{cases} \quad 1 \le j \le n_{size}(k), \ 1 \le k \le k_{num}$$

(4.9.3-1)

where $\delta = 0.05 / 2\pi$.

4.9.4 Quantization Search

Each error matrix, $e(k,i)$, is searched for the minimum error, $e_{min}(k)$, as defined by:

$$e_{min}(k) = \min\{e(k,j)MAXFLOAT\} \quad ;1 \le j \le n_{size}(k), \ 1 \le k \le k_{num}.$$

(4.9.4-1)

The codebook index to be transmitted, *LSPIDX(k)*, is defined as the index, j, where $e_{min}(k)$ was found above for all k.

4.9.5 Generation of Quantized LSP Parameters

The codebook indices, *LSPIDX(k)*, are used to generate a quantized set of LSP parameters:

$$\Omega_q(m,B(k)+i-1) = q_{rate}(k,i,LSPIDX(k)) \quad ;1 \le i \le n_{sub}(k), \ 1 \le k \le k_{num}.$$

(4.9.5-1)

4.10 Quantization of LSP Parameters for SO 68, SO 70, and SO 73

Inputs: The inputs to the LSP quantizer are:

1. ● The unquantized LSPs for the current frame, $\Omega(m)$ (also referred to in this section as l)

2. ● The rate of the current frame, *Rate*(*m*)

3. **Outputs:** The outputs from the LSP quantizer are:

4. ● The quantized LSPs for the current frame, $\Omega_q(m)$

4.10.1 Full-Rate, Half-Rate, and Quarter-Rate Unvoiced: Delta LSP VQ

For Full-Rate, Half-Rate, and Quarter-Rate unvoiced frames, a delta split LSPVQ schemed is used. The weights used in the error calculation in the LSPVQ are computed as:

$$w(i) = \begin{cases} \dfrac{0.5}{2\pi\sqrt{(l(i)-l(i-1))(l(i+1)-l(i))}}; & 1 \le i \le 8 \\[2mm] \dfrac{0.5}{2\pi\sqrt{l(i)(l(i+1)-l(i))}}; & i = 0 \\[2mm] \dfrac{0.5}{2\pi\sqrt{(l(i)-l(i-1))(0.5)}}; & i = 9 \end{cases} \tag{4.10.1-1}$$

The delta split LSPVQ is in general represented by the following equation

$$k^* = \arg\max_k \sum_{i=i_l}^{i_u} \left[\left(\frac{l(i)-ll}{lu-ll} - cb[k][i-i_l] \right)^2 w(i) \right]; \qquad 0 \le k < CBSIZE \tag{4.10.1-2}$$

with the following conditions

$$cb[k][0] = max\left(cb[k][0], \frac{0.05}{2\pi(lu-ll)} \right) \tag{4.10.1-3}$$

$$cb[k][i_u - i_l - 1] = min\left(cb[k][i_u - i_l - 1], 1 - \frac{0.05}{2\pi(lu-ll)} \right) \tag{4.10.1-4}$$

where i_u and i_l are the upper and lower indices of the input LSP vector corresponding to a segment, lu and ll are the lower and upper limit of the segment delta vector quantization, cb[x][y] is the code-vector corresponding to the xth entry in the code-book, y is index of the element in the current split, i.e., $il \le y \le lu$, and CBSIZE is the size of the codebook for the current split.

The dequantized LSP vector corresponding to the chosen index, k^*, for the split segment is given by

$$\hat{l}(i) = ll + (lu-ll)cb[k^*][i-i_l]; \qquad i_l \le i \le i_u \tag{4.10.1-5}$$

with the following two conditioning on the end points of the split segment,

$$\hat{l}(i_l) = ll + \frac{0.05}{2\pi}; \qquad\qquad if\ cb[k^*][0] < \frac{0.05}{2\pi(lu-ll)} \tag{4.10.1-6}$$

$$\hat{l}(i_u) = lu - \frac{0.05}{2\pi}; \qquad\qquad if\ cb[k^*][i_u - i_l - 1] > 1 - \frac{0.05}{2\pi(lu-ll)} \tag{4.10.1-7}$$

4.10.2 Full-Rate LSP and Unvoiced Half-Rate LSP VQ for SO 70 and SO 73 WB Mode

Number of bits = 28.

The following table shows the parameters of the LSP VQ for Full-Rate and SO 70 and SO 73 WB unvoiced Half-Rate frames.

Table 4.10.2-1 LSPVQ Parameters for Full- Rate and SO 70 and SO 73 WB Unvoiced Half-Rate

Segment	i_l	i_u	ll	lu	# of elements in split	Elements	# of bits for the split	# of entries in quantization table
0	0	1	0	0.5	2	$l(0)$, $l(1)$	6	64
1	2	3	$\hat{l}(1)$	$\hat{l}(4)$	2	$l(2)$, $l(3)$	6	64
2	4	6	0	0.5	3	$l(4)$, $l(5)$, $l(6)$	9	512
3	7	9	$\hat{l}(6)$	0.5	3	$l(7)$, $l(8)$, $l(9)$	7	128

Segments 0 and 2 are quantized absolutely using $ll = 0$, and $lu = 0.5$ first.

The quantization of Segments 1 and 3 is performed using a multiple path approach where 4 best candidates (say S0_C0, S0_C1, S0_C2, S0_C3 in decreasing order of optimality) are independently picked for Segment 0, and 6 best candidates (say S2_C0, S2_C1, S2_C2, S2_C3, S2_C4, S2_C5) are independently picked for Segment 2. The quantization of the split Segments 1 and 3 depend on the candidates picked for Segments 0 and 2. The search for the best candidates of Segments 1 and 3 are performed for the pairs of candidates of Segments 0 and 1 as shown in Table 4.10.2-2.

Table 4.10.2-2 Multiple Candidate Search in Full-Rate and SO 70 and SO 73 WB Unvoiced Half-Rate LSPVQ

Segment 0 candidates in decreasing order of optimality	Number of top candidates of Segment 2 used in search	Segment 2 candidates to be used in pair with Segment 0 candidate
S0_C0	6	S2_C0, S2_C1, S2_C2, S2_C3, S2_C4, S2_C5
S0_C1	5	S2_C0, S2_C1, S2_C2, S2_C3, S2_C4
S0_C2	4	S2_C0, S2_C1, S2_C2, S2_C3
S0_C3	3	S2_C0, S2_C1, S2_C2

The basic idea is that the best candidate in Segment 0 are paired with all of the candidates of Segment 2, the next best candidate of Segment 0 are paired with one less number of candidates of Segment 2 and so on.

For each pair of candidates of Segment 0, and Segment 2, the best candidate for Segments 1, and 3 are chosen according the above equation above.

The final set of candidates (and their codebook indices) for all these segments is chosen as the one that minimizes the following joint (10 dimensional going across segment split boundaries) error between the unquantized LSP and the quantized LSP (obtained from independently selecting the best quantized vector for the split segments).

$$e = \sum_{i=0}^{9} w(i)\big(l(i) - \hat{l}(i)\big)^2 \qquad\qquad (4.10.2-1)$$

4.10.3 Half-Rate LSPVQ for SO 68, SO 70, and SO 73 NB Mode

Number of bits = 22.

The following table shows the parameters of the LSP VQ for Half-Rate frames.

Table 4.10.3-1 LSPVQ Parameters for Half-Rate

Segment	i_l	i_u	ll	lu	# of elements in split	Elements	# of bits for the split	# of entries in quantization table
0	0	2	0	0.5	3	$l(0)$, $l(1)$, $l(2)$	7	128
1	3	5	$\hat{l}(2)$	$\hat{l}(6)$	3	$l(3)$, $l(4)$, $l(5)$	7	128
2	6	9	0	0.5	4	$l(6)$, $l(7)$, $l(8)$, $l(9)$	8	256

Segments 0 and 2 are quantized absolutely using $ll = 0$, and $lu = 0.5$ first.

The quantization of Segment 1 is performed using a multiple path approach where 4 best candidates (say S0_C0, S0_C1, S0_C2, S0_C3 in decreasing order of optimality) are independently picked for Segment 0, and 6 best candidates (say S2_C0, S2_C1, S2_C2, S2_C3, S2_C4, S2_C5) are independently picked for Segment 2. The quantization of the split Segment 1 depends on the candidates picked for Segments 0 and 2. The search for the best candidates of Segment 1 is performed for the pairs of candidates of Segments 0 and 1 as shown below in the table.

Table 4.10.3-2 Multiple Candidate Search in Half-Rate LSPVQ

Segment 0 candidates in Decreasing order of optimality	Number of top candidates of Segment 2 used in search	Segment 2 candidates to be used in pair with Segment 0 candidate
S0_C0	6	S2_C0, S2_C1, S2_C2, S2_C3, S2_C4, S2_C5
S0_C1	5	S2_C0, S2_C1, S2_C2, S2_C3, S2_C4
S0_C2	4	S2_C0, S2_C1, S2_C2, S2_C3
S0_C3	3	S2_C0, S2_C1, S2_C2

The basic idea is that the best candidate in Segment 0 are paired with all of the candidates of Segment 2, the next best candidate of Segment 0 are paired with one less number of candidates of Segment 2 and so on.

For each pair of candidates of Segment 0, and Segment 2, the best candidate for Segments 1 is chosen according the above equation.

The final set of candidates (and their codebook indices) for all these segments is chosen as the one that minimizes the following joint (10 dimensional going across segment split boundaries) error between the unquantized LSP and the quantized LSP (obtained from independently selecting the best quantized vector for the split segments).

$$e = \sum_{i=0}^{9} w(i)\left(l(i) - \hat{l}(i)\right)^2 \tag{4.10.3-1}$$

4.10.4 Quarter-Rate Unvoiced LSPVQ for SO 68 and SO 73 NB Mode

Number of bits = 16.

The following table shows the parameters of the LSP VQ for Quarter-Rate unvoiced frames.

Table 4.10.4-1 LSPVQ Parameters for Quarter-Rate Unvoiced

Segment	i_l	i_u	ll	lu	# of elements in split	Elements	# of bits for the split	# of entries in quantization table
0	0	4	0	$\hat{l}(5)$	5	$l(0)$, $l(1)$, $l(2)$, $l(3)$, $l(4)$	8	256
1	5	9	0	0.5	5	$l(5)$, $l(6)$, $l(7)$, $l(8)$, $l(9)$	8	256

Segment 1 is quantized absolutely using $ll = 0$, and $lu = 0.5$ first.

The quantization of Segment 0 is performed using a multiple path approach where 6 best candidates (say S1_C0, S1_C1, S1_C2, S1_C3, S1_C4, S1_C5) are independently picked for Segment 1. The quantization of the split Segment 0 depends on the candidates picked for Segment 1. The search for the best candidates of Segment 0 is performed for each of the 6 candidates of Segment 1 using the above equation.

The final set of candidates (and their codebook indices) for all these segments is chosen as the one that minimizes the following joint (10 dimensional going across segment split boundaries) error between the unquantized LSP and the quantized LSP (obtained from independently selecting the best quantized vector for the split segments).

$$e = \sum_{i=0}^{9} w(i)\left(l(i) - \hat{l}(i)\right)^2$$

(4.10.4-1)

4.10.5 Quarter-Rate Voiced LSPVQ for SO 68 and SO 73 NB Mode

A second order moving average (MA) predictor is used for quantizing the LSPs in Quarter-Rate voiced frame processing.

4.10.5.1 QPPP LSP Quantization Weight Computation

The weights described in Section 4.10.1 are used to compute the weighted square error distortion metrics needed to search the LSP VQ codebooks in the QPPP LSP VQ procedure described in this section.

The MA predictive quantizer ensures that the effect of frame errors discontinues after 2 frames. The quantized LSP vector at the mth frame is given as

$$\hat{\mathbf{L}}_{im} = 0.4\mathbf{C}_m + 0.4\mathbf{C}_{m-1} + 0.2\mathbf{C}_{m-2}$$

(4.10.5-1)

where $\mathbf{C}_m$ is the codebook vector for the mth frame, and $\hat{\mathbf{L}}_{im}$ is the vector representation of the quantized LSP vector denoted by $\hat{l}_i$. The target vector $\mathbf{U}_m$ to the quantizer is constructed using the input unquantized LSP vector $\mathbf{L}_{im}$, and the past codebook vectors as

$$\mathbf{U}_m = \frac{\mathbf{L}_{im} - 0.4\mathbf{C}_{m-1} - 0.2\mathbf{C}_{m-2}}{0.4}$$

(4.10.5-2)

A regular 16-bit split VQ structure (with weighted MSE as the selection criterion) is used to encode the first 6 elements of the target vector $\mathbf{U}_m$ with a 10-bit codebook and the last 4 elements using 6 bits. Let i_1^* and i_2^* be the chosen best indices out of the split VQ for the two splits. The quantizer output $\mathbf{C}_m$ is constructed as follows:

$$C_m(i) = \begin{cases} CB1[i_1^*][i]; & 0 \le i < 6 \\ CB2[i_2^*][i-6]; & 6 \le i < 10 \end{cases} \tag{4.10.5-3}$$

4.10.5.2 Constructing the MAVQ Memory During Frames That Are Not QPPP

During frames that are not QPPP, a MAVQ scheme is not used. Therefore, it is essential to construct the MAVQ memory elements, $\mathbf{C}_{m-1}$ and $\mathbf{C}_{m-2}$, as follows:

$$\mathbf{C}_{m-1} = \mathbf{C}_{m-1} = \hat{\mathbf{L}}_{im}$$

where the vector $\hat{\mathbf{L}}_{im}$ is the the quantized LSPs, $\hat{l}$.

4.10.6 Eighth-Rate LSPVQ for SO 68, SO 70, and SO 73 NB Mode

The Eighth-Rate LSPVQ for SO 68, SO 70, and SO 73 NB mode is the same as that for SO 3 and described in Section 4.9.

4.10.7 Eighth-Rate LSPVQ SO 70 and SO 73 WB Mode

Number of bits = 10.

The following table shows the parameters of the LSP VQ for Eighth-Rate SO 70 and SO 73 WB mode frames.

Table 4.10.7-1 LSPVQ Parameters for Eighth-Rate SO 70 and SO 73 WB Mode

Segment	i_l	i_u	ll	lu	# of elements in split	Elements	# of bits for the split	# of entries in quantization table
0	0	4	0	0.5	5	$l(0)$, $l(1)$, $l(2)$, $l(3)$, $l(4)$	5	32
1	5	9	$\hat{l}(4)$	0.5	5	$l(5)$, $l(6)$, $l(7)$, $l(8)$, $l(9)$	5	32

Segment 0 is quantized absolutely using $ll = 0$, and $lu = 0.5$ first.

The quantization of Segment 0 is performed using a multiple path approach where 6 best candidates (say S1_C0, S1_C1, S1_C2, S1_C3, S1_C4, S1_C5) are independently picked for Segment 0. The quantization of the split Segment 1 depends on the candidates picked for Segment 0. The search for the best candidates of Segment 1 is performed for each of the 6 candidates of Segment 0 using the above equation.

The final set of candidates (and their codebook indices) for all these segments is chosen as the one that minimizes the following joint (10 dimensional going across segment split boundaries) error between the unquantized LSP and the quantized LSP (obtained from independently selecting the best quantized vector for the split segments).

$$e = \sum_{i=0}^{9} w(i)\left(l(i) - \hat{l}(i)\right)^2 \tag{4.10.7-1}$$

4.11   Encoding at Rates 1/2 and 1 for SO 3, SO 68, SO 70, and SO 73

The algorithm for encoding at Rate 1 is similar to that used for encoding at Rate 1/2 except for the quantization tables used and the fact that certain quantities (e.g., the spectral transition indicator, *LPCFLAG*, and the delay difference, *DDELAY*) are transmitted only at Rate 1. Encoding is accomplished as follows:

Inputs: The inputs to encoding are:

- The quantized LSPs from the previous frame, $\Omega_q(m\text{-}1)$

- The unquantized LSPs from the current and previous frames, $\Omega(m)$ and $\Omega(m\text{-}1)$

- The short-term prediction residual, $\varepsilon(n)$

- The pitch delay estimate for the previous frame, $\tau(m\text{-}1)$

- The pitch delay estimate for the current frame, $\tau(m)$

- The long term prediction gain, β

- The rate of the current frame, *Rate*(*m*)

Outputs: The outputs of encoding are:

- The quantized LSPs for the current frame, $\Omega_q(m)$

- The LSP indices corresponding to the quantized LSPs, *LSPIDX(k)*

- The adaptive codebook gain index, *ACBGIDX*(*m′*), for each subframe, *m*

- The fixed codebook shape indices, *FCBSIDX*(*m′*), for each subframe, *m′*

- The fixed codebook gain indices, *FCBGIDX*(*m′,k*), for each subframe, *m′*

- The delay transmission code, *DELAY*

- The spectral transition indicator, *LPCFLAG* (SO 3, Rate 1 only)

- The delay difference, *DDELAY* (SO 3, Rate 1 only)

- The residual energy index, *RESEIDX* (SO 70 and SO 73, WB mode only)

- The composite index *K,* which comprises the codebook configuration parameter C_p, the closed loop delay adjustment indices *DELAY_ADJ_INX*(*m′*), and the fixed codebook gain indices, *FCBGIDX*(*m′,k*), for each subframe, *m′* (SO 70 and SO 73, WB mode only)

State Variables Affected:

- The adaptive codebook excitation signal, E(*n*)

- The accumulated shift counter, τ_{acc}

- The pointer to the last sample in the shifted residual, n_m

- The state of the shifted residual buffer, *shiftstate*

- The filter memories in $H_{wq}(z)$ and $H_w(z)$

- The modified residual buffer, $\hat{\varepsilon}(n)$

1 **Initialization:**

2 • The state of the shifted residual buffer, *shiftstate* = CENTER

3 • The quantized LSPs, $\Omega_q(m,k) = 0.048k$; $m = 0$, $1 \leq k \leq 10$

4 **Processing:** EVRC encoding shall comprise the procedures described in 4.11.1 through 4.11.8. A block diagram of the

5 RCELP encoding process is shown in Figure 4.11-1.

6
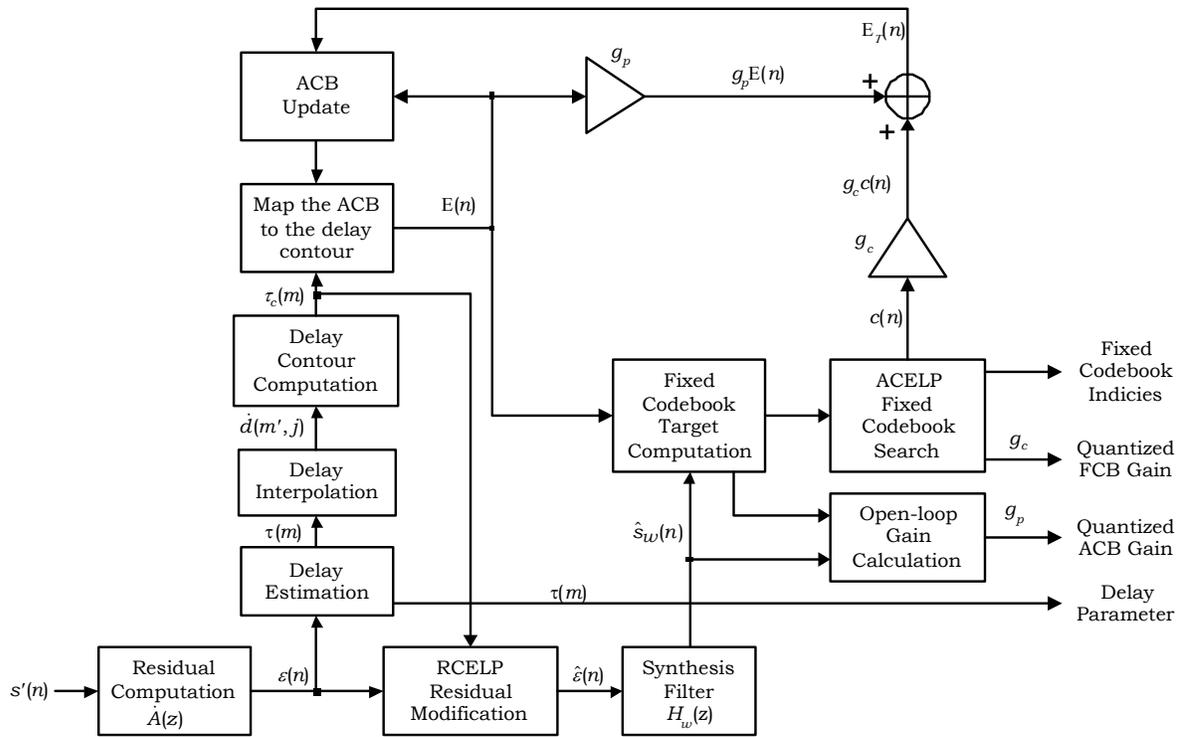
8 **Figure 4.11-1 RCELP Encoding Block Diagram**

9 ## 4.11.1 LSP Quantization

10 The encoder shall vector quantize the unquantized LSPs, $\Omega(m)$, using the procedure found in Sections 4.9 and 4.10 for

11 Rates 1, 1/2, or 1/4.

12 ## 4.11.2 RCELP Shift State Update

13 The encoder shall update the shift state with hysteresis and control the accumulated shift. This step prevents the

14 asynchrony between the original signals and the modified signals from increasing over time. It shall be implemented as

15 a state machine, given by:

16   /* if signal is not at all periodic, reset */

17   if ($\beta < 0.1$) {

18          $\tau_{acc} = 0$

19          $n_m = 0$

1   *shiftstate* = CENTER

2   }

3   /* update shift state with hysteresis */

4   if (τ_{acc} > 20) {

5   shiftstate = RIGHT

6   }

7   if (τ_{acc} < -20) {

8   *shiftstate* = LEFT

9   }

10   if (τ_{acc} <= 10 && *shiftstate* = RIGHT) {

11   *shiftstate* = CENTER

12   }

13   if (τ_{acc} >= -10 && *shiftstate* = LEFT) {

14   *shiftstate* = CENTER

15   }

16   /* control accumulated shift by adjusting the delay */

17   if (*shiftstate* == LEFT && β < 0.4) {

18         $\tau(m) = \tau(m) + 1$

19   }

20   else if (*shiftstate* == RIGHT && β < 0.4) {

21         $\tau(m) = \tau(m) - 1$

22   }

23   /* check to keep delay within bounds */

24      $\tau(m) = \min\{\ \tau(m),\ 120\ \}$

25      $\tau(m) = \max\{\ \tau(m),\ 20\ \}$

4.11.3 Delay Encoding

27 The delay parameter, $\tau(m)$, determined in 4.6.3, shall be encoded for transmission by:

28     $DELAY = \tau(m) - \tau_{\min,}$,   (4.11.3-1)

29 where τ_{min}=20.

30 In Rate 1 processing only, a differential delay parameter shall be transmitted for use in reconstructing the delay contour
31 in the decoder after frame erasures. The transmission code, *DDELAY*, is defined as:

$$DDELAY = \begin{cases} 0 & ; |\Delta_\tau| > 15, \\ \Delta_\tau + 16 & ;\text{otherwise} \end{cases} \tag{4.11.3-2}$$

where

$$\Delta_\tau = \tau(m) - \tau(m-1). \tag{4.11.3-3}$$

4.11.4 Rates 1/2 and 1 Subframe Processing

The procedures described in 4.11.4.1 through 4.11.4.15 shall be carried out for three subframes, $0 \le m' < 3$. The subframe size, L, is 53 for subframes 0 and 1, and 54 for subframe 2.

4.11.4.1 Interpolation of LSP Parameters

Interpolate the unquantized and quantized LSPs over the three subframes, m', in the current frame, m. The form of the unquantized, interpolated LSP vector for subframe m' is given by:

$$\overset{.}{\Omega}(m') = (1 - \mu_{m'})\Omega(m-1) + \mu_{m'}\Omega(m), \tag{4.11.4-1}$$

and form of the quantized, interpolated LSP vector is given by:

$$\overset{.}{\Omega}_q(m') = (1 - \mu_{m'})\Omega_q(m-1) + \mu_{m'}\Omega_q(m), \tag{4.11.4-2}$$

where the subframe interpolator constants are defined as $\mu = \{ 0.1667, 0.5, 0.8333 \}$.

4.11.4.2 LSP to LPC Conversion

Convert the unquantized, interpolated LSPs, $\overset{.}{\Omega}(m')$, to unquantized, interpolated LPC parameters, $\{\dot{a}\}$, as described in 4.6.2.2 for each subframe. Convert the quantized, interpolated LSPs, $\overset{.}{\Omega}_q(m')$, to quantized, interpolated LPC parameters, $\{\dot{a}_q\}$, as described in 4.6.2.2, using the quantized LSPs at the end-points instead of the unquantized LSPs.

4.11.4.3 Interpolated Delay Estimate Calculation

Calculate a set of three interpolated delay estimates, $\overset{.}{d}(m', j)$, for each subframe m' as follows:

$$\overset{.}{d}(m', j) = \begin{cases} \tau(m) & ; |\tau(m) - \tau(m-1)| > 15, \\ (1 - f(m'+j))\tau(m-1) + f(m'+j)\tau(m) & ;\text{otherwise}, \end{cases} \quad 0 \le m' < 3, \, 0 \le j < 3, \tag{4.11.4-3}$$

where m is the current frame, m' is the subframe index, and $m'+j$ is an index into an array of interpolator coefficients, f, defined by:

$$f = \{ 0.0, 0.3313, 0.6625, 1.0, 1.0 \}. \tag{4.11.4-4}$$

4.11.4.4 Modification of the Original Residual

Generate a modified residual signal, $\hat{\varepsilon}(n)$, by shifting the current residual signal, $\varepsilon(n)$, to match the delay contour for the current subframe. See Section 4.11.6 for details on this operation.

1   4.11.4.5 Generation of the Weighted Modified Original Speech Vector

2   Generate the weighted modified original speech vector, $\hat{s}_w(n)$, by filtering the modified residual, $\hat{\varepsilon}(n)$, through the

3   weighted synthesis filter $H_w(z)$. For narrowband (8 kHz sampled) inputs, $H_w(z)$ is given by:

$$H_w(z) = \frac{1}{\dot{A}(z)} \left[\frac{\dot{A}(\gamma_1^{-1}z)}{\dot{A}(\gamma_2^{-1}z)} \right],$$

(4.11.4-5)

5   where

$$\dot{A}(z) = 1 - \sum_{k=1}^{10} \dot{a}(k)z^{-k},$$

(4.11.4-6)

7   and

$$\dot{A}(\gamma_1^{-1}z) = 1 - \sum_{k=1}^{10} \dot{a}(k)\gamma_l^k z^{-k} \quad ; l=1,2,$$

(4.11.4-7)

9   where $\gamma_1 = 0.9, \gamma_2 = 0.5$.

10   The weighted synthesis filter for wideband (16 kHz sampled) inputs includes harmonic noise weighting (HNW), which

11   overcomes some of the limitations of low rate coding [R14]. HNW efficiently utilizes noise-masking properties by

12   exploiting the speech signal's harmonic structure. The HNW filter is given by:

$$C(z) = 1 - \varepsilon_p g'_p z^{-\tau'}$$

(4.11.4-8)

14   where $\tau' = (d(m',0) + d(m',1))/2$ is the average subframe delay (sec 4.11.4.3), and ε_p is the HNW coefficient given as

15   a decreasing function of the average subframe delay as:

$$\varepsilon_p = \begin{cases} \varepsilon_{\min}, & \tau' \ge \tau'_{\max} \\ \varepsilon_{\min} + \Delta\dfrac{(\tau'_{\max} - \tau')}{\tau'_{\max}}, & \tau' \ge \left(1 - \dfrac{\varepsilon_{\max} - \varepsilon_{\min}}{\Delta}\right)\tau'_{\max} \\ \varepsilon_{\max}, & otherwise \end{cases}$$

(4.11.4-9)

17   where $\varepsilon_{\max} = 0.6$ is the maximum allowable value of the HNW coefficient, $\varepsilon_{\min} = 0$ is the minimum allowable value of

18   the HNW coefficient, $\tau'_{\max} = 110$ is the maximum pitch delay above which HNW coefficient is set to $\varepsilon_{\min}$, and $\Delta = 1.1$

19   is the slope for the HNW coefficient. The HNW gain g'_p is found by:

$$g'_p = \frac{\sum_{n=0}^{L-1} \hat{s}_{sw}(n)\hat{s}_{sw}(n - \tau')}{\sum_{n=0}^{L-1} \hat{s}_{sw}^2(n)},$$

(4.11.4-10)

21   where $0 \le g'_p \le 1.0$, and $\hat{s}_{sw}(n)$ is the spectrally weighted modified speech which is obtained by filtering the modified

22   residual $\hat{\varepsilon}(n)$ through the spectral weighting filter given by:

$$H_{sw}(z) = \frac{1}{\dot{A}(z)} \left[\frac{\dot{A}(\gamma_1^{-1}z)}{1 - \mu z^{-1}} \right],$$

(4.11.4-11)

24   where $\gamma_1 = 0.9$ and $\mu = 0.48$ is the tilt coefficient. The signal $\hat{s}_w(n)$ is finally obtained by filtering $\hat{s}_{sw}(n)$ through the

25   composite HNW filter $C(z)$. To generate the fractional pitch delay $z^{-\tau'}$ part of the HNW filter, an interpolation is

performed using an 8x oversampled FIR low pass filter with a windowed sinc function truncated at ±31 and a cut-off frequency at approximately 1000 Hz.

4.11.4.6 Zero Input Response Calculation

Calculate the zero-input response, $a_{zir}(n)$, of the weighted synthesis filter, $H_{wq}(z)$. The weighted synthesis filter $H_{wq}(z)$ for narrowband (8 kHz sampled) inputs is defined as:

$$H_{wq}(z) = \frac{1}{\dot{A}_q(z)} \left[\frac{\dot{A}(\gamma_1^{-1}z)}{\dot{A}(\gamma_2^{-1}z)} \right] \tag{4.11.4-12}$$

where

$$\dot{A}_q(z) = 1 - \sum_{k=1}^{10} \dot{a}_q(k)z^{-k}, \tag{4.11.4-13}$$

and other symbols are previously defined. The weighted synthesis filter for wideband (16 kHz sampled) inputs is given by:

$$H_{wq}(z) = \frac{1}{\dot{A}_q(z)} \left[\frac{\dot{A}(\gamma_1^{-1}z)}{1 - \mu z^{-1}} C(z) \right], \tag{4.11.4-14}$$

where all symbols have been previously defined.

Then subtract the zero-input response, $a_{zir}(n)$, of the weighted synthesis filter, $H_{wq}(z)$, from the weighted modified original speech vector, $\hat{s}_w(n)$ to produce the weighted target speech vector:

$$\hat{s}_w(n) = \hat{s}_w(n) - a_{zir}(n) \qquad ; 0 \le n < L. \tag{4.11.4-15}$$

4.11.4.7 Impulse Response Calculation

Calculate the impulse response, $h_{wq}(n)$, of the weighted synthesis filter, $H_{wq}(z)$, to 54 terms.

4.11.4.8 Calculation of the Adaptive Codebook Contribution

Compute the adaptive codebook contribution, $E(n)$, the filtered adaptive codebook contribution, $\lambda(n)$, and the unquantized ACB gain, g_{p_unq}, described in 4.11.5. For SO 70 and SO 73 in wideband operation, the interpolated delay estimates $\dot{d}(m', j)$ may be modified to reflect closed loop delay contour adjustment, in which case, the codebook configuration parameter C_p will be set, and the delay adjustment indices will be contained in *DELAY_ADJ_INX(m')*.

4.11.4.9 Closed-Loop Gain Calculation

The closed-loop quantized gain, g_p, is calculated by first determining the cross-correlation, $R_{s\lambda}(0)$, of $\hat{s}_w(n)$ and $\lambda(n)$:

$$R_{s\lambda}(0) = \sum_{n=0}^{L-1} \hat{s}_w(n)\lambda(n), \tag{4.11.4-16}$$

the auto-correlation of $\lambda(n)$ is then calculated by:

1
$$R_{\lambda\lambda}(0) = \sum_{n=0}^{L-1} \lambda^2(n).$$
(4.11.4-17)

2 The quantized closed loop gain, g_p, is defined by:

3
$$g_p = \begin{cases} g_{pcb}(0) & ;R_{s\lambda}(0) \le 0, \\ g_{pcb}(ACBGIDX(m')) & ;R_{s\lambda}(0) > 0, \end{cases}$$
(4.11.4-18)

4 where $g_{pcb}(k)$ is an entry in the adaptive codebook gain quantization Table 4.11.4.9-1, and $ACBGIDX(m')$ is the optimal
5 index into $g_{pcb}(k)$ calculated by:

6 $\quad ACBGIDX(m') = 0$

7 $\quad$ for $(k = 1; k < g_{pcb\_size}; k = k + 1)$ {

8 $\quad\quad$ if $\left(R_{s\lambda}(0) > \left(\dfrac{g_{pcb}(k) + g_{pcb}(k-1)}{2} R_{\lambda\lambda}(0) \right) \right)$ {

9 $\quad\quad\quad ACBGIDX(m') = k$

10 $\quad\quad$ }

11 $\quad$ }

12 $\quad$ where $g_{pcb\_size} = 8$.

13

14 **Table 4.11.4.9-1 Adaptive Codebook Gain Quantization Table**

k	$g_{pcb}(k)$
0	0.00
1	0.30
2	0.55
3	0.70
4	0.80
5	0.90
6	1.00
7	1.20

15

16 ## 4.11.4.10 Fixed Codebook Search Target Vector Generation

17 ## 4.11.4.10.1 Perceptual Domain Target Vector

18 For SO 3, SO 68, SO 70, and SO 73 narrowband interoperability mode, calculate the target vector in the perceptual
19 domain, $x_w(n)$, defined by:

20 $\quad x_w(n) = \hat{s}_w(n) - g_p \lambda(n) \quad ;0 \le n < L,$ (4.11.4-19)

21 where $\hat{s}_w(n)$ and $\lambda(n)$ are defined in 4.5.4.9. For SO 70 and SO 73 wideband mode, calculate the target vector in the
22 perceptual domain, $x_w(n)$, as:

1

$$x_w(n) = \begin{cases} \hat{s}_w(n) - g_p \lambda(n) & ; \dot{\tau} < L \\ \hat{s}_w(n) - g_{p\_unq} \lambda(n) & ; \dot{\tau} \geq L \end{cases}, \quad 0 \leq n < L,$$

(4.11.4-20)

2   where $\dot{\tau}$ is the average subframe delay given in Equation 4.11.4-22.

3   The unquantized ACB gain g_{p_unq} is used in cases where the joint optimization of ACB/FCB gains is performed (see
4   4.11.8.2)

5 **4.11.4.10.2 Conversion of the Target Vector to the Residual Domain**

6   For SO 3, SO 68, SO 70, and SO 73 narrowband interoperability mode, convert the target vector from the perceptual
7   domain to the residual domain by filtering $x_w(n)$ through the zero-state inverse of the weighting filter, $H_{wq}(z)$:

8

$$H_{wq}^{-1}(z) = \dot{A}_q(z) \frac{\dot{A}(\gamma_2^{-1} z)}{\dot{A}(\gamma_1^{-1} z)},$$

(4.11.4-21)

9   and all other terms are defined in 4.11.4.3.

10   The resulting target vector in the residual domain is defined as $x(n)$.

11 **4.11.4.10.3 Delay Calculation for Current Subframe**

12   The fixed codebook search requires an estimate of the delay, $\dot{\tau}$, for the current subframe, defined by:

13

$$\dot{\tau} = round\left\{ \frac{\dot{d}(m',0) + \dot{d}(m',1)}{2} \right\},$$

(4.11.4-22)

14   where $\dot{d}(m',j)$ is defined in 4.11.4.3.

15 **4.11.4.11 Fixed Codebook Search**

16   Perform the fixed codebook search as described in 4.11.7.

17 **4.11.4.12 Fixed Codebook Gain Quantization**

18   The fixed codebook gain, g_c, computed in 4.11.4.11, is constrained by:

19

$$g_c = \begin{cases} (1.0 - 0.15 g_p) g_c & ; \text{Rate}(m) == \text{Rate1}, \\ (0.9 - 0.1 g_p) g_c & ; \text{Rate}(m) == \text{Rate1/2}, \end{cases}$$

(4.11.4-23)

20   which is then quantized (for SO 3, SO 68, SO 70, and SO 73 narrowband interoperability mode) by the following
21   process:

22       $FCBGIDX(m') = 0$

23   for ($k = 1; k < g_{ccb\_size}; k = k + 1$)

24   if $\left(g_c > \left(\frac{g_{ccb}(k) + g_{ccb}(k-1)}{2} \right) \right)$ {

25             $FCBGIDX(m') = k$

1
$$\}$$

2
$$g_c = g_{ccb}(FCBGIDX(m'))$$

3

4 where $g_{ccb}(k)$ is a fixed codebook gain quantization entry defined in Table 8-15 and Table 8-16 for Rate 1 and Rate 1/2,

5 respectively, and $FCBGIDX(m')$ is the optimal index into the appropriate table for each subframe m'.

6 For SO 70 and SO 73 wideband mode, the FCB gain is quantized using the residual energy of the current frame as a first

7 order prediction estimate of the FCB gain. The prediction error for each subframe is then quantized according to a

8 minimum mean squared error criteria. The residual energy for the current frame is calculated as:

9
$$E_r = \sqrt{\sum_{n=80}^{239} \varepsilon^2(n)} ,$$
(4.11.4-24)

10 which is then quantized using 3 bits according to the following expression:

11
$$E_{rq} = \exp\left(\min_{i \in 0,...,7} \left((\ln(E_r) - (4.0 + 7.5i/8.0))^2 \right) \right),$$
(4.11.4-25)

12 and where $RESEIDX = i$ is the index corresponding the minimum log domain distortion. The FCB gain is then given by:

13
$$g_c = g_{ccb2}(FCBGIDX(m'))E_{rq}\sqrt{\frac{1 - g_p^2/3}{\sum_{i=0}^{L-1} c_k^2(i)}} ,$$
(4.11.4-26)

14 where $\mathbf{c}_k$ is the FCB excitation vector, g_p is the quantized ACB gain from Table 4.11.4.9-1, and $g_{ccb2}(i)$ is the i-th

15 quantized FCB correction factor, as defined by the 16 element lookup table:

16
$$\mathbf{g}_{ccb2} = \{0.0251, 0.0321, 0.0410, 0.0524, 0.0671, 0.0857, 0.1096, 0.1401,$$
$$0.1791, 0.2290, 0.2928, 0.3743, 0.4786, 0.6118, 0.7822, 1.0000\} .$$
(4.11.4-27)

17 From the given error expression $e(n) = \hat{s}_w(n) - g_p\lambda(n) - g_c y_2(n)$, where $y_2(n) = h_{wq}(n) * c_k(n)$ is the filtered FCB

18 excitation, the optimal FCB/ACB gains are determined by solving for the minimum squared error, according to the

19 following expression:

20
$$\min\left\{ \hat{s}_w^t \hat{s}_w - 2g_p \hat{s}_w^t \lambda - 2g_c \hat{s}_w^t \mathbf{y}_2 - 2g_p g_c \lambda^t \mathbf{y}_2 + g_p^2 \lambda^t \lambda + g_c^2 \mathbf{y}_2^t \mathbf{y}_2 \right\}.$$
(4.11.4-28)

21 For cases when pitch sharpening is not active (i.e., $\dot{\tau} \geq L$), all possible values g_p and g_c (Table 4.11.4.9-1 and Equation

22 4.11.4-26 respectively) are evaluated to produce the minimum distortion. For cases when pitch sharpening is active

23 (i.e., $\dot{\tau} < L$), the ACB gain g_p has already been quantized, and only the terms containing the FCB gain g_c are required to

24 be calculated.

25 ## 4.11.4.13 Combined Excitation Vector Computation

26 The combined excitation vector, $E_T(n)$, for the current subframe, m' is defined as the sum of the adaptive codebook

27 contribution, $E(n)$, and the fixed codebook contribution, $c(n)$:

28
$$E_T(n) = g_p E(n) + g_c c(n) \quad ; 0 \leq n < L.$$
(4.11.4-29)

4.11.4.14 Excitation Input to Highband Analysis (SO 70 and SO 73 Wideband Mode)

The combined excitation vector, $E_T(n)$, for the current subframe, m' is filtered through a pitch enhancement filter prior to input to highband analysis in 4.18. The pitch enhancement filter is a time varying filter of the form:

$$P(z,n) = \frac{1}{1 - 0.4 g_p z^{-\tau_c(n)}} ; \quad 0 \le n < L ,$$

(4.11.4-30)

where $\tau_c(n)$ is the delay contour obtained from interpolating the delay adjusted endpoints, $\dot{d}(m',0)$ to $\dot{d}(m',1)$, as in 4.11.5.1. The filter uses an 8x oversampling fractional pitch interpolation similar to that used for generating the adaptive codebook contribution. This notation, however, is a mix of time domain and z-transform symbols, where the z^{-1} operator is to be viewed as a unit delay operator rather than a z-transform. The state of this filter is reset to all zeros on subframe $m' = 0$ when the previous frame was not RCELP Rate 1 or Rate 1/2. In addition, the energy of the output of the pitch enhancement filter is normalized to the input energy.

4.11.4.15 Encoder State Variable Update

Update the weighted synthesis filter memory by filtering the combined excitation vector, $E_T(n)$, through the weighted synthesis filter, $H_{wq}(z)$, which is given in Equation 4.11.4-12.

Update the modified residual memory by copying n_m elements of the current modified residual to the beginning of the modified residual buffer:

$$\hat{\varepsilon}(n) = \hat{\varepsilon}(n+L); \quad 0 \le n < n_m .$$

(4.11.4-31)

Update the excitation memory for the adaptive codebook with the combined excitation for the current subframe:

$$E(n) = \begin{cases} E(n+L) & ; -128 \le n < -L \\ E_T(n+L) & ; -L \le n < 0 \end{cases}$$

(4.11.4-32)

Conditionally reset the accumulated shift counter, τ_{acc}, and the pointer to the last sample in the shifted residual, n_m, according to the following:

$$\text{if } \left(\left(\sum_{i=0}^{2} ACBGIDX(i) \le 1 \right) and \left(ACBGIDX(2) \neq 1 \right) \right) \ \{$$

$$\tau_{acc} = 0$$

$$n_m = 0$$

$$\}$$

4.11.5 Computation of the Adaptive Codebook Contribution

Inputs: The inputs to the adaptive codebook computation are:

- The vector of interpolated delays, $\dot{d}(m',j)$, calculated in 4.11.4.3

- The current subframe number, m'

- The current subframe size, L

And for SO 70 and SO 73 wideband inputs:

1. • The weighted target speech vector, $\hat{s}_w(n)$

2. • The open loop delay estimates for the current and previous frames, $\tau(m)$ and $\tau(m\text{-}1)$, respectively

3. **Outputs:**

4. • The adaptive codebook excitation, $E(n)$, where $\text{-}128 \le n \le 64$

5. • The filtered adaptive codebook excitation, $\lambda(n)$

6. And for SO 70 and SO 73 wideband inputs:

7. • The codebook configuration parameter C_p

8. • The closed loop delay adjustment indices, $DELAY\_ADJ\_INX(m')$

9. • The adjusted vector of interpolated delay endpoints, $\dot{d}(m', j)$

10. • The unquantized constrained optimum ACB gain, g_{p_unq}

11. **Initialization:** The variables used in the computation of the adaptive codebook contribution are initialized as follows:

12. • The adaptive codebook excitation memory, $E(n) = 0$; $\text{-}128 \le n \le 64$

13. **Processing:** Computation of the adaptive codebook contribution for SO 3, SO 68, SO 70, and SO 73 narrowband
14. interoperability mode shall be performed according to the procedures described in 4.11.5.1 and 4.11.5.3. For SO 70 and
15. SO 73 wideband mode, a second adaptive codebook is added that performs closed loop delay contour adjustment. A
16. codebook configuration parameter C_p is used to indicate which of 2 ACB/FCB combinations is active, and if the delay
17. contour adjustment ACB is used, a 2 bit delay adjustment index $DELAY\_ADJ\_INX(m')$ is provided to indicate which of 4
18. possible adjusted delay contours has been selected. These processes are described in the following sections.

4.11.5.1 Delay Contour Computation

20. Compute the 8-times oversampled delay contour vector, $\tau_c(n)$, defined by:

21.
$$\tau_c(n) = \begin{cases} \dot{d}(m',0) + \dfrac{n\left[\dot{d}(m',1) - \dot{d}(m',0)\right]}{L} & ;0 \le n < L, \\[4mm] \dot{d}(m',1) + \dfrac{(n-L)\left[\dot{d}(m',2) - \dot{d}(m',1)\right]}{L} & ;L \le \text{n} < (L+10) \end{cases}$$
(4.11.5-1)

22. where m' is the current subframe, L is the current subframe size, $\dot{d}(m', j)$ is a set of interpolated delays calculated in
23. 4.11.4.3.

4.11.5.2 Mapping of the Adaptive Codebook to the Delay Contour

25. The new adaptive codebook contribution, $E(n)$, is calculated by:

26.
$$E(n) = \sum_{i=0}^{2f_l} E(i + n - T_E(n) - f_l)I_E\left(i + (2f_l + 1)T_{IE}(n)\right) \quad ;0 \le n < (L+10),$$
(4.11.5-2)

27. where $T_E(n)$ and $T_{IE}(n)$ are calculated for every n by:

$$T_E(n) = \begin{cases} round\{\tau_c(n)\} & ; \tau_c(n) > 0, \\ -round\{-\tau_c(n)\} & ; \tau_c(n) \le 0, \end{cases}$$

$$T_{IE}(n) = trunc\{(T_E(n) - \tau_c(n) + 0.5)R + 0.5\},$$

if ($T_{IE}(n) == R$) {

$\qquad T_{IE}(n) = 0$

$\qquad T_E(n) = T_E(n) - 1$

$\qquad$ }

and L is the current subframe size, $f_l = 8$, $R = 8$, and $\{I_E(n)\}$ is a set of interpolation coefficients found in Table 8-12 through Table 8-14. Then convolve the impulse response, $h_{wq}(n)$, with the adaptive codebook excitation, $E(n)$, generating the filtered adaptive codebook excitation, $\lambda(n)$:

$$\lambda(n) = \sum_{j=0}^{n} h_{wq}(j)E(n-j) \quad ; 0 \le n < L. \tag{4.11.5-3}$$

4.11.5.3 Closed Loop Delay Contour Adjustment

The Closed Loop Delay Contour Adjustment process utilizes an adaptive step size that is based on the variability of the open loop delay parameter $\tau(m)$. For the variabilty estimate, the sample standard deviation of the current and previous frame's delays are used, which is given as:

$$\sigma_\tau = \begin{cases} \dfrac{1}{\sqrt{2}}|\tau(m) - \tau(m-1)| & ; Rate(m-1) = \text{Rate}1 \\ 0 & ; otherwise \end{cases} \tag{4.11.5-4}$$

The variability estimate σ_τ and the open-loop pitch delay $\tau(m)$ are then used to calculate the adaptive step size $\delta(m)$ as:

$$\delta(m) = \alpha(\sigma_\tau)\left(\frac{\tau(m) + \tau(m-1)}{2}\right) \tag{4.11.5-5}$$

where $\alpha(\sigma_\tau)$ is a function of the variability estimate of pitch delay given as:

$$\alpha(\sigma_\tau) = \min\{0.0020\sigma_\tau + 0.0025, 0.2\}. \tag{4.11.5-6}$$

The adaptive step-size $\delta(m)$ is then used to generate the delay adjust values $\Delta_{adj}(i)$ calculated as a function of the delay adjust index i as:

$$\Delta_{adj}(i) = f(i)\delta(m); \quad f(i) = \{+1, -1, +2, -2\}, \quad 0 \le i \le 3. \tag{4.11.5-7}$$

The delay adjust values are then used to generate a set of candidate delay contours for the subsequent closed loop search. The four candidates are generated by moving the endpoints of the delay contour up or down according to the following expression:

$$d'(m', j, i) = \min\{\max\{d(m', j) + \Delta_{adj}(i), 20\}, 120\}; \quad 0 \le i \le 3 \tag{4.11.5-8}$$

and then use these endpoints in the calculation of the delay contour and adaptive codebook contributions, as in Equations 4.11.5-1 and 4.11.5-2 respectively, to produce the delay adjusted adaptive codebook contributions $\mathbf{E}(i, n)$, $0 \le i \le 3$. These signals are then convolved with the impulse response $h_{wq}(n)$ to produce the candidate filtered ACB excitation vectors $\lambda(i, n)$, $0 \le i \le 3$.

In order to evaluate the candidate vectors from the two ACBs, the delay contour adjusted ACB must perform significantly better than the default ACB in order to justify dynamically reallocating bits from the FCB to the ACB. Therefore, a modified mean squared error criteria is used, along with a prediction gain metric, in the closed loop adjusted delay contour selection process. The baseline performance metrics for the default ACB are given as:

$$MSE = \sum_{n=0}^{L-1} \left(\hat{s}_w(n) - g_{p\_unq} \lambda(n) \right)^2 , \tag{4.11.5-9}$$

where g_{p_unq} is the unquantized constrained optimum ACB gain given by:

$$g_{p\_unq} = \min\left\{ \max\left\{ \frac{\sum_{n=0}^{L-1} \hat{s}_w(n) \lambda(n)}{1 + \sum_{n=0}^{L-1} \lambda^2(n)}, 0.0 \right\}, 1.2 \right\} . \tag{4.11.5-10}$$

By clamping the gain to $0 \le g_{p\_unq} \le 1.2$, the *MSE* metric becomes more reliable in that only error metrics with quantifiable ACB gains are considered. Also, the prediction gain metric, defined by:

$$PGM = \frac{\sum_{n=0}^{L-1} \hat{s}_w^2(n)}{MSE} , \tag{4.11.5-11}$$

further increases the reliability of the ACB candidate selection by allowing another dimension of the ACB performance to be evaluated. Given this background information, the process for selecting the ACB candidate vector is given in the following pseudo code:

$C_p = 0$	/* Indicate use of default ACB */
$MSE_{best} = MSE/1.1$	/* Set best MSE to default divided by a bias */
for $i = 0$ to 3 {	/* Evaluate Delay Contour Adjustment ACB
if $MSE_{\lambda(i,n)} < MSE_{best}$ and $PGM_{\lambda(i,n)} > 1.2$ {	/* *MSE* and *PGM* for ACB candidate $\lambda(i, n)$ */
$C_p = 1$	/* Indicate use of Delay Contour Adjustment ACB */
$DELAY\_ADJ\_INX(m') = i$	/* Update delay adjust index */
$MSE_{best} = MSE_{\lambda(i,n)}$	/* Update best *MSE* */
$\dot{d}(m',j) \leftarrow d'(m',j,i)$	/* Update delay contour endpoints */
$\mathbf{E}(n) \leftarrow \mathbf{E}(i,n)$	/* Update unfiltered ACB excitation */
$\lambda(n) \leftarrow \lambda(i,n)$	/* Update filtered ACB excitation */
$g_{p\_unq} = g_{p\_unq}(i)$	/* Update the unquantized ACB gain */
}	/* end if */
}	/* end for */

4.11.6 Modification of the Residual

Inputs: The inputs to the residual modification are:

1. • The vector of interpolated delays, $\dot{d}(m', j)$, calculated in 4.11.4.3

2. • The short-term prediction residual, $\varepsilon(n)$

3. • The pitch prediction gain, β

4. • The current subframe number, m'

5. • The current subframe size, L

6. **Outputs:** The outputs from the residual modification are:

7. • The modified residual signal, $\hat{\varepsilon}(n)$

8. **State Variables Affected:**

9. • The modified target residual buffer, $\hat{\varepsilon}_t(n)$, where $-128 \leq n \leq 64$

10. • The pointer to the end of the last residual frame modified, n_m

11. • The accumulated shift, τ_{acc}

12. **Initialization:** The variables used in the residual modification are initialized as follows:

13. • The modified target residual buffer, $\hat{\varepsilon}_t(n) = 0$; $-128 \leq n \leq 64$

14. • The pointer to the end of the last residual frame modified, $n_m = 0$

15. • The accumulated shift, $\tau_{acc} = 0$

16. **Processing:** The modification of the residual signal to generate a target residual for the fixed codebook search shall be
17. computed according to 4.11.6.1 through 4.11.6.5. A shifted target residual, $\hat{\varepsilon}_t(n)$, is generated in 4.11.6.1 using the past
18. modified residual buffer and the delay contour of the current frame. This shifted residual is used as a target for shifting
19. the residual of the current subframe. All the pitch pulses in the original residual, $\varepsilon(n)$, must be shifted individually to
20. match the delay contour of the modified target residual, $\hat{\varepsilon}_t(n)$. The modification is performed by finding a shift frame
21. containing each pulse in the original residual by the process described in 4.11.6.2. Once the shift frame is found, the
22. accumulated shift, τ_{acc}, must be adjusted as described in 4.11.6.3 to determine the proper shift needed to match the pulse
23. to the delay contour. The shift frame containing the pulse is then shifted by τ_{acc} as described in 4.11.6.4. The procedures
24. described in 4.11.6.2 through 4.11.6.5 must be carried out for all pulses found in the residual of the current subframe.

4.11.6.1 Mapping of the Past Modified Residual to the Delay Contour

26. Calculate the target for the residual modification process by warping the past modified residual buffer, $\hat{\varepsilon}_t(n)$, by the
27. delay contour, $\tau_c(n)$, calculated in 4.11.5.1.

28. The new target residual buffer, $\hat{\varepsilon}_t(n)$, is calculated by:

29.
$$\hat{\varepsilon}_t(n) = \sum_{i=0}^{2f_l} \hat{\varepsilon}_t(i + n - T_\varepsilon(n) - f_l)I_\varepsilon\left(i + (2f_l + 1)T_{I\varepsilon}(n)\right) \quad ;0 \leq n < (L+10),$$
(4.11.6-1)

30. where $T_\varepsilon(n)$ and $T_{I\varepsilon}(n)$ are calculated for every n by:

31.
$$T_\varepsilon(n) = \begin{cases} round\left\{\tau_c(n)\right\} & ;\tau_c(n) > 0, \\ -round\left\{-\tau_c(n)\right\} & ;\tau_c(n) \leq 0, \end{cases}$$

$$T_{I\varepsilon}(n) = trunc\left\{\left(T_\varepsilon(n) - \tau_c(n) + 0.5\right)R + 0.5\right\},$$

if $((T_{I\varepsilon}n) == R)$

$\{$

$\quad T_{I\varepsilon}(n) = 0$

$\quad T_\varepsilon(n) = \text{T}_\varepsilon(n) - 1$

$\}$

and L is the current subframe size, $f_l = 3$, $R = 8$, $\{I_\varepsilon(n)\}$ is a set of interpolation coefficients found in Table 8-10 and Table 8-11, and $\tau_c(n)$ is computed in Equation 4.11.5-1.

4.11.6.2 Calculation of the Residual Shift Frame Parameters

First define the current subframe residual as:

$$\varepsilon_s(n) = \varepsilon(n + 80 + 53m') \quad ;0 \le n < 240\text{-}53\text{m}', \tag{4.11.6-2}$$

and proceed with the following sections.

4.11.6.2.1 Search for Pulses in the Subframe Residual

Calculate the location, n_{emax}, that contains the pulse having maximum energy in the subframe residual. The search window is centered at 1/2 the pitch lag, $\dot{d}(m',1)$, after the last modified residual sample, and is 1.5 times the pitch lag in length. This location, n_{emax}, is defined as the location, n, that maximizes:

$$E_\varepsilon(n) = \sum_{i=-2}^{2} \varepsilon_s^2(T + n + i) \quad ;0 \le n < 1, \tag{4.11.6-3}$$

where the index of the start of the energy search, T, is defined by:

$$T = \max\left\{\left(n_m + trunc\{-\tau_{acc} + 0.5\} - trunc\left\{\frac{1}{4}\dot{d}(m',1)\right\}\right), (\text{-78 - 53m'})\right\}, \tag{4.11.6-4}$$

where n_m is an index to the last residual sample modified, and τ_{acc} is the accumulated shift; the size of the window to be searched, l, is defined by:

$$l = \min\left\{trunc\left\{1.5\dot{d}(m',1)\right\}, 238 - 53m' - T\right\}. \tag{4.11.6-5}$$

Adjust n_{emax} by τ_{acc}:

$$n_{emax} = n_{emax} + T - trunc\{-\tau_{acc} + 0.5\}. \tag{4.11.6-6}$$

4.11.6.2.2 Location of the First Pulse in the Residual

The location of maximum pulse energy, n_{emax}, is checked to insure it is the first pitch pulse in the residual. If ($n_{emax} < n_m$), the pulse located is in a region that already has been modified, so the residual must be searched again using a smaller window size. The new search window is centered one pitch lag, $\dot{d}(m',1)$, after the pulse location found in 4.11.6.2.1, and is one half of a pitch lag in length. The new location, n_{emax}, is defined as the location, n, that maximizes:

1

$$E_\varepsilon(n) = \sum_{i=-2}^{2} \varepsilon_s^2(T+n+i) \quad ; 0 \le n < 1,$$

(4.11.6-7)

2 where the index of the start of the energy search, *T*, is defined by:

3

$$T = \max\left\{\left(n_{emax} + trunc\{-\tau_{acc} + 0.5\} + trunc\left\{\frac{3}{4}\dot{d}(m',1) + 0.5\right\}\right), \left(-78 - 53\text{m}'\right)\right\},$$

(4.11.6-8)

4 where n_{emax} is the index of the pulse located in 4.11.6.2.1, τ_{acc} is the accumulated shift; the size of the window to be

5 searched, *l*, is defined by:

6

$$l = \min\left\{trunc\left\{1.5\dot{d}(m',1)\right\}, 238 - 53m' - T\right\}.$$

(4.11.6-9)

7 The location of the maximum pulse, n_{emax}, is then re-defined and adjusted by:

8

$$n_{emax} = n_{emax} + T - trunc\{-\tau_{acc} + 0.5\}.$$

(4.11.6-10)

9 ### 4.11.6.2.3 Location of a Pulse Inside of the Lag Window

10 If $n_{emax} > \left(n_m + \dot{d}(m',1)\right)$, where n_{emax} was determined in 4.11.6.2.1 or 4.11.6.2.2, the pulse found has a larger lag than

11 expected. A final pulse energy search of the lag window must be made to insure that the pulse found at n_{emax} is the

12 desired pulse to be shifted. The new search window is centered one pitch lag, $\dot{d}(m',1)$, before the pulse location found in

13 4.11.6.2.1 or 4.11.6.2.2, and is one half of a pitch lag in length. The location, n''_{emax}, of the pulse with maximum energy

14 in the current lag window is defined as the location, *n*, that maximizes:

15

$$E_\varepsilon(n) = \sum_{i=-2}^{2} \varepsilon_s^2(T+n+i) \quad ; 0 \le n < 1,$$

(4.11.6-11)

16 where the index of the start of the energy search, *T*, is defined by:

17

$$T = \max\left\{\left(n_{emax} + trunc\{-\tau_{acc} + 0.5\} - trunc\left\{\frac{5}{4}\dot{d}(m',1) + 0.5\right\}\right), \left(-78 - 53\text{m}'\right)\right\},$$

(4.11.6-12)

18 where n_{emax} is the index of the pulse determined in 4.11.6.2.1 or 4.11.6.2.2, τ_{acc} is the accumulated shift; the size of the

19 window to be searched, *l*, is defined by:

20

$$l = \min\left\{trunc\left\{0.5\dot{d}(m',1)\right\}, 238 - 53m' - T\right\}$$

(4.11.6-13)

21 The location, n''_{emax}, of the maximum pulse in the lag window, is adjusted by τ_{acc}:

22

$$n''_{emax} = n''_{emax} + T - trunc\{-\tau_{acc} + 0.5\},$$

(4.11.6-14)

23 The location of the pulse to be shifted, n_{emax}, is then re-defined by:

24

$$n_{emax} = \begin{cases} n''_{emax} & ; n''_{emax} \ge n_m, \\ n_{emax} & ; \text{otherwise.} \end{cases}$$

(4.11.6-15)

4.11.6.2.4 Shift Frame Boundary Calculation

The boundaries, T_{start} and T_{end}, for the frame in the residual to be shifted are calculated by:

$$T_{start} = n_m,$$ (4.11.6-16)

$$T_{end} = \begin{cases} L & ;L\text{-}10 < n_{emax} < L-5, \\ L+10 & ;L<n_{emax} < L+5, \\ L & ;n_{emax} \geq L+5, \\ n_{emax}+10 & ;\text{otherwise.} \end{cases}$$ (4.11.6-17)

4.11.6.2.5 Shift Decision

Calculate a shift decision flag, ε_{shift}, by:

$$\varepsilon_{shift} = \begin{cases} FALSE & ;\left(n_{emax} \geq T_{end}\right) or \left(n_{emax} < T_{start}\right), \\ TRUE & ;\text{otherwise.} \end{cases}$$ (4.11.6-18)

4.11.6.2.6 Peak to Average Ratio Calculation

If the shift decision flag is true ($\varepsilon_{shift} == TRUE$) as evaluated in 4.11.6.2.5, the ratio of the peak energy to the average energy in the residual frame to be shifted must be calculated to insure that the frame contains a valid pulse. The residual frame to be shifted is defined as $\varepsilon_s(n)$, where $T_{start} \leq n \leq T_{end}$ (see 4.5.6.2 through 4.5.6.4).

Calculate a vector of smoothed residual energies, $E_{win}(n)$, by:

$$E_{win}(n) = \sum_{i=0}^{4} \varepsilon_s^2\left(T_{adj} + i + n\right), \qquad 0 \leq n \leq l,$$ (4.11.6-19)

where the window size is calculated by:

$$l = T_{end} - T_{start} - 5,$$ (4.11.6-20)

and T_{adj} is calculated as:

$$T_{adj} = T_{start} - trunc\left\{\tau_{acc}\right\}.$$ (4.11.6-21)

The peak energy, E_{peak}, is then calculated as:

$$E_{peak} = \max\left\{E_{win}(n)\right\} \qquad ;0 \leq n \leq l,$$ (4.11.6-22)

and the average energy, $E_{avg}(n)$, is calculated as:

$$E_{avg}(n) = \begin{cases} \alpha E_{avg}(n-1) + (1-\alpha)\varepsilon_s^2\left(T_{adj} + n+4\right); & \varepsilon_s^2\left(T_{adj} + n+4\right) < 4E_{avg}(n-1), \\ E_{avg}(n-1) & ;\text{otherwise} \end{cases} \qquad 0 < n \leq l,$$ (4.11.6-23)

where $\alpha=0.875$, and $E_{avg}(0) = E_{win}(0)$.

The ratio of the peak to average energy is then calculated by:

$$E_{ratio} = \begin{cases} 0 & ;E_{avg}(l) = 0, \\ \left(\dfrac{E_{peak}}{E_{avg}(l)}\right)\left(\dfrac{54}{T_{end} - T_{start}}\right) & ;\text{otherwise}, \end{cases}$$

(4.11.6-24)

The shift decision flag, ε_{shift}, is then re-defined depending on the peak-to-average ratio by:

$$\varepsilon_{shift} = \begin{cases} FALSE & ;\left(E_{ratio} < 16.0\right), \\ TRUE & ;\text{otherwise}. \end{cases}$$

(4.11.6-25)

4.11.6.3 Matching the Residual to the Delay Contour

If the shift decision flag is true ($\varepsilon_{shift} == TRUE$) for the current shift frame, update the accumulated shift, τ_{acc}. The accumulated shift is adjusted by the shift required to match the residual shift frame determined in 4.5.6.2 to the modified residual target, $\hat{\varepsilon}_t(n)$. This operation is detailed in 4.5.6.3.1 through 4.5.6.3.4.

4.11.6.3.1 Computation of the Shift Range

The size of the residual frame to be shifted, l, is calculated by:

$$l = T_{end} - T_{start},$$

(4.11.6-26)

The residual frame is shifted between T_{srl} and T_{srr}, calculated by:

$$T_{srl} = \begin{cases} S_r + 1 & ;\tau_{acc} < 0, \\ S_r & ;\text{otherwise}, \end{cases}$$

(4.11.6-27)

$$T_{srr} = \begin{cases} S_r + 1 & ;\tau_{acc} > 0, \\ S_r & ;\text{otherwise}, \end{cases}$$

(4.11.6-28)

where $S_r = 3$.

For non-periodic signals, T_{srl} and T_{srr} are limited as defined by:

if [($\beta < 0.2$) and ($|\tau_{acc}| > 15$)] or [($\beta < 0.3$) and ($|\tau_{acc}| > 30$)]

    if ($\tau_{acc} < 0$)

        $T_{srr} = 1$

    else

        $T_{srl} = 1$

For both periodic and non-periodic signals, limit the shift bounds, T_{srl} and T_{srr}, as defined by:

$$T_{srl} = \min\left\{72 - trunc\left\{\tau_{acc}\right\}, T_{srl}\right\},$$

(4.11.6-29)

$$T_{srr} = \min\left\{72 + trunc\left\{\tau_{acc}\right\}, T_{srr}\right\}.$$

(4.11.6-30)

4.11.6.3.2 Generation of a Temporary Modified Residual Signal for Matching

Generate a temporary modified residual, $\hat{\varepsilon}_{tmp}(n)$, by shifting the current residual shift frame, $\varepsilon_s(n)$, by $\tau_{acc} + T_{srl}$:

$$\hat{\varepsilon}_{tmp}(n) = \sum_{i=0}^{2f_l} \hat{\varepsilon}_s \left(i + T_{start} - T - f_l + n\right) I_\varepsilon \left(i + \left(2f_l + 1\right)T_l\right) \quad ;0 \le n \le \left(l + T_{srl} + T_{srr}\right), \tag{4.11.6-31}$$

where T and T_I are defined by:

$$T_E(n) = \begin{cases} round\left\{\tau_{acc} + T_{srl}\right\} & ;\tau_{acc} + T_{srl} > 0, \\ -round\left\{-(\tau_{acc} + T_{srl})\right\} & ;\tau_{acc} + T_{srl} \le 0, \end{cases}$$

$$T_I = trunc\left\{\left(T - \left(\tau_{acc} + T_{srl}\right) + 0.5\right)R + 0.5\right\},$$

if ($T_I = R$) {

       $T_I = 0$

       $T = T - 1$

    }

and l is the shift frame size, $f_l = 3$, $R = 8$, and $\{I_\varepsilon(k)\}$ is a set of interpolation coefficients found in Table 8-10 and Table 8-11.

4.11.6.3.3 Matching the Temporary Modified Residual to the Target Residual

Generate an integer energy correlation vector, $E_I(n)$:

$$E_I = \sum_{i=0}^{l-1} \hat{\varepsilon}_{tmp}(n+i)\hat{\varepsilon}_t(T_{start} + i) \quad ;0 \le n \le T_{srl} + T_{srr}, \tag{4.11.6-32}$$

where $\left\{\hat{\varepsilon}_t\right\}$ was obtained in 4.5.6.1.

Interpolate $E_I(n)$ to obtain the fractional energy correlation vector, $E_f(n)$:

$$E_f\left(8(k-1)+j\right) = \sum_{i=-1}^{1} I_f(i,j)E_I(i+K) \quad ;0 \le j \le 7, 0 < T_{srl} + T_{srr}, \tag{4.11.6-33}$$

where $\left\{I_f(i,j)\right\}$ is a set of interpolation coefficients found in Table 8-17.

The optimal shift, τ_{opt}, that will match the temporary modified residual to the target residual is then defined as the index, n, that maximizes $E_f(n)$.

4.11.6.3.4 Adjustment of the Accumulated Shift

The accumulated shift is then adjusted by:

$$\tau_{acc} = \begin{cases} \tau_{acc} - \left[\dfrac{\tau_{acc} - RT_{srl} + R/2}{R}\right] & ;\alpha > 0.7, \\ \\ \tau_{acc} & ;\text{otherwise}, \end{cases} \tag{4.11.6-34}$$

where R = 8, and the gain, α, of the new shift, τ_{opt}, is calculated by:

$$\alpha = \begin{cases} 0 & ;E_\varepsilon E_T = 0, \\ \dfrac{E_f(\tau_{opt})}{\sqrt{E_\varepsilon E_T}} & ;\text{otherwise}. \end{cases} \tag{4.11.6-35}$$

where the energy of the temporary modified residual is calculated by:

$$E_\varepsilon = \sum_{i=T_{srl}}^{l+T_{srl}-1} \hat{\varepsilon}_{tmp}^2 (i + T_{start}),\tag{4.11.6-36}$$

and the energy of the target modified residual is defined by:

$$E_T = \sum_{i=0}^{l-1} \hat{\varepsilon}_t^2 (i + T_{start}).\tag{4.11.6-37}$$

4.11.6.4 Modification of the Residual

The current subframe residual, $\varepsilon_s(n)$, is shifted by τ_{acc}, to create the modified residual, $\hat{\varepsilon}(n)$, for the fixed codebook search:

$$\hat{\varepsilon}(n + n_m) = \sum_{i=0}^{2f_l} \varepsilon_s (i + n + n_m - T - f_l)I_E (i + T_I (2f_l + 1)) \quad ; 0 \le n \le T_{end} - T_{start},\tag{4.11.6-38}$$

where T and T_I are calculated by:

$$T = \begin{cases} round\{\tau_{acc}\} & ;\tau_{acc} > 0, \\ -round\{-\tau_{acc}\} & ;\tau_{acc} \le 0, \end{cases}$$

$$T_I = trunc\{(T - \tau_{acc} + 0.5)R + 0.5\},$$

if ($T_I = R$) {

$\quad\quad T_I = 0$

$\quad\quad T = T - 1$

$\quad$}

and $f_l = 8$, $R = 8$, and $\{I_E(n)\}$ is a set of interpolation coefficients found in Table 8-12 through Table 8-14.

Next, use the following pseudo-code to update n_m and determine whether the pulse searching procedure is complete:

$\quad$ if ($T_{end} < L$) {

$\quad\quad n_m = T_{end}$

$\quad\quad$ Go to 4.5.6.2

$\quad$ } else {

$\quad\quad n_m = T_{end} - L$

$\quad\quad$ Go to 4.5.6.5

$\quad$ }

4.11.6.5 Modified Target Residual Update

After having completed the modification of the residual, $\hat{\varepsilon}(n)$, for the entire subframe, the modified target residual buffer, $\hat{\varepsilon}_t(n)$, is updated as follows:

$$\hat{\varepsilon}_t(n) = \begin{cases} \hat{\varepsilon}_t(n + L) & ; -128 \le n < -L, \\ \hat{\varepsilon}(n + L) & ; -L \le n < 0 \end{cases},\tag{4.11.6-39}$$

1 ## 4.11.7 Computation of the ACELP Fixed Codebook Contribution for SO 3

2 The fixed codebook is based on an algebraic codebook structure, which has advantages in terms of storage, search
3 complexity, and robustness. The codebook structure is based on an interleaved single-pulse permutation (ISPP) design.
4 The codebook is searched on a subframe basis for the best index and gain to minimize the mean-squared weighted error
5 between the original and synthesis speech.

6 **Inputs**: The inputs to the algebraic codebook coding routine are:

7 - The length L impulse response of the weighted synthesis filter, $h_{wq}(k)$, zero extended to length 55

8 - The length L residual domain target vector, $x(k)$, zero extended to length 55

9 - The length L perceptual domain target vector, $x_w(k)$, zero extended to length 55

10 - The average pitch delay for the current subframe, τ

11 - The quantized pitch prediction gain, g_p

12 **Outputs**: The outputs of the algebraic codebook coding routine are:

13 - The codeword of the algebraic codebook, $FCBSIDX(m',i)$; $0 \le i \le 3$

14 - The fixed codebook excitation vector, $\mathbf{c}_k$

15 - The fixed codebook gain, g_c

16 **Processing**: The algebraic codebook shall be implemented using the procedures described in 4.11.7.1 through 4.11.7.4.

17 ## 4.11.7.1 Algebraic Codebook Structure, Rate 1

18 The Rate 1 fixed codebook is a 35-bit algebraic codebook. In this codebook, every codebook vector of length 55
19 contains at most 8 non-zero pulses. All pulses can have the amplitudes +1 or -1. The 55 positions in a subframe are
20 divided into 5 tracks, as shown in Table 4.11.7.1-1. The algebraic codebook search always examines all the 55 positions
21 as shown, regardless of subframe size. Based on the possible subframe size of 53, 53, and 54, the extra positions are
22 ignored.

23

24 **Table 4.11.7.1-1 Positions of Individual Pulses in the Rate 1 Algebraic Codebook**

Track	Positions
T0	0, 5, 10, 15, 20, 25, 30, 35,40,45,50
T1	1, 6, 11, 16, 21, 26, 31, 36,41,46,51
T2	2, 7, 12, 17, 22, 27, 32, 37,42,47,52
T3	3, 8, 13, 18, 23, 28, 33, 38,43,48,53
T4	4, 9, 14, 19, 24, 29, 34, 39,44,49,54

25

26 Of the 5 tracks, 3 are allocated two pulses each and 2 are allocated one pulse each. This accounts for a total of 8 pulses.
27 The single-pulse tracks can be either T3-T4, T4-T0, T0-T1, or T1-T2. The choice of single-pulse tracks is encoded with
28 2 bits. The positions of the pulses in the 2 single-pulse tracks are encoded with 7 bits (11x11=121<128) and their signs
29 are encoded with 2 bits. For each double-pulse track, both positions and signs of the two pulses are encoded with 8 bits,
30 which will be explained in more detail in 4.11.7.3. This gives a total of 35 bits (2+7+2+8x3).

1 The codebook vector, $\mathbf{c}_k$, is constructed according to:

2
$$c_k(j) = \sum_{i=0}^{N_p-1} s_i \delta(j - p_i) \quad ; 0 \le j \le 54 , \tag{4.11.7-1}$$

3 where $\delta(j - p_i)$ is a unit pulse at the i-th pulse position p_i of the k-th codevector, s_i is the sign of the i-th pulse, and N_p is

4 the number of pulses, and k is the range of all possible code vectors.

5 A special feature incorporated in the algebraic codebook is that the selected codebook vector is dynamically shaped by

6 filtering it through an adaptive pre-filter. In this implementation, the prefilter:

7
$$F(z) = \begin{cases} 1 & ; \tau \ge L, \\ \dfrac{1}{1 - g_p z^{-\tau}} & ; \text{otherwise,} \end{cases} \tag{4.11.7-2}$$

8 shall be used, where τ is the average subframe pitch delay and g_p is the pitch gain. The pitch gain is the quantized

9 pitch gain in the current subframe bounded by [0.2, 0.9]. For delays less than 55, the codebook vector, $\mathbf{c}_k$, is modified

10 according to:

11
$$c_k(j) = \begin{cases} c_k(j) & ; 0 \le j < \tau, \\ c_k(j) + g_p c_k(j - \tau) & ; \tau \le j \le 54. \end{cases} \tag{4.11.7-3}$$

12 This modification is incorporated in the fixed codebook search by including the prefilter in the impulse response

13 $h_{wq}(j)$. That is, prior to codebook search, the impulse response $h_{wq}(j)$ shall be modified according to:

14
$$h_{wq}(j) = \begin{cases} h_{wq}(j) & ; 0 \le j < \tau, \\ h_{wq}(j) + g_p h_{wq}(j - \tau) & ; \tau \le j \le 54. \end{cases} \tag{4.11.7-4}$$

15 ## 4.11.7.2 Algebraic Codebook Search

16 The algebraic codebook is searched by minimizing the mean-squared error between the weighted input speech and the

17 weighted synthesis speech. The perceptual domain target signal $x_w(n)$ is used in the closed-loop fixed-codebook search

18 and is given by Equation 4.11.4-19.

19 Let $\mathbf{c}_k$ be the algebraic codebook vector at index k. The algebraic codebook is searched by maximizing the term:

20
$$T_k = \frac{C_k}{E_k} = \frac{\left(\mathbf{d}^t \mathbf{c}_k\right)^2}{\mathbf{c}_k^t \mathbf{\Phi} \mathbf{c}_k} , \tag{4.11.7-5}$$

21 where $\mathbf{d} = \mathbf{H}^t \mathbf{x}_w$ is the cross-correlation between the perceptual domain target signal $x_w(n)$ and the impulse response

22 $h_{wq}(n)$, $\Phi = \mathbf{H}^t \mathbf{H}$ is the correlation matrix of the impulse response $h_{wq}(n)$, and $\mathbf{H}$ is a lower triangular Toeplitz

23 matrix with diagonal $h_{wq}(0)$ and lower diagonals $h_{wq}(1),..., h_{wq}(54)$, i.e.:

$$\mathbf{H} = \begin{bmatrix} h_{wq}(0) & 0 & 0 & \cdots & 0 \\ h_{wq}(1) & h_{wq}(0) & 0 & \cdots & 0 \\ h_{wq}(2) & h_{wq}(1) & h_{wq}(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{wq}(54) & h_{wq}(53) & h_{wq}(52) & \cdots & h_{wq}(0) \end{bmatrix}.$$

(4.11.7-6)

The cross-correlation vector **d** and the matrix Φ are computed prior to the codebook search. The elements of the vector **d** are computed by:

$$d(n) = \sum_{j=n}^{54} x_w(j) h_{wq}(j-n) \quad ; 0 \le n \le 54,$$

(4.11.7-7)

and the (i, j)-th element of the symmetric matrix Φ is computed by:

$$\phi(i, j) = \begin{cases} \sum_{n=\max\{i,j\}}^{L-1} h_{wq}(n-i) h_{wq}(n-j) & ; (0 \le j < L) \, and \, (0 \le i < L) \\ 0 & ; (L \le j < 55) \, and \, (L \le i < 55) \end{cases}.$$

(4.11.7-8)

The algebraic structure of the codebook allows for very fast search procedures since the innovation vector, $\mathbf{c}_k$, contains only a few non-zero pulses. The correlation in the numerator of Equation 4.11.7-5 is given by:

$$c_k = \left(\sum_{i=0}^{N_p-1} s_i d(p_i) \right)^2 .$$

(4.11.7-9)

The energy in the denominator of Equation 4.9.7-5 is given by:

$$E_k = \sum_{i=0}^{N_p-1} \phi(p_i, p_i) + 2 \sum_{i=0}^{N_p-2} \sum_{j=i+1}^{N_p-1} s_i s_j \phi(p_i, p_j) .$$

(4.11.7-10)

In order to determine the optimal algebraic codebook vector which maximizes the term in Equation 4.11.7-5, the correlation and energy terms in Equation 4.11.7-9 and Equation 4.11.7-10 should be computed for all possible combinations of pulse positions and signs. This, however, is a prohibitive task. In order to simplify the search, two strategies for searching the pulse signs and positions as explained below shall be used.

4.11.7.2.1 Pre-Setting of Pulse Signs

In order to simplify the search procedure, the pulse signs are preset (outside the closed loop search) by considering the sign of an appropriate reference signal. In this case, the signal $e(i)$, given by:

$$e_i = \sqrt{\frac{\sum_{j=0}^{54} d^2(j)}{\sum_{j=0}^{54} x^2(j)}} x(i) + 2d(i) ; \quad 0 \le i \le 54 ,$$

(4.11.7-11)

shall be used, where $x(i)$ is the residual domain target vector which is described in 4.11.4.10.

Amplitude pre-setting shall be done by setting the amplitude of a pulse at position i equal to the sign of $e(i)$. Hence, once the sign signal $s_i = \text{sign}\{e(i)\}$ and the signal $d'(i) = d(i)s_i$ are computed, then the matrix Φ shall be modified by including the sign information, that is, $\phi'(i, j) = s_i s_j \phi(i, j)$. The correlation in Equation 4.11.7-9 is now given by:

1

$$c_k = \left(\sum_{i=0}^{N_p-1} d'(p_i) \right)^2 , \qquad (4.11.7\text{-}12)$$

2   and the energy in Equation 4.5.7.2-6 is given by:

3

$$E_k = \sum_{i=0}^{N_p-1} \phi'(p_i, p_i) + 2 \sum_{i=0}^{N_p-2} \sum_{j=i+1}^{N_p-1} \phi'(p_i, p_j) . \qquad (4.11.7\text{-}13)$$

4   ### 4.11.7.2.2 Non-Exhaustive Pulse Position Search

5   Having preset the pulse amplitudes, as explained in 4.11.7.2.1, the optimal pulse positions shall be determined using an
6   efficient non-exhaustive analysis-by-synthesis search technique. In this technique, the term in Equation 4.11.7-5 is
7   tested for a small percentage of position combinations, using an iterative "depth-first" tree search strategy. In this
8   approach, the 8 pulses are grouped into 4 pairs of pulses. The pulse positions shall be determined sequentially one pair
9   at a time. In the first iteration, the single-pulse tracks are T3 and T4. The search process shall be repeated for other 3
10   iterations, by assigning the single-pulse tracks to T4-T0, T0-T1, and T1-T2 respectively. The codeword, q, used to
11   represent the various chosen tracks are given in Table 4.11.7.2.2-1.

12   **Table 4.11.7.2.2-1 Codeword for the Track Orders**

Double-pulse track order (p_0, p_1), (p_2, p_3), (p_4, p_5)	Single-pulse track order (p_6, p_7)	Codeword (q)
T0-T1-T2	T3-T4	00
T1-T2-T3	T4-T0	01
T2-T3-T4	T0-T1	10
T3-T4-T0	T1-T2	11

13

14   Once the positions and signs of the excitation pulses are determined, the codebook vector, $\mathbf{c}_k$, shall be built as in
15   Equation 4.11.7-1 and shall be modified according to Equation 4.11.7-3.

16   ### 4.11.7.3 Codeword Computation of the Algebraic Codebook

17   Upon completion of algebraic codebook search, the positions and signs of the pulses are encoded into 35 bits. The
18   algebraic codebook routine outputs 4 indices: the first 3 indices (8 bits each) represent the positions and signs of each
19   double-pulse track and the 4th index (11 bits) represents the positions and signs of pulses in the two single-pulse tracks
20   (9 bits) along with the codeword for the two single-pulse tracks (2 bits). For the codeword computations, the following
21   variables are defined for each of the single and double pulse tracks:

22

$$\lambda_0 = \lfloor p_i / 5 \rfloor; \quad i \in \{0,2,4,6\} , \qquad (4.11.7\text{-}14)$$

23

$$\lambda_1 = \lfloor p_j / 5 \rfloor; \quad j \in \{1,3,5,7\} . \qquad (4.11.7\text{-}15)$$

24   These are the decimated pulse positions and $s_{\lambda 0}$ and $s_{\lambda 1}$ are the signs of the respective decimated positions, and $b_{\lambda 0}$ and
25    $b_{\lambda 1}$ are defined as:

26

$$b_{\lambda i} = \begin{cases} 0 & ; s_{\lambda i} > 0, \\ 1 & ; s_{\lambda i} < 0 \end{cases} \qquad 0 \le i \le 1. \qquad (4.11.7\text{-}16)$$

1 In the case of coding the 3 double-pulse tracks, each 8 bit codeword *FCBSIDX*(*m′*,*i*), is $\left(0 \leq i \leq 2\right)$ calculated according

2 to:

3
$$FCBSIDX(m',i) = \begin{cases} b_{\lambda k} \times 128 + \min\{\lambda_0, \lambda_1\} \times 11 + \max\{\lambda_0, \lambda_1\}; & \text{if } s_{\lambda 0} = s_{\lambda 1}, \\ b_{\lambda k} \times 128 + \max\{\lambda_0, \lambda_1\} \times 11 + \min\{\lambda_0, \lambda_1\}; & \text{if } s_{\lambda 0} = -s_{\lambda 1}, \end{cases}$$
(4.11.7-17)

4 where $b_{\lambda k}$ follows the sign of the pulse located at the larger decimated position, i.e., $\max\{\lambda_0, \lambda_1\}$. The three double-

5 pulse codewords *FCBSIDX*(*m′*,*i*), $0 \leq i \leq 2$, are packed according to the order specified in the Table 4.11.7.2.2-1, which

6 depends on the positions of the single-pulse tracks.

7 In the case of coding the two single-pulse tracks, the 11 bit codeword *FCBSIDX*(*m′*,3) is calculated according to:

8
$$FCBSIDX(m',3) = q \times 512 + b_{\lambda 0} \times 256 + b_{\lambda 1} \times 128 + \lambda_0 \times 11 + \lambda_1,$$
(4.11.7-18)

9 where $0 \leq q \leq 3$ is the single-pulse track codeword in Table 4.11.7.2.2-1.

10 4.11.7.4 Algebraic Codebook Structure, Rate 1/2

11 A 10 bit algebraic codebook is used for Rate 1/2 packets. The innovation vector contains 3 non-zero pulses. Each pulse

12 has 8 possible positions, which are coded by 3 bits. The pulse positions and corresponding codewords are given in Table

13 4.11.7.4-1. All pulses have a fixed signs (+1 for T_0 and T_2 and -1 for T_1). An additional bit, however, is used to change

14 the signs of all three pulses simultaneously, i.e., *s* = 1 indicates polarity inversion, s = 0 otherwise. Therefore, the total

15 bits per subframe for Rate 1/2 is 3*3+1=10.

16 The codebook is searched using techniques similar to that for Rate 1. However, in the Rate 1/2 case, all the pulse

17 position combinations are exhaustively searched by maximizing Equation 4.11.7-5. Upon completion of the search, the

18 sign information is processed as follows:

19   if (sign{ $\mathbf{d}^t\mathbf{c}_k$ } < 0) {

20          $\mathbf{c}_k = -\mathbf{c}_k$

21   *s* = 1

22   } else {

23   *s* = 0

24   }

25 By referencing Table 4.11.7.4-1, the 10-bit codeword can then be calculated by:

26
$$FCBSIDX(m',0) = s \times 512 + q_{T0} \times 64 + q_{T1} \times 8 + q_{T2},$$
(4.11.7-19)

Table 4.11.7.4-1 Positions of Individual Pulses in the Rate 1/2 Algebraic Codebook

Pulse	Positions							
T_0	0	7	14	21	28	35	42	49
T_1	2	9	16	23	30	37	44	51
T_2	4	11	18	25	32	39	46	53
Codewords q_{Ti} ; $0 \leq i \leq 2$	000	001	010	011	100	101	110	111

4.11.8 Computation of the Factorial Fixed Codebook Contribution for SO 68, SO 70, and SO 73

The fixed codebook is based on a factorial codebook structure [R13], which has advantages in terms of storage, search complexity, and robustness. The codebook is searched on a subframe basis for the best index and gain to minimize the mean-squared weighted error between the original and synthesis speech.

Inputs: The inputs to the factorial codebook coding routine are:

- The length L impulse response of the weighted synthesis filter, $h_{wq}(k)$, zero extended to length 55

- The length L residual domain target vector, $x(k)$, zero extended to length 55

- The length L perceptual domain target vector, $x_w(k)$, zero extended to length 55

- The average pitch delay for the current subframe, τ

- The quantized pitch prediction gain, g_p

- For SO 70 and SO 73 wideband mode, the codebook configuration parameter, C_p

- For SO 70 and SO 73 wideband mode, the filtered ACB excitation, $\lambda(n)$

- For SO 70 and SO 73 wideband mode, the unquantized ACB gain, g_{p_unq}

Outputs: The outputs of the factorial codebook coding routine are:

- The codeword of the factorial codebook, $FCBSIDX(m',i)$; $0 \leq i \leq 3$

- The fixed codebook excitation vector, $\mathbf{c}_k$

- The fixed codebook gain, g_c

Processing: The factorial codebook shall be implemented using the procedures described in Sections 4.11.8.1 through 4.11.8.6.

4.11.8.1 Fixed Codebook (FCB) Structure, Rate 1

For SO 68, SO 70, and SO 73 narrowband interoperability mode, the Rate 1 fixed codebook is a 35-bit sparse pulse codebook. In this codebook, every codevector of length L contains at most 7 nonzero pulse positions. All pulses can have the amplitudes of 1 with a sign of +1 or -1. The 7 pulses can take any position in the subframe, however, any pulses that occupy the same position shall maintain the same sign. Further, up to 7 of these pulses can be stacked onto the same

1   position. Stacking reduces the number of nonzero pulses, but gives greater flexibility in controlling the shape for the

2   fixed code vector.

3   For SO 70 and SO 73 wideband mode, the Rate 1 FCB code vector contains either 5 or 6 pulses, depending on the

4   codebook configuration parameter C_p. That is, if the default ACB is active ($C_p = 0$), then a 6 pulse fixed codebook is

5   used. When the delay contour adjustment ACB is active ($C_p = 1$), then a 5 pulse fixed codebook is used. The number of

6   pulses generated by the various factorial codebooks are shown in Table 4.11.8.1-1.

7   **Table 4.11.8.1-1 Number of Pulses Generated by Factorial Codebooks**

SO 68, S0 70, and SO 73 narrowband interoperability mode	SO 70 and SO 73 wideband mode	
	$C_p = 0$	$C_p = 1$
$N_p = 7$	$N_p = 6$	$N_p = 5$

8

9   The maximum number of code-vectors is given by

$$N = {}^{L}FPC_{N_p} = \sum_{p=1}^{N_p} {}^{L}C_p {}^{(N_p-1)}C_{(p-1)} 2^p \; , \tag{4.11.8-1}$$

11   where N_p is the number of unit amplitude pulses, L is the subframe size, and:

$$ {}^{n}C_r = \frac{n!}{r!(n-r)!} \tag{4.11.8-2}$$

13   is the number or ways to choose r of n items. It can be easily seen that for $N_p = 7$ and $L = 54$, then the number of bits

14   required to code the N unique codeword combinations is $\lceil \log_2(N) \rceil = 35$ bits. The codeword k spans the range $0 \le k < N$.

15   The positions of the pulses are searched as described in Section 4.11.4.11, where the FCB codevector, $\mathbf{c}_k$, is constructed

16   according to:

$$c_k(j) = \sum_{i=0}^{N_p-1} s_j \delta(j - p_i) \quad ; 0 \le j < L \; , \tag{4.11.8-3}$$

18   where $\delta(j - p_i)$ is a unit pulse at the i-th pulse position p_i, and s_j is the sign of the j-th pulse position within the

19   subframe length L.

20   A special feature incorporated in the factorial codebook is that the selected codebook vector is dynamically shaped by

21   filtering it through an adaptive pre-filter. In this implementation, the prefilter:

$$F(z) = \begin{cases} 1 & ; \dot{\tau} \ge L, \\ \dfrac{1}{1 - g_p z^{-\dot{\tau}}} & ; \text{otherwise,} \end{cases} \tag{4.11.8-4}$$

23   shall be used, where $\dot{\tau}$ is the average subframe pitch delay and g_p is the pitch gain. The pitch gain is the quantized

24   pitch gain in the current subframe bounded by [0.2, 0.9]. For delays less than the subframe length L, the codebook

25   vector, $\mathbf{c}_k$, is modified according to:

$$c_k(j) = \begin{cases} c_k(j) & ; 0 \le j < \dot{\tau} \\ c_k(j) + g_p c_k(j - \dot{\tau}) & ; \dot{\tau} \le j < L \end{cases} \tag{4.11.8-5}$$

1 This modification is incorporated in the fixed codebook search by including the prefilter in the impulse response $h_{wq}(j)$.

2 That is, prior to codebook search, the impulse response $h_{wq}(j)$ shall be modified according to:

3
$$h_{wq}(j) = \begin{cases} h_{wq}(j) & ; 0 \leq j < \dot{\tau} \\ h_{wq}(j) + g_p h_{wq}(j - \dot{\tau}) & ; \dot{\tau} \leq j < L \end{cases}$$
(4.11.8-6)

4.11.8.2 Codebook Search

5 The factorial codebook is searched by minimizing the mean-squared error between the weighted input speech and the
6 weighted synthesis speech. The perceptual domain target signal $x_w(n)$ is used in the closed-loop fixed-codebook search
7 and is given by Equation 4.9.4-15.

8 Let $\mathbf{c}_k$ be the factorial codebook vector at index k. The factorial codebook is searched by maximizing the term:

9
$$T_k = \frac{C_k}{E_k} = \frac{(\mathbf{d}^t \mathbf{c}_k)^2}{\mathbf{c}_k^t \mathbf{\Phi} \mathbf{c}_k},$$
(4.11.8-7)

10 where $\mathbf{d} = \mathbf{H}^t \mathbf{x_w}$ is the cross-correlation between the perceptual domain target signal $x_w(n)$ and the impulse response
11 $h_{wq}(n)$, $\mathbf{\Phi} = \mathbf{H}^t \mathbf{H}$ is the correlation matrix of the impulse response $h_{wq}(n)$, and $\mathbf{H}$ is a lower triangular Toeplitz matrix
12 with diagonal $h_{wq}(0)$ and lower diagonals $h_{wq}(1),..., h_{wq}(L-1)$, i.e.:

13
$$\mathbf{H} = \begin{bmatrix} h_{wq}(0) & 0 & 0 & \cdots & 0 \\ h_{wq}(1) & h_{wq}(0) & 0 & \cdots & 0 \\ h_{wq}(2) & h_{wq}(1) & h_{wq}(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{wq}(L-1) & h_{wq}(L-2) & h_{wq}(L-3) & \cdots & h_{wq}(0) \end{bmatrix}$$
(4.11.8-8)

14 The cross-correlation vector $\mathbf{d}$ and the matrix $\mathbf{\Phi}$ are computed prior to the codebook search. The elements of the vector $\mathbf{d}$
15 are computed by:

16
$$d(n) = \sum_{j=n}^{L-1} x_w(j) h_{wq}(j-n) \quad ; 0 \leq n < L,$$
(4.11.8-9)

17 and the (i, j)-th element of the symmetric matrix $\mathbf{\Phi}$ is computed by:

18
$$\phi(i, j) = \sum_{n=\max\{i,j\}}^{L-1} h_{wq}(n-i) h_{wq}(n-j) \quad ; 0 \leq i < L, \quad 0 \leq j < L.$$
(4.11.8-10)

19 For SO 70 and SO 73 wideband mode, a joint parameter optimization approach proposed in [R15] shall be used for the
20 factorial codebook search. This method jointly optimizes the adaptive codebook and the fixed codebook gains together
21 with the fixed codebook excitation. This is accomplished by modifying the correlation matrix $\mathbf{\Phi}$ based on the filtered
22 adaptive codebook excitation $\lambda(n)$, to produce a modified correlation matrix $\tilde{\mathbf{\Phi}}$, which is then used in place of the
23 original correlation matrix during the subsequent FCB search procedures, i.e., $\mathbf{\Phi} \leftarrow \tilde{\mathbf{\Phi}}$.

24 The first step in producing the modified correlation matrix requires the generation of the joint search weighting fact λ_{js}
25 given by:

1

$$\lambda_{js} = \begin{cases} 0.0 & ; \dot{\tau} < L \\ \dfrac{0.97(1.3 - g_{p\_unq})}{\sqrt{1.01 R_{\lambda\lambda}(0)}} & ; \dot{\tau} \geq L, g_{p\_unq} \geq 1.15 \\ \dfrac{0.97 g_{p\_unq}}{\sqrt{1.01 R_{\lambda\lambda}(0)}} & ; \dot{\tau} \geq L, g_{p\_unq} \leq 0.05 \\ \dfrac{0.97}{\sqrt{1.01 R_{\lambda\lambda}(0)}} & ; otherwise \end{cases}$$

(4.11.8-11)

2 where $R_{\lambda\lambda}(0)$ is the energy of the filtered ACB excitation $\lambda(n)$. Then, the backward filtered ACB vector **y** as:

3

$$y(n) = \lambda_{js} \sum_{j=n}^{L-1} \lambda(j) h_{wq}(j-n) \quad ; 0 \leq n < L,$$

(4.11.8-12)

4 Finally, the elements of the modified correlation matrix are compluted as:

5

$$\widetilde{\phi}(i, j) = \phi(i, j) - y(i)y(j) \quad ; 0 \leq i < L, \quad 0 \leq j < L.$$

(4.11.8-13)

6 which then replaces the original correlation matrix $\mathbf{\Phi} \leftarrow \widetilde{\mathbf{\Phi}}$.

7 The sparse structure of the codebook allows for very fast search procedures since the innovation vector, $\mathbf{c}_k$, contains only
8 a few nonzero pulses. The correlation in the numerator of Equation 4.11.8-7 is given by:

9

$$C_k = \left(\sum_{i=0}^{N_p - 1} s_i d(p_i) \right)^2.$$

(4.11.8-14)

10 The energy in the denominator of Equation 4.11.8-7 is given by:

$$E_k = \sum_{i=0}^{N_p - 1} \phi(p_i, p_i) + 2 \sum_{i=0}^{N_p - 2} \sum_{j=i+1}^{N_p - 1} s_i s_j \phi(p_i, p_j)$$

11
.

(4.11.8-15)

12 In order to determine the optimal factorial codebook vector which maximizes the term in Equation 4.11.8-7, the
13 correlation and energy terms in Equation 4.11.8-14 and Equation 4.11.8-15 should be computed for all possible
14 combinations of pulse positions and signs. This, however, is a prohibitive task. In order to simplify the search, two
15 strategies for searching the pulse signs and positions as explained below shall be used.

16 ## 4.11.8.2.1 Pre-Setting of Pulse Signs

17 In order to simplify the search procedure, the pulse signs are preset (outside the closed loop search) by considering the
18 sign of an appropriate reference signal. For SO 68, SO 70, and SO 73 narrowband interoperability mode, the signal $e(i)$,
19 given by:

20

$$e(i) = \sqrt{\frac{\sum_{j=0}^{L-1} d^2(j)}{\sum_{j=0}^{L-1} x^2(j)}} x(i) + 2d(i); \qquad 0 \leq i < L,$$

(4.11.8-16)

21 shall be used, where $x(i)$ is the residual domain target vector which is described in Section 4.11.4.10. For SO 70 and SO
22 73 wideband mode, the signal $e(i)$ is defined simply as $e(i) = d(i); 0 \leq i < L$.

1 Sign pre-setting shall be done by setting the sign of a pulse at position *i* equal to the sign of *e*(*i*). Hence, once the sign

2 signal $s_i = \text{sign}\{e(i)\}$ and the signal $d'(i) = s_i d(i)$ are computed, then the matrix $\mathbf{\Phi}$ shall be modified by including the sign

3 information, that is, $\phi'(i,j) = s_i s_j \phi(i,j)$. The correlation in Equation 4.11.8-14 is now given by:

$$c_k = \left(\sum_{i=0}^{N_p-1} d'(p_i) \right)^2,$$

4 (4.11.8-17)

5 and the energy in Equation 4.11.8-15 is given by:

$$E_k = \sum_{i=0}^{N_p-1} \phi'(p_i, p_i) + 2 \sum_{i=0}^{N_p-2} \sum_{j=i+1}^{N_p-1} \phi'(p_i, p_j).$$

6 (4.11.8-18)

7 4.11.8.2.2 Non-Exhaustive Pulse Position Search

8 Having preset the pulse amplitudes, as explained in Section 4.11.8.2.1, the optimal pulse positions shall be determined

9 using an efficient non-exhaustive analysis-by-synthesis search technique. Once the positions and signs of the excitation

10 pulses are determined, the codebook vector, $\mathbf{c}_k$, shall be built as in Equation 4.11.8-3 and shall be modified according to

11 Equation 4.11.8-5.

12 4.11.8.3 Codeword Computation of the Factorial Codebook

13 Upon completion of factorial codebook search, the positions and signs of the pulses for SO 68, SO 70, and SO 73

14 narrowband interoperability mode are encoded into 35 bits based on the chosen codebook excitation vector, $\mathbf{c}_k$.

15

16 **Table 4.11.8.3-1 Definitions for Full-Rate CELP Factorial Codebook**

Code	Definition
L	Subframe size
T	Total number of unit amplitude pulses
$\widetilde{T}$	Number of nonzero pulses (or locations) in the best code vector; $1 \le \widetilde{T} \le T$
$P_i;\quad 0 \le i \le \widetilde{T}-1$	Positions in the code vector that contain the $\widetilde{T}$ nonzero pulses; $0 \le P_i \le L-1$
$S_i;\quad 0 \le i \le \widetilde{T}-1$	Sign information of the $\widetilde{T}$ nonzero pulses; $S_i = \begin{cases} 0; & \text{if sign of pulse at } P_i \text{ is zero of positive} \\ 1; & \text{if sign of pulse at } P_i \text{ is negative} \end{cases}$
$A_i;\quad 0 \le i \le \widetilde{T}-1$	Number of unit amplitude pulses in each of the $\widetilde{T}$ nonzero pulse locations; $\sum_{i=0}^{\widetilde{T}-1} A_i = T$
$T_d^k;\quad 1 \le k \le T$	Number of pulse locations with at least k stacked unit amplitude pulses. Note: $T_d^1 = \widetilde{T}, T_d^{k-1} \ge T_d^k$
$T_m^k;$	Total number of unit amplitude pulse (T) - number of pulse locations with at most k stacked unit amplitude pulses.
$P_d^k;\quad 1 \le k \le T$	Set of positions in the code vector that contain at least k stacked unit amplitude pulses; $0 \le P_d^k \le L-1$

Code	Definition
$I_{off\_S}\left(\widetilde{T},T,L\right)$	Total number of ways of putting T pulses into greater than $\widetilde{T}$ nonzero locations. That is, $$I_{off\_S}\left(\widetilde{T},T,L\right)=\sum_{p=\widetilde{T}+1}^{T}{}^{54}C_p{}^{(7-1)}C_{(p-1)}2^p$$ This is the minimum offset in the code-vector space corresponding to a vector that has exactly $\widetilde{T}$ nonzero pulses
$I_{off}\left(T_d^k,T_m^k,T_d^{k-1}\right)$	Total number of ways of putting T_m^k pulses into greater than T_d^k nonzero locations out of maximum possible T_d^{k-1} locations. That is, $$I_{off}\left(T_d^k,T_m^k,T_d^{k-1}\right)=\sum_{p=T_d^k+1}^{T_m^k}{}^{T_k^{k-1}}C_p{}^{\left(T_m^k-1\right)}C_{(p-1)}$$
$I_{pos}\left(P_d^k,T_d^k\right)$	Sum of the number of ways of putting p pulses in $P_d^k(p)$, for all p such that $$\sum_{p=0}^{T_d^k}\left(P_d^k(p)\right)C_p$$

1

2 The code-word is given by

3
$$C=\sum_{i=0}^{\widetilde{T}-1}\left(S_i 2^{\widetilde{T}-i}\right)+I_{off\_S}\left(\widetilde{T},T,L\right)+\left(\sum_{k=1}^{T}\left(I_{off}\left(T_d^k,T_m^k,T_d^{k-1}\right)+I_{pos}\left(P_d^k,T_d^k\right)D\left(T_m^k,T_d^k\right)\right)\right)2^{\widetilde{T}}$$
(4.11.8-19)

4 Basically, given the number of locations of nonzero pulses, their signs, and the number of stacked pulses at each of the
5 nonzero locations, the code-word in the above formula is built progressively using offsets.

6 For SO 70 and SO 73 wideband mode, the codebook configuration parameter C_p, closed loop delay adjustment indices
7 *DELAY_ADJ_INX(m′)*, and factorial codebook excitation vector $\mathbf{c}_k$ are coded for all 3 subframes using a single 94 bit
8 codeword. Details of how the bits are dynamically allocated withing the codeword are given in Table 4.11.8.3-2. From
9 this table, it can be seen that the codebook configuration parameter is coded for the 3 subframes using a variable length
10 Huffman code, and that the Huffman code indicates which subframes contain the default ACB/6 pulse FCB and
11 conversely, which subframes contain the delay adjust ACB/5 pulse FCB.

Table 4.11.8.3-2 Dynamic Bit Allocation of Excitation Indices

Codebook configuration C_p (number of pulses N_p) per subframe m'			Huffman Code	Huffman bits	Delay adjust bits per subframe m'			FCB bits per frame	Total bits per frame
0	**1**	**2**			**0**	**1**	**2**		
0 (6)	0 (6)	0 (6)	0	1	0	0	0	93	94
1 (5)	0 (6)	0 (6)	100	3	2	0	0	89	94
0 (6)	1 (5)	0 (6)	101	3	0	2	0	89	94
0 (6)	0 (6)	1 (5)	110	3	0	0	2	89	94
0 (6)	1 (5)	1 (5)	11100	5	0	2	2	85	94
1 (5)	0 (6)	1 (5)	11101	5	2	0	2	85	94
1 (5)	1 (5)	0 (6)	11110	5	2	2	0	85	94
1 (5)	1 (5)	1 (5)	11111	5	2	2	2	81	92

In order to efficiently code the factorial codebook indices, the codewords for each of the 3 subframes shall be combined together to form a single large codeword. The number of bits for the codeword is determined by the relation:

$$Total\ FCB\ bits\,/\,frame = \left\lceil \log_2\left(\prod_{i=0}^{2} {}^{L_i}FPC_{N_i} \right) \right\rceil \tag{4.11.8-20}$$

where ${}^{L_i}FPC_{N_i}$ is the number of combinations for the i-th subframe, as described in Equation 4.11.8-1, and where L_i and N_i are the respective subframe lengths and number of pulses for the i-th subframe. The total number of FCB bits can range from 81 to 93 bits. The FCB codeword is constructed such that the factorial codeword $k_{m'}$ for each subframe is split into a most significant portion $G_{m'}$ and a least significant portion $H_{m'}$, such that $k_{m'} = [G_{m'}\ H_{m'}]$, and then combined in a manner that does not require extended precision arithmetic operations. This is accomplished by using traditional combinatiorial coding only on the most significant parts of the 3 subframe codewords $G_{m'}$, given in general terms as:

$$\mathbf{\Gamma} = \sum_{j=0}^{2} \left(\prod_{i=0}^{j} G_{i-1}^{\max} \right) G_j, \tag{4.11.8-21}$$

where $G_i^{\max}$ is the number of possible combinations of G_i, and $G_{-1}^{\max}=1$. The composite FCB codeword K is then formed by mutiplexing (i.e., concatenating) the combined portion of the most significant parts $\mathbf{\Gamma}$ with the least significant portions of the individual codewords $H_{m'}$, expressed as:

$$K = \begin{bmatrix} \mathbf{\Gamma} & H_0 & H_1 & H_2 \end{bmatrix}. \tag{4.11.8-22}$$

4.11.8.4 Converting a Code-Word to a Code-Vector

Given a code-word, C , the code-vector , $\mathbf{c}_k$, is reconstructed using the following iterative procedure, where the notation $D(x,y) = {}^{(x-1)}C_{(y-1)}$.

```
        Tlen = L;

        for (indx=0; indx<Tlen; indx++) {
```

```
1        retvector[indx]=0;
2        signvector[indx]=1;
3      }
4
5    Nd=0;
6    do {
7      Tcode = Ioff_S(Nd, Np, Tlen);
8      Nd++;
9    } while(Tcode>code);
10   Nd--;
11   Spos = code & ((0x0001<<Nd) - 1);
12
13   Tcode = (code-Ioff_S(Nd, Np, Tlen))>>Nd;
14   pos = Tcode/ D(Np,Nd);
15   for(indx=0; indx<Nd; indx++) {
16     TNd=Nd-indx-1;
17     while(F(TNd, Nd-indx)<=pos) {
18       TNd++;
19     }
20     TNd--;
21     posvect[Nd-indx-1]=TNd;
22     retvector[TNd]++;
23     signvector[TNd]=(Spos & (0x0001<<(indx)))?-1:1;
24     pos=pos-F(TNd, Nd-indx);
25   }
26   Tcode = (code-Ioff_S(Nd, Np, Tlen))>>Nd;
27   pos = Tcode/D(Np,Nd);
28   code = Tcode - pos*D(Np,Nd);
29   Np = Np - Nd;
30   Tlen = Nd;
31   pv_offset=Tlen;
32   while(Np>0) {
33     Nd=0;
34     while(Ioff(Nd, Np, Tlen)>code) {
35       Nd++;
36     }
37
38     Tcode = (code-Ioff(Nd, Np, Tlen));
39     pos = Tcode/D(Np,Nd);
40     for(indx=0; indx<Nd; indx++) {
41       TNd=Nd-indx-1;
42       while(F(TNd, Nd-indx) <= pos) {
43         TNd++;
44       }
```

```
1                TNd--;
2                retvector[posvect[TNd+pv_offset-Tlen]]++;
3                posvect[pv_offset-indx-1]=posvect[TNd+pv_offset-Tlen];
4                pos=pos-F(TNd, Nd-indx);
5             }
6          Tcode = (code-Ioff(Nd, Np, Tlen));
7          pos = Tcode/D(Np,Nd);
8          code = Tcode - pos*D(Np,Nd);
9          Np = Np - Nd;
10         Tlen = Nd;
11      }
12    for(indx=0; indx<len; indx++) {
13       c_k [indx]=retvector[indx]*signvector[indx];
14    }
15
```

4.11.8.5 Codebook Structure, Rate 1/2

A 10 bit sparse codebook is used for Rate 1/2 packets. The innovation vector contains 3 nonzero pulses. The 3-pulse codebook takes different structures, i.e., allows different pulse positions, depending on the nature of the input signal, pitch lag, periodicity, etc. The bit-usage is reduced by allowing only certain combinations of pulse-positions for the 3-pulses, based on the nature of the input signal.

The codebook is searched using techniques similar to that for Rate 1.

Case 1 ($lag \geq 95$ or ($lag > 66$, and adaptive codebook gain, $g_p > 0.3$))

$\underline{lag \geq 95}$

$sign = 1$

$ratio = min(max(0.9 * (lag - 95), 0.1), 0.8)$

$offset = \begin{cases} 1; & lag < 100 \\ 2; & otherwise \end{cases}$

$lag > 66$, and adaptive codebook gain, $g_p > 0.3$

$sign = -1$

$ratio = 0.0$

$offset = 0$

1

Table 4.11.8.5-1 Pulse Positions for Half-Rate CELP Codebook

Pulse	Positions
Possible positions for pulse 0, p_0	1, 3, 6, 8, 10, 13, 15, 18, 20, 22, 25, 27, 30, 32, 34, 37, 39, 42, 44, 46, 49, 51
Possible positions for pulse 1, p_1	$p_0 + offset$
Possible positions for pulse 2, p_2	0, 2, 4, 7, 9, 12, 14, 16, 19, 21, 24, 26, 28, 31, 33, 36, 38, 40, 43, 45, 48, 50, 52

2

3 Numerator correlation is

4
$$c_k = d(p_0) + r \cdot d(p_r) + sign \cdot d(p_1) \tag{4.11.8-23}$$

5
$$E_k = \phi(p_0, p_0) + r \cdot (r \cdot \phi(p_1) + 2 \cdot \phi(p_0, p_1)) + \phi(p_2, p_2) + 2.sign.(\phi(p_0, p_2) + r \cdot \phi(p_2, p_1)) \tag{4.11.8-24}$$

6 Once the best set of p_0, p_1, and p_2 are chosen, the best code vector is given by

7
$$cb(n) = \begin{cases} s(p_0); & n = p_0 \\ r \cdot s(p_1); & n = p_1 \\ s(p_2); & n = p_2 \end{cases} \tag{4.11.8-25}$$

8 Case 0 ($lag \le 66$, or ($lag < 95, and$ adaptive codebook gain, $g_p \le 0.3$))

9
$$span = \begin{cases} 24; & g_p > 0.3 \& (lag < SFSIZE) \& (lag < 24) \\ lag; & g_p > 0.3 \& (lag < SFSIZE) \& (lag \ge 24) \\ SFIZE & otherwise \end{cases} \tag{4.11.8-26}$$

10 *span* is divided into three parts t_0 , t_1 , t_2 such that

11
$$t_0 = \begin{cases} \lfloor span/3 \rfloor & span\%3 = 0 \\ \lfloor span/3 \rfloor + 1 & otherwise \end{cases} \tag{4.11.8-27}$$

12
$$t_1 = \begin{cases} \lfloor span/3 \rfloor + 1 & span\%3 = 2 \\ \lfloor span/3 \rfloor & otherwise \end{cases} \tag{4.11.8-28}$$

13
$$t_2 = \lfloor span/3 \rfloor \tag{4.11.8-29}$$

14
$$skew0 = skew\_table0[54 - span] \tag{4.11.8-30}$$

15
$$skew1 = skew\_table1[54 - span] \tag{4.11.8-31}$$

16 where

17

Table 4.11.8.5-2 Skew Table for Half-Rate CELP Codebook

Skew table	Positions
skew_table0	0, -2, -3, -1, 1, 0, -2, -3, -1, -1, 2, 0, 2, 0, 3, -3, 0, 3, -3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
skew_table1	1, -1, -2, -3, 0, 0, 3, -2, 0, -1, -3, -3, 1, 2, 0, -3, 1, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

18

Table 4.11.8.5-3 Pulse Positions for Half-Rate CELP Codebook

Pulse position	Definition
Possible positions for p_0	$3 \cdot \left(\left\lfloor \dfrac{i \cdot (t_0 t_1 t_2)/512 + (t_0 t_1 t_2)/512/2}{t_1 t_2} \right\rfloor \%t_0 \right)$
Possible positions for p_1	$3 \cdot \left(\left\lfloor \dfrac{i \cdot (t_0 t_1 t_2)/512 + (t_0 t_1 t_2)/512/2}{t_2} \right\rfloor \%t_1 \right) + 1$
Possible positions for p_2	$3 \cdot \left(\left\lfloor i \cdot (t_0 t_1 t_2)/512 + (t_0 t_1 t_2)/512/2 + l_0 skew_0 + l_1 skew_1 \right\rfloor \%t_2 \right) + 2$

If lag < 54, and gp>0.3, the impulse response is pitch sharpened as according to:

$$h_{wq}(j) = \begin{cases} h_{wq}(j), & 0 \le j \le \tau \\ h_{wq}(j) + h_{wq}(j - \tau), & \tau \le j \le 54 \end{cases}$$

4.11.8.5.1 Converting a Rate 1/2 Code-Word to a Code-Vector

The following pseudo-code illustrates how to convert the code-vector to a code-word:

```
            for (i=0; i<55; i++) cod[i] = 0.0;

            if((*gain_code = Exy / Eyy) < 0) {
                    *gain_code = -(*gain_code);
                    s = -1.0;
                    index = 1;
            } else {
                    s = 1.0;
                    index = 0;
            }

            if (flag > 0) {
                    for(i=0; i<3; i++) {
                            pos = pos_table[ix][i];
                            cod[pos] += s;
                            s = -s;
                    }
                    index <<= 9;
                    index += ix;
            } else {
                    pos = codvec[0];
                    cod[pos] += s;
                    cod[pos+offset[1]] += s * r;
                    i0 = (pos - pos/6)/2;

                    pos = codvec[1];
```

```
cod[pos] += s*sign;
i1 = (pos - pos/6)/2;


index = (index <<9) + i0*23 + i1;
}


*indices = index;


if(flag == 2)
        for (i = pit_lag; i < l_subfr; i++) cod[i] += cod[i -
pit_lag];
```

If lag < 54, and gp>0.3, the impulse response is pitch sharpened as according to:

$$c_k(j) = \begin{cases} c_k(j) & ; 0 \le j < \tau, \\ c_k(j) + g_p c_k(j - \tau) & ; \tau \le j \le 54. \end{cases}$$

4.11.8.6 Fixed Codebook Gain Calculation

The fixed codebook gain for both Rate 1 and Rate 1/2 shall be found by:

$$g_c = \frac{\mathbf{d}^t \mathbf{c}_k}{\mathbf{c}_k^t \Phi \mathbf{c}_k}, \tag{4.11.8-32}$$

where all variables are previously defined.

4.12 Voiced Encoding (Full-Rate PPP and Quarter-Rate PPP)

Stationary voiced frames are coded using the Prototype Pitch Period (PPP) synthesis model. In the PPP synthesis model, only one prototype pitch period waveform is extracted from approximately the end of a frame of the modified residual is quantized and sent to the decoder. The whole frame of the excitation is reconstructed at the decoder by pitch synchronous interpolation between the quantized prototype pitch period of the current frame, and the prototype pitch period of the previous frame.

4.12.1 Modification of the Original Residual

Generate a modified residual signal, $r(n)$, by shifting the current residual signal, $\varepsilon(n)$, to match the delay contour for the current subframe. See Section 4.11.4.4 for details on this operation.

4.12.2 Prototype Pitch Period Processing

The Discrete Fourier Series (DFS) representation of the prototype pitch period is used in various aspects of the voiced encoder. The DFS representation for a signal $x(n)$ of length L is defined as

$$X(k) = \frac{1}{L} \sum_{n=0}^{L-1} x(n) e^{j\frac{2\pi}{L}kn}; \qquad 0 \le k < L$$

$$\tag{4.12.2-1}$$

As X is a complex vector, it is equivalently maintained in the speech coder as two vectors − $X.a$, the real vector, and $X.b$, the imaginary vector. Further, note that x is a real vector in the time domain. Therefore, the DFS components $X.a$, and $X.a$ are symmetric about the frequency π . Therefore, $X.a$, and $X.a$ are defined by

$$X.a(k) = \begin{cases} \dfrac{2}{L}\sum_{n=0}^{L-1} x(n)\cos\left(\dfrac{2\pi}{L}kn\right); & 1 \le k < \left\lfloor \dfrac{L-1}{2} \right\rfloor \\ 0; & k = 0 \\ \dfrac{1}{L}\sum_{n=0}^{L-1} x(n)\cos\left(\dfrac{2\pi}{L}\dfrac{L}{2}n\right); & k = \dfrac{L}{2}; \textit{if } L \text{ is even} \end{cases}$$

(4.12.2-2)

$$X.b(k) = \begin{cases} \dfrac{2}{L}\sum_{n=0}^{L-1} x(n)\sin\left(\dfrac{2\pi}{L}kn\right); & 1 \le k < \left\lfloor \dfrac{L-1}{2} \right\rfloor \\ 0; & k = 0 \\ 0; & k = \dfrac{L}{2}; \textit{if } L \text{ is even} \end{cases}$$

(4.12.2-3)

The inverse DFS of X is defined as

$$x(n) = X.a(0) + \sum_{k=1}^{\left\lfloor \frac{L}{2} \right\rfloor} X.a(k)\cos\left(\dfrac{2\pi}{L}nk\right) + X.b(k)\sin\left(\dfrac{2\pi}{L}nk\right); \qquad 0 \le n < L$$

(4.12.2-4)

For simplicity of notation, we will henceforth simply use the unsubscripted notations like lower-case x , and the upper-case X to refer to the time domain vectors, and the DFS representation respectively.

4.12.3 Alignment Extraction between Two DFS, *X*1 and *X*2

We refer to the alignment between $X1$ and $X2$ as the circular shift to $X2$ that would maximize the correlation between the two DFS' in the perceptual weighted domain. Let $X1p$ and $X2p$ be the pole-filtered version of $X1$ and $X2$ through the LPC filter as described in Section 4.12.4.

4.12.4 Pole-Filtering a DFS

Pole-filtering is represented in a generic case for a DFS X , LPC coefficients A , and output Xp by the following Equation:

$$Xp.a(k) = \dfrac{X.a(k)\cdot\left(1 + \sum_{v=0}^{9} A(v)\cos\left(\dfrac{2\pi}{L}kv\right)\right) + X.b(k)\cdot\left(\sum_{v=0}^{9} A(v)\sin\left(\dfrac{2\pi}{L}kv\right)\right)}{\left(1 + \sum_{v=0}^{9} A(v)\cos\left(\dfrac{2\pi}{L}kv\right)\right)^2 + \left(\sum_{v=0}^{9} A(v)\sin\left(\dfrac{2\pi}{L}kv\right)\right)^2}; \qquad 0 \le k \le \left\lfloor \dfrac{L}{2} \right\rfloor$$

(4.12.4-1)

$$Xp.b(k) = \frac{- X.a(k) \cdot \left(\sum_{v=0}^{9} A(v)\sin\left(\frac{2\pi}{L}kv\right) \right) + X.b(k) \cdot \left(1 + \sum_{v=0}^{9} A(v)\cos\left(\frac{2\pi}{L}kv\right) \right)}{\left(1 + \sum_{v=0}^{9} A(v)\cos\left(\frac{2\pi}{L}kv\right) \right)^2 + \left(\sum_{v=0}^{9} A(v)\sin\left(\frac{2\pi}{L}kv\right) \right)^2}; \qquad 0 \le k \le \left\lfloor \frac{L}{2} \right\rfloor$$

(4.12.4-2)

Let $X1p$ and $X2p$ be the pole-zero filtered version of $X1$ and $X2$ obtained by zero-filtering $X1p$ and $X2p$ through A_w, the weighted LPC filter, $A_w(k) = A(k)(0.7^k)$, $0 \le k < 9$

4.12.5 Zero-Filtering a DFS

$$Xp.a(k) = X.a(k) \cdot \left(1 + \sum_{v=0}^{9} A(v)\cos\left(\frac{2\pi}{L}kv\right) \right) - X.b(k) \cdot \left(\sum_{v=0}^{9} A(v)\sin\left(\frac{2\pi}{L}kv\right) \right); \qquad 0 \le k \le \left\lfloor \frac{L}{2} \right\rfloor \qquad (4.12.5\text{-}1)$$

$$Xp.b(k) = X.a(k) \cdot \left(\sum_{v=0}^{9} A(v)\sin\left(\frac{2\pi}{L}kv\right) \right) + X.b(k) \cdot \left(1 + \sum_{v=0}^{9} A(v)\cos\left(\frac{2\pi}{L}kv\right) \right); \qquad 0 \le k \le \left\lfloor \frac{L}{2} \right\rfloor$$

(4.12.5-2)

The alignment between $X1$ and $X2$ is given by

$$alignment = \arg\max_{n} \left| \sum_{k=0}^{\left\lfloor \frac{L}{2} \right\rfloor} \left(\begin{array}{l} (X1.a(k)X2.a(k) + X1.b(k)X2.b(k))\cos\left(\frac{2\pi}{L}nk\right) \\ + (X1.b(k)X2.a(k) - X1.a(k)X2.b(k))\sin\left(\frac{2\pi}{L}nk\right) \end{array} \right) (1 - 0.01|n|) \right|;$$

(4.12.5-3)

$$-\max(4, L/8) \le n \le \max(4, L/8)$$

4.12.6 Applying a Phase Shift to a DFS

A phase shift ϕ is applied to a DFS, X, according to the following Equations

$$\left. \begin{array}{l} Xs.a(k) = X.a(k)\cos(k\phi) - X.b(k)\sin(k\phi) \\ Xs.b(k) = X.a(k)\sin(k\phi) + X.b(k)\cos(k\phi) \end{array} \right\}; \qquad 0 \le k \le \left\lfloor \frac{L}{2} \right\rfloor \qquad (4.12.6\text{-}1)$$

4.12.7 Energy of a DFS, *X*

$$energy = \begin{cases} (X.a(0))^2 + 0.5 \sum_{k=1}^{\left\lfloor \frac{L-1}{2} \right\rfloor + 1} \left((X.a(k))^2 + (X.b(k))^2 \right), & \text{if } L \text{ is even} \\ (X.a(0))^2 + 0.5 \sum_{k=1}^{\left\lfloor \frac{L-1}{2} \right\rfloor} \left((X.a(k))^2 + (X.b(k))^2 \right), & \text{otherwise} \end{cases}$$

(4.12.7-1)

4.12.8 Converting a DFS from Cartesian to Polar Representation

Let X_c and X_p be the Cartesian and polar respectively.

$$X_p.a(k) = \begin{cases} \dfrac{\sqrt{(X_c.a(k))^2 + (X_c.b(k))^2}}{2}; & 1 \le k \le \left\lfloor \dfrac{L\text{-}1}{2} \right\rfloor; \\ \sqrt{(X_c.a(k))^2 + (X_c.b(k))^2}; & k = \dfrac{L}{2}; \ \textit{if L is even} \end{cases} \qquad (4.12.8\text{-}1)$$

$$X_p.b(k) = \tan^{-1}\left(\dfrac{X_c.b(k)}{\sqrt{(X_c.a(k))^2 + (X_c.b(k))^2}} \right); \qquad \begin{cases} 1 \le k \le \left\lfloor \dfrac{L\text{-}1}{2} \right\rfloor; \ \textit{if L is odd} \\ 1 \le k \le \dfrac{L}{2}; \ \textit{if L is even} \end{cases} \qquad (4.12.8\text{-}2)$$

4.12.9 Converting a DFS from Polar to Cartesian Representation

$$X_c.a(k) = \begin{cases} 2X_p.a(k)\cos(X_p.b(k)); & 1 \le k \le \left\lfloor \dfrac{L\text{-}1}{2} \right\rfloor; \\ X_p.a(k)\cos(X_p.b(k)); & k = L/2; \ \textit{if L is even} \end{cases} \qquad (4.12.9\text{-}1)$$

$$X_c.b(k) = \begin{cases} 2X_p.a(k)\sin(X_p.b(k)); & 1 \le k \le \left\lfloor \dfrac{L\text{-}1}{2} \right\rfloor; \\ X_p.a(k)\sin(X_p.b(k)); & k = L/2; \ \textit{if L is even} \end{cases} \qquad (4.12.9\text{-}2)$$

4.12.10 Normalizing the Band Energy of a DFS in Polar Representation

The average energy in a DFS between the frequencies f_1 and f_2 is

$$en = \dfrac{1}{k_2 - k_1 + 1}\sum_{k=k_1}^{k_2}(X_p.a(k))^2; \qquad f_1 < k_1\dfrac{8000}{L} \le k\dfrac{8000}{L} \le k_2\dfrac{8000}{L} \le f2 \qquad (4.12.10\text{-}1)$$

This is used to scale the DFS between a wider frequency range, $g_1 \le f_1$, and $g_2 \ge f_2$, such that the average energy between f_1, and f_2 is normalized to a value en_o as shown below

$$X_{1p}.a(k) = \sqrt{\dfrac{en_o}{en}}X_p.a(k); \qquad g_1 < k\dfrac{8000}{L} \le g_2 \qquad (4.12.10\text{-}2)$$

4.12.11 Converting the DFS Amplitudes to Equivalent Rectangular Bandwidth (ERB) Amplitudes

The ERB amplitudes of a DFS (in the polar form) are obtained as

$$ERB_X(i) = \dfrac{\sum_k X_p.a(k)}{N_i}; \qquad \textit{for all k such that } erbfreq[i] \le k\dfrac{8000}{L} < erbfreq[i+1] \qquad (4.12.11\text{-}1)$$

where $erbfreq[22] = \{0.0, 92.8061, 185.6121, 278.4182, 371.2243, 464.0304, 556.8364, 649.6425, 746.4, 853.6, 972.5, 1104.0, 1251.8, 1415.8, 1599.2, 1804.6, 2035.2, 2294.9, 2588.4, 2921.2, 3300.1, 3733.7, 4001.0\}$ contains the list of 22 frequency edges of the ERB bands.

$$N_j = \left| \left\lfloor \frac{erbfreq[j+1]}{8000/L} \right\rfloor - \left\lfloor \frac{erbfreq[j]}{8000/L} \right\rfloor \right| \quad (4.12.11\text{-}2)$$

is the number of harmonics in the frequency range between $erbfreq[i]$, and $erbfreq[i+1]$.

4.12.12 LPC Power-Spectrum

The power-spectrum of a set of LPC coefficients, A, at a set of NUMFREQ frequencies represented by $F[NUMFREQ]$ is given by

$$PowSpect[i] = \frac{1}{\left(1 + \sum_{v=0}^{9} A(v)\cos\left(\frac{2\pi F[i]}{8000}\right)\right)^2 + \left(\sum_{v=0}^{9} -A(v)\sin\left(\frac{2\pi F[i]}{8000}\right)\right)^2}; \quad 0 \le i < NUMFREQ$$

$$(4.12.12\text{-}1)$$

4.12.13 Current to Previous PPP Energy Ratio

$$res\_en\_ratio = \frac{currp\_nq\_en}{prev\_ppp\_en} \quad (4.12.13\text{-}1)$$

where $currp\_nq\_en$ is the current extract pitch period's energy, computed from $CURRP\_NQ$ using the procedure in Section 4.12.7.

4.12.14 Full Search Alignment between Two DFS

The alignment, searched in the full-range of 0 to $[0, L2)$, of a DFS $X2$, with lag $L2$, w.r.t another DFS $X1$ is given by

$$alignment = \arg\max_{n} \left(\sum_{k=0}^{\left\lfloor \frac{L2}{2} \right\rfloor} \left(\begin{array}{l} (X1_2.a(k)X2.a(k) + X1_2.b(k)X2.b(k))\cos\left(\frac{2\pi}{L}nk\right) \\ + (X1_2.b(k)X2.a(k) - X1_2.a(k)X2.b(k))\sin\left(\frac{2\pi}{L}nk\right) \end{array} \right) \right); \quad (4.12.14\text{-}1)$$

$$0 \le n < L2$$

where $X1_2$ is a modified version of $X1$ such that the lag of $X1_2$ is equal to that of $X2$, if $L1 < L2$, achieved using zero-padding shown in Section 4.12.17.

4.12.15 Fine Search Alignment between Two DFS

The alignment, searched in the range of $[0, L2)$ (i.e., around an expected alignment of n), of a DFS $X2$, with lag $L2$, w.r.t another DFS $X1$ is given by

$$alignment = \arg\max_{n} \left(\sum_{k=0}^{\left\lfloor \frac{L}{2} \right\rfloor} \left(\begin{array}{l} (X1_2.a(k)X2.a(k) + X1_2.b(k)X2.b(k))\cos\left(\frac{2\pi}{L}nk\right) \\ + (X1_2.b(k)X2.a(k) - X1_2.a(k)X2.b(k))\sin\left(\frac{2\pi}{L}nk\right) \end{array} \right) (1 - 0.01|n|) \right);$$

$$-4 \le n \le 4; \; n \text{ increasing in steps of } 0.25$$

$$(4.12.15\text{-}1)$$

where $X1_2$ is a modified version of $X1$ such that the lag of $X1_2$ is equal to that of $X2$, if $L1 < L2$, achieved using zero-padding shown in Section 4.12.17.

4.12.16 Weighted Alignment Search between Two DFS, *X*1 and *X*2

We refer to the alignment between $X1$ and $X2$ as the circular shift to $X2$ that would maximize the correlation between the two DFS' in the perceptual weighted domain. The alignment is searched in the range of $[n - 4, n + 4]$, i.e., around an expected alignment of n in the perceptually weighted domain. Let $X1p$ and $X2p$ be the pole-filtered version of $X1$ and $X2$ through the LPC filter as described in Section 4.12.4.

Let $X1p$ and $X2p$ be the pole-zero filtered version of $X1$ and $X2$ obtained by zero-filtering $X1p$ and $X2p$ through A_w, the weighted LPC filter, $A_w(k) = A(k)(0.7^k)$, $0 \le k < 9$ as described in Section 4.12.5.

The alignment between $X1$ and $X2$ is given by

$$weighted\_alignment = \arg\max_n \left(\left| \sum_{k=0}^{\left\lfloor \frac{L}{2} \right\rfloor} \left(\begin{array}{l} (X1.a(k)X2.a(k) + X1.b(k)X2.b(k))\cos\left(\frac{2\pi}{L}nk\right) \\ + (X1.b(k)X2.a(k) - X1.a(k)X2.b(k))\sin\left(\frac{2\pi}{L}nk\right) \end{array} \right) \right| (1 - 0.01|n|) \right);$$

$$E - \max(6, 0.15 \cdot L) \le n \le E + \max(6, 0.15 \cdot L)$$
$$n \text{ increases in steps of } 0.25 \text{ if } L < 60$$
$$n \text{ increases in steps of } 0.5 \text{ if } L \ge 60$$

(4.12.16-1)

4.12.17 Zero Padding a DFS to Have the Same Lag as another DFS

$$X1_2.a(k) = \begin{cases} X1.a(k); & 0 \le k < \dfrac{L1}{2} \\ 0; & \dfrac{L1}{2} \le k \le \dfrac{L2}{2} \end{cases}$$

(4.12.17-1)

$$X1_2.b(k) = \begin{cases} X1.b(k); & 0 \le k < \dfrac{L1}{2} \\ 0; & \dfrac{L1}{2} \le k \le \dfrac{L2}{2} \end{cases}$$

(4.12.17-2)

4.12.18 Band-Limited Correlation between Two DFS

The correlation between two DFS' focused in between frequencies $f1 < f < f2$ is given by

$$corr = \sum_k \left(\begin{array}{l} (X1_2.a(k)X2.a(k) + X1_2.b(k)X2.b(k)) \\ + (X1_2.b(k)X2.a(k) - X1_2.a(k)X2.b(k)) \end{array} \right); \qquad \text{for all k such that } f1 \le k\frac{8000}{L} < f2$$

(4.12.18-1)

4.12.19 Speech Domain Energy of a DFS

The speech domain energy in between frequencies $f1 \le f < f2$ corresponding to a residual domain DFS, X_p, and LPC parameters A of order M, is given by

1

$$sp\_energy = \begin{cases} \sum_k \dfrac{\left(X_p.a(k)\right)^2}{\left(\left(1+\sum\limits_{v=0}^{M-1}\left(A(v)\cos\left(\dfrac{2\pi vk}{L}\right)\right)\right)^2 + \left(\sum\limits_{v=0}^{M-1}\left(-A(v)\sin\left(\dfrac{2\pi vk}{L}\right)\right)\right)^2\right)}; \\ \qquad\qquad\qquad\qquad\qquad\qquad k=0, \text{ or } \left(k=\lfloor L/2\rfloor \text{ and } L \text{ is even}\right) \\ \sum_k \dfrac{2\cdot\left(X_p.a(k)\right)^2}{\left(\left(1+\sum\limits_{v=0}^{M-1}\left(A(v)\cos\left(\dfrac{2\pi vk}{L}\right)\right)\right)^2 + \left(\sum\limits_{v=0}^{M-1}\left(-A(v)\sin\left(\dfrac{2\pi vk}{L}\right)\right)\right)^2\right)}; \quad other\ k \end{cases}$$

for all k such that $f1 \le k\dfrac{8000}{L} < f2$

2
(4.12.19-1)

4.12.20 Grouping DFS Harmonics into ERB Slots

The representative (central) frequency of the ERB in terms of the harmonics of a DFS is the arithmetic mean of the harmonic frequencies in the ERB band given by, for $0 \le j < 22$.

$$erbhfreq(j) = \begin{cases} \dfrac{k\dfrac{8000}{L}}{N_j}; & N_j \ne 0, \text{ for all } k \text{ such that } erbfreq[j] \le k\dfrac{8000}{L} < erbfreq[j+1] \\ 0; & N_j = 0 \end{cases}$$

(4.12.20-1)

where as earlier

$$N_j = \left\lfloor \dfrac{erbfreq\ [j+1]}{8000\big/L} \right\rfloor - \left\lfloor \dfrac{erbfreq\ [j]}{8000\big/L} \right\rfloor$$

(4.12.20-2)

is the number of harmonics in the frequency range between $erbfreq[j]$, and $erbfreq[j+1]$. Clearly, N_j can be zero for some j depending on the lag, L.

4.12.21 Zero Inserting into a DFS

Interleaving the DFS coefficients with zeros is equivalent to periodically extending the time domain waveform corresponding to the DFS. This zero insertion is described below.

$$X_o.a(k) = \begin{cases} X_i.a\left(\dfrac{k}{I}\right); & k\%I = 0 \\ 0; & otherwise \end{cases}; \qquad 0 \le k \le \left\lfloor \dfrac{I\cdot L}{2} \right\rfloor$$

where the lag of the input DFS is Li, and the lag of the output DFS is I*Li.

4.12.22 Peak to Average Ratio of PPP from its DFS

The DFS X is first converted to the time-domain signal. We compute the positive and negative peak-to-average of x as follows:

$$max\_positive = \max\left((x(j))^2\right) \qquad \text{for all } j \text{ in } [0,L) \text{ such that } x(j) \geq 0 \tag{4.12.22-1}$$

$$max\_negative = \max\left((x(j))^2\right) \qquad \text{for all } j \text{ in } [0,L) \text{ such that } x(j) < 0 \tag{4.12.22-2}$$

$$peaktoaverage\_pos = \sqrt{\frac{max\_positive \cdot L}{\sum_{j=0}^{L-1}(x(j))^2}} \tag{4.12.22-3}$$

4.12.23 Converting ERB Amplitudes to DFS Amplitudes

The DFS amplitudes are reconstructed from the ERB amplitudes by linear interpolation.

$$X.a(k) = \frac{currerbQ(j)\left(k\frac{8000}{L} - erbhfreq(m-1)\right) + currerbQ(j-1)\left(erbhfreq(m) - k\frac{8000}{L}\right)}{erbhfreq(m) - erbhfreq(m-1)} \tag{4.12.23-1}$$

Using the procedure mentioned in Section 4.12.10, the lower (0 to 1104.5 Hz) and higher frequency bands (1104.5 to 4000 Hz) of the polar form of $PREV\_PPP\_E$ are first independently normalized using the energies $last\_en_l$ (the energy between 92.0 and 1104.5), and $last\_en_u$ (the energy between 1004.5 to 3300) respectively.

4.12.24 Prototype Pitch Period Extraction

While the goal is to extract a prototype pitch period from the end of the current frame of the formant residual, it is not always as direct as to extract the last L values of the current frame. This is because if the high energy pitch spike regions happen to be very close to the ends of the final pitch period, i.e., either close to the end of the frame, or beginning of the final L samples, the last L values may not contain the main pitch spike completely. This would lead to artifacts during the synthesis. In such cases, the solution is to extract the prototype pitch period is from slightly ahead in the frame such that the main pitch spike is obtained not from the end of the frame but from the last but one pitch period of the frame.

Let $spike$ ($0 \leq spike < L$) be the location of the maximum absolute value in the last L samples of the current frame of modified residual, $r(n)$. That is,

$$spike = \underset{k=0}{\overset{L-1}{\arg\max}}\left(abs(r(160 - L + k))\right) \tag{4.12.24-1}$$

Let $range$ be the number of samples that is considered too close to the end points of the last pitch period, such that,

Case 1: Spike too close to beginning of last L samples i.e., $spike < range$.

$$currp\_nq\_extracted(n) = \begin{cases} r(160 - L + n); & 0 \leq n < L + spike - range \\ r(160 - 2L + n); & L + spike\text{-}range \leq spike < L \end{cases} \tag{4.12.24-2}$$

Case 2: Spike too close to end of last L samples i.e., $spike \geq L - range$.

$$currp\_nq\_extracted(n) = \begin{cases} r(160 - L + n); & 0 \leq n < spike - range \\ r(160 - 2L + n); & 160\text{-}2L + spike\text{-}range \geq 0 \\ r(160 - L + n); & otherwise \end{cases}; \quad spike - range \leq n < L \tag{4.12.24-3}$$

Case 3: Spike not near boundary.

$$currp\_nq\_extracted(n) = r(160 - L + n); \qquad 0 \le n < L \qquad (4.12.24\text{-}4)$$

Since the extracted pitch period doesn't always belong to the last L samples of the frame, the energy of the extracted period is matched with the energy of the last L samples of the frame as follows:

$$currp\_nq\_extracted(n) = \sqrt{\frac{\sum_{nn=0}^{L-1}(r(160 - L + nn))^2}{\sum_{nn=0}^{L-1}(currp\_nq\_extracted(nn))^2}}\, currp\_nq\_extracted(n); \qquad 0 \le n < L \qquad (4.12.24\text{-}5)$$

Let *CURRP_NQ_EXTRACTED* be the Discrete Fourier Series (DFS) of the above extracted prototype pitch period.

Further, since the extracted pitch period doesn't always belong to the last L samples of the frame, the extracted period is circularly rotated so that it is aligned with the last L samples of the frame. This alignment is performed through correlation search followed by phase shifting. Let $tmp(n) = r(160 - L + n); \quad 0 \le n < L$ be a temporary signal containing the last L samples of the frame, and *TMP* be the DFS of *tmp* . The alignment, α , required between *CURRP_NQ_EXTRACTED*, and *TMP* is obtained using the following generic method, and this is phase shift, α , is applied (using the method mentioned below) to *CURRP_NQ_EXTRACTED* to yield *CURRP_NQ*.

4.12.25 Restoring PPP Memories if Last Frame Is Non-PPP or Full-Rate PPP

If the previous frame was not a PPP frame, then the previous prototype pitch period's DFS, *PREV_PPP*, is computed from the last PL values of the pitch memory (excitation memory).

The phase offset, *ph_offset_E* used in the PPP synthesis is set to 0.

The previous prototype pitch period's energy, *prev_ppp_en* is computed from *PREV_PPP* using the procedure in Section 4.12.7.

Using the procedure mentioned in Section 4.12.10, the lower (0 to 1104.5 Hz) and higher frequency bands (1104.5 to 4000 Hz) of the polar form of *PREV_PPP* are first independently normalized using the energies $last\_en_l$ (the energy between 92.0 and 1104.5), and $last\_en_u$ (the energy between 1004.5 to 3300) respectively.

Further, the previous frame's lower and higher band energy gains are computed as

$$lastLgainE = \log_{10}(L \cdot last\_en_l) \qquad (4.12.25\text{-}1)$$

$$lastHgainE = \log_{10}(L \cdot last\_en_h) \qquad (4.12.25\text{-}2)$$

The normalized previous PPP's DFS in the polar form is used to obtain the last frame's ERB amplitudes, *lasterbE* , as described in Section 4.12.11.

4.12.26 Correlation between Current and Previous PPP

To determine the correlation between the current and previous PPP, *PREV_PPP* is first aligned using a full-search w.r.t *CURRP_NQ* as described in Section 4.12.14, and rotated as mentioned in Section 4.12.6 (equivalently, *CURRP_NQ* can be rotated in the opposite direction to yield *CURRP_NQ_rotated*). The residual correlation, *res_corr* , is then calculated between 100 and 3700 Hz, as mentioned in Section 4.12.18.

In addition, the pole-filtered (as in Section 4.12.4) version of $PREV\_PPP\_E$, and $CURRP\_NQ\_rotated$ are used to compute the speech domain energy ratio, sp_en_ratio, using a procedure same as the one used to calculate res_en_ratio as in Section 4.12.13.

4.12.27 Open-Loop Coding Scheme Change

Depending on some features of the current frame that detect the suitability of PPP processing, PPP processing is discontinued (we use the notation $PPP\_MODE\_E = 'B'$, where PPP_MODE_E represents the PPP processing mode, and can take on three values $'F', 'Q', 'B'$ representing Full-Rate PPP, Quarter-Rate PPP processing, and Full-Rate CELP processing respectively) and CELP processing is applied to code the current frame. These rules/features satisfying discontinue of PPP processing are described below.

If $(\Delta L = L - PL) < -7$, or $(\Delta L = L - PL) > 8$

If the anchor operating point, $OP = 1$, or, $OP = 2$

If $NS\_SNR < 25$, $res\_en\_ratio > 5$, AND $res\_corr < 0.65$

Else if $NS\_SNR \geq 25$, $res\_en\_ratio > 3$, AND $res\_corr < 1.2$

Rapid ramp-down frame:

If the anchor operating point, $OP = 1$

If $NS\_SNR < 25$, AND $res\_en\_ratio < 0.025$

Else if $NS\_SNR \geq 25$, AND $res\_en\_ratio < 0.075$

If the anchor operating point, $OP = 2$

If $NS\_SNR < 25$, AND $res\_en\_ratio < 0.025$

Else if $NS\_SNR \geq 25$, AND $res\_en\_ratio < 0.075$

Else if $NS\_SNR \geq 25$, AND $MIN(res\_en\_ratio, sp\_en\_ratio) < 0.075$,

AND $res\_corr < 0.5$

Rapid ramp-up frame:

If the anchor operating points, $OP = 1$, or $OP = 2$

If $NS\_SNR < 25$, AND $res\_en\_ratio > 14.5$

Else if $NS\_SNR \geq 25$, AND $res\_en\_ratio > 7$

If the anchor operating point, $OP = 2$

If $NS\_SNR \geq 25$, AND $res\_corr \leq 0.0$

If the previous frame was Quarter-Rate NELP, or silence

4.12.28 Quarter-Rate PPP Quantization

The Excitation memory from the previous frame is extended into the current frame as described in the Section 4.11.4.8. We call this signal $acb(n)$. A prototype pitch period of length L is extracted from $acb(n)$ as described in Section 4.12.24, and let ACB be the DFS of the extracted PPP from the extended excitation memory.

4.12.28.1 Delta Delay Quantization

The pitch lag for the PPP in QPPP processing is quantized differentially as

$$DELTADELAY = L - PL + 7 .$$

4.12.28.2 ERB Differential Amplitude Quantization

Let $CURRP\_NQ_p$ be the polar representation of the DFS of $CURRP\_NQ$.

The speech domain energy of $CURRP\_NQ_p$ is computed in two frequency bands $[0, 1104.5)$, and $[1104.5, 4000)$ is computed as shown in Section 4.12.19. Let these be denoted by sp_en1, and sp_en2 respectively. We obtain two speech domain log ratios corresponding to the two frequency bands as,

$$sp\_en\_log\_ratio1 = \frac{\log_{10}(sp\_en1)}{\log_{10}(sp\_en1) + \log_{10}(sp\_en2)} \tag{4.12.28-1}$$

$$sp\_en\_log\_ratio2 = \frac{\log_{10}(sp\_en2)}{\log_{10}(sp\_en1) + \log_{10}(sp\_en2)} \tag{4.12.28-2}$$

Using the procedure mentioned in Section 4.12.10, the lower (0 to 1104.5 Hz) and higher frequency bands (1104.5 to 4000 Hz) $CURRP\_NQ_p$ are first independently normalized using the energies $curr\_en_l$ (the energy between 92.0 and 1104.5), and $curr\_en_u$ (the energy between 1004.5 to 3300) respectively. Let the normalized DFS be $CURRP\_NQ_p\_norm$.

Further, the current frame's lower and higher band energy gains are computed as

$$currLgainE = \log_{10}(L \cdot curr\_en_l) \tag{4.12.28-3}$$

$$currHgainE = \log_{10}(L \cdot curr\_en_h) \tag{4.12.28-4}$$

The change in the lower and higher band energy gains between the current and the previous frames are given by,

$$\Delta LgainE = currLgainE - prevLgainE \tag{4.12.28-5}$$

$$\Delta HgainE = currHgainE - prevHgainE \tag{4.12.28-6}$$

These delta lower and higher band energy gains are vector quantized using the speech domain log ratios as weights in the MSE calculation.

$$error(j) = \begin{pmatrix} sp\_en\_log\_ratio1 \cdot \left(\left| \Delta LgainE - PowerCB[j][0] \right| \right) + \\ sp\_en\_log\_ratio2 \cdot \left(\left| \Delta HgainE - PowerCB[j][1] \right| \right) \end{pmatrix} \tag{4.12.28-7}$$

Where $PowerCB[64][2]$ is the codebook for the delta-band-energy VQ.

The error is further conditioned as

$$error(j) = 0.8 \cdot error(j) \text{ if } \Delta LgainE > PowerCB[j][0] \text{ and } \Delta HgainE > PowerCB[j][1] \qquad (4.12.28-8)$$

The best VQ codebook index is given by

$$POWER\_IDX = \underset{j}{\arg\min}(error(j)); \qquad 0 \le j < 64 \qquad (4.12.28-9)$$

The normalized current PPP's DFS in the polar form, $CURRP\_NQ_p\_norm$, is used to obtain the current frame's ERB amplitudes, $currerb$, as described in Section 4.12.11.

The current and previous lags are used as mentioned in Section 4.12.20 to give two sets of ERB slot representative frequencies $erbhfreq\_c$, and $erbhfreq\_p$ respectively, and $N_{j,c}$, and $N_{j,p}$ being the number of harmonics in the different ERB bands.

Before using it in the amplitude quantization, the empty slots in the previous frame's ERB, $lasterbE$ are populated based on whether $PL > L$ or $L > PL$ as follows

$$lasterbE(j) = \begin{cases} \begin{cases} lasterbE(k); & \text{if } N_{j,p} = 0, \text{ } k \text{ is the largest integer } < j, \text{ AND } N_{j,p} \ne 0 \\ lasterbE(j); & \text{if } N_{j,p} \ne 0 \end{cases}; PL > L \\ \begin{cases} lasterbE(k); & \text{if } N_{j,p} = 0, \text{ } k \text{ is the smallest integer } > j, \text{ AND } N_{j,p} \ne 0 \\ lasterbE(j); & \text{if } N_{j,p} \ne 0 \end{cases}; PL < L \end{cases}$$

$$0 \le j < 22 \qquad (4.12.28-10)$$

Next, the delta between the current and previous ERB amplitudes is computed as

$$\Delta erb(j) = currerb(j) - lasterbE(j); \qquad 0 \le j < 22 \qquad (4.12.28-11)$$

The weighted LPC power-spectrum, $PowSpectW$, is computed using A_w, the weighted LPC filter, $A_w(k) = A(k)(0.78^k)$, $0 \le k < 9$, using the method described in Section 4.12.12.

The Δerb is Vector Quantized in two splits, the first split corresponding to the ERB indices 1 to 10, and the second split corresponding to the ERB indices 11 to 19. The best indice for the first split is selected as

$$errordist 1(i) = \begin{cases} (\Delta erb(i))^2; & AmpCB 1[k][i-1] < -lasterbE(i) \\ (currerb(i) - AmpCB 1[k][i-1])^2; & \text{otherwise} \end{cases}$$

$$AmpIDX 1 = \left\{ \underset{k}{\arg\min}\left(\sum_{i=1}^{10} w1_i (PowSpectW(i) \cdot errordist 1(i)) \right) \right\}$$

$$(4.12.28-12)$$

$$where$$

$$w1_i = \begin{cases} 0.9; & if \ AmpCB1[k][i-1] < \Delta erb(i) \\ 1.0; & otherwise \end{cases}$$

$$i \ such \ th \ at \ N_i \ne 0$$
$$0 \le k < 64$$

The best indices for the second split is selected as

$$errordist\ 2(i) = \begin{cases} (\Delta erb\,(i))^2; & AmpCB\ 2[k][i-11] < -lasterbE\ (i) \\ (currerb\ (i) - AmpCB\ 2[k][i-1])^2; & \text{otherwise} \end{cases}$$

$$AmpIDX\ 2 = \left\{ \arg \min_{k} \left(\sum_{i=11}^{19} w1_i \left(PowSpectW\ (i) \cdot errordist\ 2(i) \right) \right) \right\}$$

where

$$w1_i = \begin{cases} 0.9; & if\ AmpCB1\ [k][i-11] < \Delta erb\,(i) \\ 1.0; & otherwise \end{cases}$$

$$i\ such\ that\ N_i \neq 0$$
$$0 \leq k < 64$$

(4.12.28-13)

where $AmpCB1[10][64]$, and $A\_AmpCB2[9][64]$ are the codebook for the VQ for the two splits.

4.12.28.3 ERB Differential Amplitude Dequantization

The current and previous lags are used as mentioned in Section 4.12.20 to give two sets of ERB slot representative frequencies $erbhfreq\_c$, and $erbhfreq\_p$ respectively, and $N_{j,c}$, and $N_{j,p}$ being the number of harmonics in the different ERB bands. Before using it in the amplitude quantization, the empty slots in the previous frame's ERB, $lasterbE$ are populated based on whether $PL > L$ or $L > PL$ as follows:

$$lasterbE(j) = \begin{cases} \begin{cases} lasterbE(k); & if\ N_{j,p} = 0,\ k\ is\ the\ largest\ integer\ < j,\ AND\ N_{j,p} \neq 0 \\ lasterbE(j); & if\ N_{j,p} \neq 0 \end{cases} ; PL > L \\ \begin{cases} lasterbE(k); & if\ N_{j,p} = 0,\ k\ is\ the\ smallest\ integer\ > j,\ AND\ N_{j,p} \neq 0 \\ lasterbE(j); & if\ N_{j,p} \neq 0 \end{cases} ; PL < L \end{cases}$$

(4.12.28-14)

$$0 \leq j < 22$$

The quantized ERB amplitudes are dequantized according to the following Equation:

$$currerbQ(i) = \begin{cases} MAX(0, lasterbE(i) + AmpCB1[AmpIDX1][i-1]); & i\ such\ that\ N_i \neq 0,\ and\ 0 \leq i < 10 \\ MAX(0, lasterbE(i) + AmpCB2[AmpIDX1][i-11]); & i\ such\ that\ N_i \neq 0,\ and\ 10 \leq i < 19 \\ 0; & N_i = 0 \end{cases}$$

$$currerbQ(0) = 0.3 \cdot currerbQ(1)$$
$$currerbQ(20) = 0.3 \cdot currerbQ(19)$$
$$currerbQ(21) = 0$$

(4.12.28-15)

4.12.28.4 Evaluating the Quality of Quantization

The Quarter-Rate PPP amplitude quantization is evaluated for its performance to detect outliers of large errors, and these would be bumped up to Full-Rate. The following distance metric is computed after the quantization:

$$amerror = \frac{\sum_{i=1}^{10} |currerb(i) - currerbQ(i)|}{N_i}; \qquad i\ such\ that\ N_i \neq 0$$

(4.12.28-16)

If $amperror > 0.47$ AND $\Delta LgainE > -0.4$, Quarter-Rate PPP processing is abandoned and the processing is bumped up to Full-Rate processing.

4.12.28.5 Obtaining the Dequantized Current PPP DFS

The procedure in Section 4.12.23 is used to convert the quantized *currerbQ* to quantized current frame PPP DFS, $CURRP\_Q_pA$.

Using the procedure mentioned in Section 4.12.10, the lower (0 to 1104.5 Hz) and higher frequency bands (1104.5 to 4000 Hz) $CURRP\_Q_pA$ are first independently normalized using the energies $currq\_en_l$ (the energy of $CURRP\_Q_pA$ between 92.0 and 1104.5 calculated as in Section 4.12.7), and $currq\_en_u$ (the energy of $CURRP\_Q_pA$ between 1004.5 to 3300 calculated as in Section 4.12.7) respectively. Let the normalized DFS be $CURRP\_Q_pB$.

This normalized DFS $CURRP\_Q_pB$ is converted back to ERB amplitudes as in Section 4.12.11, and backed up into *lasterbE*.

The lower and upper quantized band energy gains are given by

$$currLgainE = PowerCB[POWER\_IDX][0] + prevLgainE \qquad (4.12.28\text{-}17)$$

$$currHgainE = PowerCB[POWER\_IDX][1] + prevHgainE \qquad (4.12.28\text{-}18)$$

These are backed into *prevLgainE*, and *prevHgainE*.

Using the procedure mentioned in Section 4.12.10, the lower (0 to 1104.5 Hz) and higher frequency bands (1104.5 to 4000 Hz) of $CURRP\_Q_pA$ are first independently normalized using the energies $currq\_en_l$ (the energy of $CURRP\_Q_pA$ between 92.0 and 1104.5 calculated as in Section 4.12.7) to $\left(10^{currLgainE}\right)\!/\!L$, and $currq\_en_u$ (the energy of $CURRP\_Q_pA$ between 1004.5 to 3300 calculated as in Section 4.12.7) to $\left(10^{currHgainE}\right)\!/\!L$ respectively. This gives the amplitudes of the quantized current PPP DFS $CURRP\_Q_p$

4.12.28.6 Copying the Phase Spectrum of Previous PPP DFS

In Quarter-Rate PPP, the phase of the current DFS is set to be equal to the phase of the previous DFS after it is extended into the current frame. That is, the phase of $CURRP\_Q_p$ (which is in polar form) is obtained as follows:

$$CURRP\_Q_p.b(k) = ACB_p.b(k); \qquad 0 \le k \le \lfloor L/2 \rfloor \qquad (4.12.28\text{-}19)$$

where ACB_p is the polar representation of ACB.

In addition a target DFS, $TARGET\_Q_p$, whose amplitude is quantized, but which has the unquantized prototype's phases is computed as follows:

$$TARGET\_Q_p.a(k) = CURRP\_Q_p.a(k); \qquad 0 \le k \le \lfloor L/2 \rfloor \qquad (4.12.28\text{-}20)$$

$$TARGET\_Q_p.b(k) = CURRP\_NQ_p.b(k); \qquad 0 \le k \le \lfloor L/2 \rfloor \qquad (4.12.28\text{-}21)$$

The Cartesian representations of $CURRP\_Q_p$, and $TARGET\_Q_p$ (let's call these $CURRP\_Q$, and $TARGET\_Q$ respectively) are used in the fine alignment method mentioned in Section 4.12.15 to shift $TARGET\_Q$ to be aligned with $CURRP\_Q$.

4.12.28.7 Closed Loop Bump-Up from QPPP to Full-Rate CELP Coding Scheme

After the above Quarter-Rate PPP quantization, the following features are used to determine whether the QPPP scheme is sufficient for the current frame. If not, Full-Rate CELP coding scheme is pursued.

- ~~Operating~~ Capacity operating point average rate

- Signal-to-noise ratio of the current frame from noise suppression

- Positive and negative peak to average of $CURRP\_Q$ (quantized amplitudes, phase copy from previous PPP)

- Positive and Negative peak to average of $TARGET\_Q$ (true amplitudes, true current phase)

- Energy of the unquantized current PPP

- Energy of the previous quantized PPP

- Positive and negative peak to average of $CURRP\_NQ$ (quantized amplitudes, true current phase)

- LPC prediction gain

- SNR between $TARGET\_Q$, and $CURRP\_Q$

- $CURRP\_NQ$ residual energy ratio

- $CURRP\_NQ$ speech energy ratio

The basic idea is to make sure that some features of the $CURRP\_NQ$, the true PPP, should not deviate too much upon amplitude quantization, and/or, the phase quantization, i.e., copying of the phase from the past PPP extended into the current frame. The tolerances for the deviation are dependent on the Capacity Operating Point Average, noise suppression SNR, and LPC prediction gain.

4.12.29 FPPP DFS Amplitude Vector Quantization

Let $CURRP\_NQ_p$ be the polar representation of the DFS of $CURRP\_NQ$.

The speech domain energy of $CURRP\_NQ_p$ is computed in two frequency bands $[0,1104.5)$, and $[1104.5,4000)$ is computed as shown in Section 4.12.19. Let these be denoted by sp_en1, and sp_en2 respectively. We obtain two speech domain log ratios corresponding to the two frequency bands as,

$$sp\_en\_\log\_ratio1 = \frac{\log_{10}(sp\_en1)}{\log_{10}(sp\_en1) + \log_{10}(sp\_en2)} \qquad (4.12.29\text{-}1)$$

$$sp\_en\_\log\_ratio2 = \frac{\log_{10}(sp\_en2)}{\log_{10}(sp\_en1) + \log_{10}(sp\_en2)} \qquad (4.12.29\text{-}2)$$

Using the procedure mentioned in Section 4.12.10, the lower (0 to 1104.5 Hz) and higher frequency bands (1104.5 to 4000 Hz) $CURRP\_NQ_p$ are first independently normalized using the energies $curr\_en_l$ (the energy between 92.0 and 1104.5), and $curr\_en_u$ (the energy between 1004.5 to 3300) respectively. Let the normalized DFS be $CURRP\_NQ_p\_norm$.

Further, the current frame's lower and higher band energy gains are computed as

1    $$currLgainE = \log_{10}(L \cdot curr\_en_l) \qquad (4.12.29\text{-}3)$$

2    $$currHgainE = \log_{10}(L \cdot curr\_en_h) \qquad (4.12.29\text{-}4)$$

3   These lower and higher band energy gains are vector quantized using the speech domain log ratios as weights in the
4   MSE calculation.

5    $$error(j) = \begin{pmatrix} sp\_en\_\log\_ratio1 \cdot (|currLgainE - A\_PowerCB[j][0]|) + \\ sp\_en\_\log\_ratio2 \cdot (|currHgainE - A\_PowerCB[j][1]|) \end{pmatrix} \qquad (4.12.29\text{-}5)$$

6   where $A\_PowerCB[256][2]$ is the codebook for the delta-band-energy VQ.

7   The error is further conditioned as

8    $$error(j) = 0.8 \cdot error(j) \text{ if } currLgainE > A\_PowerCB[j][0] \text{ and}$$
9    $$currHgainE > A\_PowerCB[j][1] \qquad (4.12.29\text{-}6)$$

10   The best VQ codebook index is given by

11   $$POWER\_IDX = \arg\min_{j}(error(j)); \qquad 0 \le j < 256 \qquad (4.12.29\text{-}7)$$

12   The normalized current PPP's DFS in the polar form, $CURRP\_NQ_p\_norm$, is used to obtain the current frame's ERB
13   amplitudes, $currerb$, as described in Section 4.12.11.

14   The current lag are used as mentioned in Section 4.12.20 to give a set of ERB slot representative frequencies
15   $erbhfreq\_c$, and $N_{j,c}$, being the number of harmonics in the different ERB bands.

16   The weighted LPC power-spectrum, $PowSpectW$, is computed using A_w, the weighted LPC filter,
17   $A_w(k) = A(k)(0.78^k)$, $0 \le k < 9$, using the method described in Section 4.12.12.

18   The $currerb$ is Vector Quantized in three splits, the first split corresponding to the ERB indices 0 to 4, the second split
19   corresponding to the ERB indices 5 to 9, and the third split corresponding to the ERB indices 10 to 18.

20   The best indices for the first split is selected as

$$A\_AmpIDX1 = \arg\min_{k}\left(\sum_{i=0}^{4} w1_i \left(PowSpectW(i) \cdot (currerb(i) - A\_AmpCB1[k][i])^2\right)\right);$$

where

21   $$w1_i = \begin{cases} 0.9; & \text{if } A\_AmpCB1[k][i] < currerb(i) \\ 1.0; & \text{otherwise} \end{cases} \qquad (4.12.29\text{-}8)$$

$$i \text{ such that } N_i \ne 0$$
$$0 \le k < 64$$

22   The best indices for the second split is selected as

$$A\_AmpIDX2 = \arg\min_{k}\left(\sum_{i=5}^{9} w2_i\left(PowSpectW(i)\cdot(currerb(i) - A\_AmpCB2[k][i-6])^2\right)\right);$$

where

$$w2_i = \begin{cases} 0.9; & \text{if } A\_AmpCB2[k][i] < currerb(i) \\ 1.0; & \text{otherwise} \end{cases}$$

(4.12.29-9)

$$i \text{ such that } N_i \neq 0$$
$$0 \leq k < 64$$

The best indices for the 3rd split is selected as

$$A\_AmpIDX3 = \arg\min_{k}\left(\sum_{i=10}^{18} w3_i\left(PowSpectW(i)\cdot(currerb(i) - A\_AmpCB3[k][i-10])^2\right)\right);$$

where

$$w3_i = \begin{cases} 0.9; & \text{if } A\_AmpCB3[k][i] < currerb(i) \\ 1.0; & \text{otherwise} \end{cases}$$

(4.12.29-10)

$$i \text{ such that } N_i \neq 0$$
$$0 \leq k < 256$$

where $A\_AmpCB1[5][64]$, $A\_AmpCB2[5][64]$ and $A\_AmpCB3[9][256]$ are the codebook for the VQ for the two splits.

4.12.29.1 ERB Amplitude Dequantization

The quantized ERB amplitudes are dequantized according to the following Equation

$$currerbQ(i) = \begin{cases} A\_AmpCB1[AmpIDX1][i]; & i \text{ such that } N_i \neq 0, \text{ and } 0 \leq i < 5 \\ A\_AmpCB2[AmpIDX2][i-5]; & i \text{ such that } N_i \neq 0, \text{ and } 5 \leq i < 9 \\ A\_AmpCB3[AmpIDX3][i-10]; & i \text{ such that } N_i \neq 0, \text{ and } 10 \leq i < 19 \\ 0; & N_i = 0 \end{cases}$$

$$currerbQ(0) = 0.3 \cdot currerbQ(1)$$
$$currerbQ(20) = 0.3 \cdot currerbQ(19)$$
$$currerbQ(21) = 0$$

(4.12.29-11)

The procedure in Section 4.12.23 is used to convert the quantized *currerbQ* to quantized current frame PPP DFS, $CURRP\_Q_pA$.

Using the procedure mentioned in Section 4.12.10, the lower (0 to 1104.5 Hz) and higher frequency bands (1104.5 to 4000 Hz) of $CURRP\_Q_pA$ are first independently normalized using the energies $currq\_en_l$ (the energy of $CURRP\_Q_pA$ between 92.0 and 1104.5 calculated as in Section 4.12.7) to $(10^{currLgainE})/L$, and $currq\_en_u$ (the energy of $CURRP\_Q_pA$ between 1004.5 to 3300 calculated as in Section 4.12.7) to $(10^{currHgainE})/L$ respectively. This gives the amplitudes of the quantized current PPP DFS $CURRP\_Q_p$.

4.12.29.2 Phase Quantization

The phases of DFS, $CURRP\_Q_p$,is set to zero as follows:

$$CURRP\_Q_p.b(k) = 0; \qquad 0 \le k \le \lfloor L/2 \rfloor$$

(4.12.29-12)

A target DFS, $TARGET\_Q_p$, whose amplitude is quantized, but which has the unquantized prototype's phases is computed as follows:

$$TARGET\_Q_p.a(k) = CURRP\_Q_p.a(k); \qquad 0 \le k \le \lfloor L/2 \rfloor$$

(4.12.29-13)

$$TARGET\_Q_p.b(k) = CURRP\_NQ_p.b(k); \qquad 0 \le k \le \lfloor L/2 \rfloor$$

(4.12.29-14)

$CURRP\_Q_p$, and $TARGET\_Q_p$ are converted to Cartesian representation as shown in Section 4.12.9, and then converted to the perceptually weighted domain by pole-filtering using A_w , the weighted LPC filter, $A_w(k) = A(k)(0.8^k), 0 \le k < 9$. Let's identify the resulting DFS' by $CURRP\_Q_S$, and $TARGET\_Q_S$ respectively.

4.12.29.2.1 Global Alignment

The alignment between $CURRP\_Q_S$, and $TARGET\_Q_S$, is searched in the full-range of 0 to $[0, L)$, as shown below.

$$alignment = \arg\max_n \left(\left\lfloor \sum_{k=0}^{\lfloor \frac{L2}{2} \rfloor} \left(\begin{pmatrix} CURRP\_Q_S.a(k)TARGET\_Q_S.a(k) \\ + CURRP\_Q.b(k)TARGET\_Q_S.b(k) \end{pmatrix} \cos\left(\frac{2\pi}{128}nk\right) \\ + \begin{pmatrix} CURRP\_Q_S.b(k)TARGET\_Q_S.a(k) \\ - CURRP\_Q_S.a(k)TARGET\_Q_S.b(k) \end{pmatrix} \sin\left(\frac{2\pi}{128}nk\right) \right) \right);$$

$$0 \le n < 128$$

(4.12.29-15)

$CURRP\_Q_S$ is then shifted by the negative of this alignment as described in Section 4.12.6. This corresponds to applying an expected linear phase shift to the PPP represented by $CURRP\_Q_S$ to maximally align with the target (quantized amplitude but true phase) PPP. $TARGET\_Q_S$.

4.12.29.2.2 Band Alignment

Since the linear phase shift is insufficient to capture the true phase of all harmonics, we compute band-focused alignments, in addition to the global alignment, in multiple bands (whose band-edges are given below).

F_BAND[18]={0,200,300,400,500,600,850,1000,1200,1400,1600,1850,2100,2375,2650,2950,3250,4000};

The band alignment between $CURRP\_Q_S$, and $TARGET\_Q_S$, is searched in the range of $[-16,16)$, for a band of frequencies $[lband, hband)$ as shown below:

$$band\_alignment(j) = \frac{16}{\eta} + \arg\max_{n}\left(\sum_{k}\left(\begin{array}{l}\left(\begin{array}{l}CURRP\_Q_S.a(k)TARGET\_Q_S.a(k)\\+ CURRP\_Q.b(k)TARGET\_Q_S.b(k)\end{array}\right)\cos\left(\frac{2\pi}{L}n\eta k\right)\\+\left(\begin{array}{l}CURRP\_Q_S.b(k)TARGET\_Q_S.a(k)\\- CURRP\_Q_S.a(k)TARGET\_Q_S.b(k)\end{array}\right)\sin\left(\frac{2\pi}{L}n\eta k\right)\end{array}\right)\right);$$

$$0 \le j < 17$$

$$-\frac{16}{\eta} \le n < \frac{16}{\eta}; n \text{ increasing in steps of 1}$$

$$\text{all } k \in \left[0, \left\lfloor\frac{L}{2}\right\rfloor\right] \text{ such that } lband(j) \le k\frac{8000}{L} < hband(j)$$

1

$$\text{where, } \eta = \begin{cases} 1; & j < 3 \\ 0.5; & 3 \le j < 17 \end{cases}$$

$$\text{and, } lband(j) = F\_BAND[j]$$
$$hband(j) = F\_BAND[j+1]$$

If for a given *lband, hband,* and *L,* there is no *k* such that $lband \le k\frac{8000}{L} < hband$,

$$band\_alignment(j) = INVALID\_ID;$$

2

3   Such band alignments with INVALID_ID are not packed.

4   The band phase shift ϕ is applied to a DFS, $CURRP\_Q_S$, according to the following equations

$$CURRP\_Q_S.a(k) = \left(\begin{array}{l}CURRP\_Q_S.a(k)\cos\left(-k\frac{2\pi}{L}\eta\left(-\frac{16}{\eta}+band\_alignment(j)\right)\right)-\\CURRP\_Q_S.b(k)\sin\left(-k\frac{2\pi}{L}\eta\left(-\frac{16}{\eta}+band\_alignment(j)\right)\right)\end{array}\right)$$

$$CURRP\_Q_S s.b(k) = \left(\begin{array}{l}CURRP\_Q_S.a(k)\sin\left(-k\frac{2\pi}{L}\eta\left(-\frac{16}{\eta}+band\_alignment(j)\right)\right)+\\CURRP\_Q_S.b(k)\cos\left(-k\frac{2\pi}{L}\eta\left(-\frac{16}{\eta}+band\_alignment(j)\right)\right)\end{array}\right);$$

$$0 \le k \le \left\lfloor\frac{L}{2}\right\rfloor$$

$$0 \le j < 17$$

$$-\frac{16}{\eta} \le n < \frac{16}{\eta}; n \text{ increasing in steps of 1}$$

for every j,

5

$$k \in \left[0, \left\lfloor\frac{L}{2}\right\rfloor\right] \text{ such that } lband(j) \le k\frac{8000}{L} < hband(j)$$

$$\text{where, } \eta = \begin{cases} 1; & j < 3 \\ 0.5; & 3 \le j < 17 \end{cases}$$

$$\text{and, } lband(j) = F\_BAND[j]$$
$$hband(j) = F\_BAND[j+1]$$

CURRP_Q_S is then zero filtered using the procedure in Section 4.12.5, and A_w, the weighted LPC filter,

$A_w(k) = A(k)(0.8^k)$, $0 \le k < 9$ to obtain the quantized Full-Rate PPP DFS, $CURRP\_Q$.

4.12.30 PPP Synthesis

Now that we have the quantized current PPP DFS, $CURRP\_Q$, and the previous DFS, $PREV\_PPP$, we generate the 160 samples of the current frame as below, adjusting the previous and current PPP for synthesis if pitch double or halving.

The running phase-offset of the previous the previous frame's PPP is first updated if the current pitch is more than the two times the previous frame's pitch lag. That is

$$phase\_offset = phase\_offset / I; \qquad if \; L/PL > 2 \; .$$

Further, if $L > 2PL$, the previous frame DFS, $PREV\_PPP$, is zero-inserted as described in Section 4.12.21, and a flag signal, *synthesis_ flag* is set to 1. This corresponds to interpolating the time-domain signal.

If $PL > L$, the current frame DFS, $CURRP\_Q$, is zero-inserted as described in Section 4.12.21, and the flag signal, *synthesis_ flag* is set to 2.

4.12.30.1 Calculating the Expected Alignment Shift

The expected alignment shift between the previous PPP and the current PPP is computed as

$$E_{rot} = rem\left(\left(\left\lfloor \left\lfloor rem\left(160, \left\lfloor \frac{L+PL}{2} \right\rfloor \right)\right\rfloor \right\rfloor + alignment\_offset \right), L\right)$$

where

$$alignment\_offset = \begin{cases} 2\pi \cdot PL \cdot phase\_offset \cdot round\left(\dfrac{L}{PL}\right); & if \; synthesis\_flag = 1 \\ 2\pi \cdot PL \cdot phase\_offset; & otherwise \end{cases}$$

4.12.30.2 Computing the Actual Alignment Shift and Shifting the Current PPP

The actual alignment between the previous PPP and the current PPP is computed around E_{rot} using the weighted alignment method described in Section 4.12.16.

Let the result of this alignment be *alignment_synthesis*. The current PPP, $CURRP\_Q$, is then phase shifted by

$$phaseshift\_synthesis = \frac{2\pi \cdot alignment\_synthesis}{L}$$ as described in Section 4.12.6.

4.12.30.3 Computation of the Cubic Phase Track for the Synthesis of One Frame of Residual

The phase track used to synthesis the whole frame of residual from the previous and current PPP is modeled using a cubic polynomial as below

$$phase\_out(n) = \begin{cases} rem(phase\_offset, 2\pi) & ;n = 0 \\ ((coef[0] \cdot n + coef[1]) \cdot n + coef[2]) \cdot n + coef[3] & ;1 \le n < 160 - L \\ phase\_out(n-1) + \dfrac{2\pi}{L} & ;1 \le n < 160 - L \end{cases}$$

where the coefficients of the cubic polynomial, $coef[4]$, are computed as below

$$coef[3] = rem(phase\_offset, 2\pi)$$

$$coef[2] = \frac{2\pi}{PL}$$

$$coef[0] = \left(N_{rem}\left(2\pi\left(\frac{1}{L} - \frac{1}{PL} \right) \right) \right) - \left(2 \cdot \left(phaseshift\_synthesis - phase\_offset - N_{rem} \cdot \frac{2\pi}{PL} + 2\pi\zeta \right) \right)$$

where, $N_{rem} = (160 - L)$, represents the initial region of the current frame not containing the current PPP (which represents the end of the frame), and

$$\zeta = round\left(\frac{phase\_offset - phaseshift\_synthesis + 0.5 N_{rem} 2\pi\left(\frac{1}{L} + \frac{1}{PL} \right)}{2\pi} \right)$$

$$coef[1] = \frac{2\pi\left(\frac{1}{L} + \frac{1}{PL} \right) - 3 N_{rem}^2 coef[0]}{2 N_{rem}}$$

4.12.30.4 Generating the Whole Frame of Residual

Using the current, and previous PPP, and the above described cubic phase track, the whole frame of residual is synthesized as described below.

The number of intermediate Pitch periods to be synthesized is given by

$$N_{synth} = round\left(\frac{N}{(\lfloor 0.5(L + PL) \rfloor)} \right)$$

The PPP DFS with the shorter lag between the current and the previous PPP is first zero-padded to the larger lag as described in Section 4.12.17. Thus, $CURRP\_Q$, and, $PREV\_PPP$, are made to be of the same pitch lag.

$r_q(n)$, the residual of the frame is generated for $0 \le n < 160$, inverse discrete Fourier series inverse transform calculated at n, of a phase-shifted (given by the phase-track) time varying DFS, $PPP\_SYNTH(n)$ that is the interpolated version of $CURRP\_Q$, and, $PREV\_PPP\_E$, and the phase track, computed as below.

$$PPP\_SYNTH(n).a(k) = (1 - w(n))(PREV\_PPP.a(k)) + w(n)(CURRP\_Q.a(k)); \quad 0 \le k \le \lfloor L/2 \rfloor$$

$$PPP\_SYNTH(n).b(k) = (1 - w(n))(PREV\_PPP.b(k)) + w(n)(CURRP\_Q.b(k)); \quad 0 \le k \le \lfloor L/2 \rfloor$$

where the time varying interpolation weight is given by

$$w(n) = \begin{cases} 1 - e^{-\left((n+1)\log\left(\frac{0.2}{L-160}\right)\right)}; & \text{if } L < 140 \\ \dfrac{n+1}{160}; & \text{otherwise} \end{cases}$$

4.12.31 Perceptual Weighting Filter Update

Although there is no closed-loop search at Rate 1 and Rate 1/4 PPP coding, it is still necessary to update the memory of the perceptual weighting filter with the new excitation determined in 4.6.7 for each subframe. Update the weighted synthesis filter memory by filtering the excitation vector, rq(n), through the weighted synthesis filter, Hwq(z), which is given in Equation 4.11.4-5.

4.13 Encoding at Rate 1 Generic Audio for SO 70 and SO 73

The Generic Audio frame type of SO 70 and SO 73 is used to code music, tones, and other types of non-speech input signals.

Input:

- The pre-processed speech input vector, $\{s'(n)\}$

Output:

- The quantized LSP indices, *LSPIDX(k)*

- The quantized MDCT coefficient (residual line spectrum) index, *RLSIDX*

- The quantized global scale factor index, *GSFIDX*

- The quantized Noise Injection level indices, *NILIDX*

Initialization:

The following state variables shall be set to zero at initialization (frame $m = 0$):

- The formant emphasis and de-emphasis filter memories

- The IMDCT overlap-and-add buffer, $y_{ov}(n)$

Processing: Figure 4.13-1 depicts the Generic Audio encoding system that is described in the following sections. Some of the processing functions are common to the Rate 1 RCELP processing, but are included again here for completeness.

s'(n) → LPC Analysis/Quantization → LSPIDX(k)

$\dot{\mathbf{A}}_q$

Residual Generation → $\varepsilon_q(n)$ → Formant Emphasis → $\varepsilon_f(n)$ → MDCT → X_k → Residual Line Spectrum/Scale Factor Quantization → RLSIDX, GSFIDX

X_c

$\varepsilon'_q(n)$ → RCELP State Update ← Formant Deemphasis ← $\varepsilon'_f(n)$ ← IMDCT ← X'_k ← Noise Injection → NILIDX
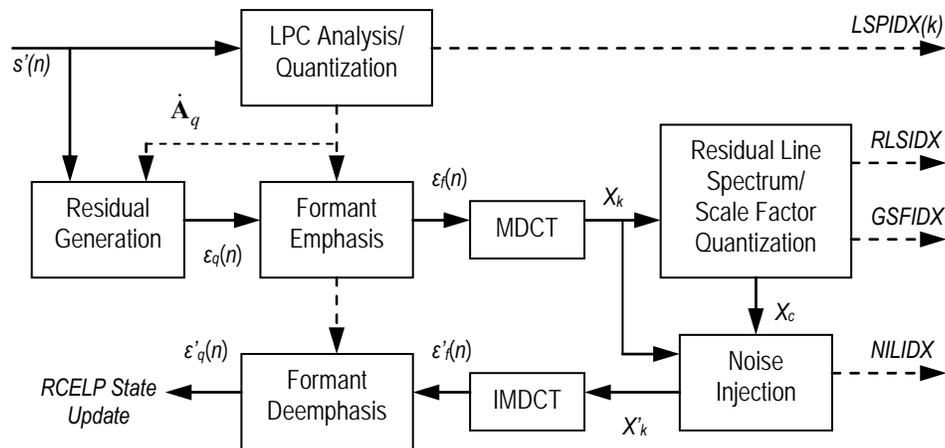
Figure 4.13-1 Rate 1 Generic Audio Encoding Block Diagram

4.13.1 LPC Analysis/Quantization

The LPC Analysis is identical to that used in the Model Parameter Estimation routines in 4.6.1. The resulting LSP parameters are quantized using the Rate 1 quantizer given in 4.10.2.

4.13.2 Short-Term Residual Generation

The short-term residual is calculated in the same way as in 4.6.2, except that the quantized LSPs $\mathbf{\Omega}_q$ are used. The short-term residual signal is designated as $\varepsilon_q(n); 0 \le n < 320$.

4.13.3 Formant Emphasis

The formant emphasis filter places perceptual weighting on the low frequency formant regions of the short-term residual signal. The transfer function of the filter is given as:

$$H_f(z) = \frac{1 + \mu_f z^{-1}}{\dot{A}_q(\gamma_f^{-1} z)}$$

(4.13.3-1)

where $\mu_f = 0.2$ is the tilt coefficient, $\gamma_f = 0.4$ is the formant emphasis coefficient, and $\dot{\mathbf{A}}_q$ is the set of interpolated LPC coefficients derived in the same manner as in 4.11.4.1 and 4.11.4.2 for the perceptual weighting filter. The filter is applied to the short term residual $\varepsilon_q(n)$ in a zero-state manner to produce the formant weighted residual $\varepsilon_f(n); 0 \le n < 320$.

4.13.4 Modified Discrete Cosine Transform (MDCT)

First, a modified sine window is defined as:

1

$$w(n) = \begin{cases} \sin\left(\dfrac{\pi}{160}\left(n+\dfrac{1}{2}\right)\right) & ;0 \le n < 80 \\ 1 & ;80 \le n < 160 \\ \sin\left(\dfrac{\pi}{160}\left(n-80+\dfrac{1}{2}\right)\right) & ;160 \le n < 240 \end{cases}$$

(4.13.4-1)

2   This window is applied to the formant weighted residual signal such that the first 160 samples correspond to the current

3   frame of the residual $\varepsilon_f(n); 80 \le n < 240$, and the last 80 samples of the window correspond to the look-ahead portion

4   of the residual $\varepsilon_f(n); 240 \le n < 320$. The MDCT frame x_n is thus formulated as:

5

$$x_n = \begin{cases} 0 & ;0 \le n < 40 \\ w(n-40)\varepsilon_f(n+40) & ;40 \le n < 280 \\ 0 & ;280 \le n < 320 \end{cases}$$

(4.13.4-2)

6   and the MDCT coefficients are then obtained as follows:

7

$$X_k = \frac{2}{\sqrt{320}} \sum_{n=0}^{319} x_n \cos\left[\frac{\pi}{160}\left(n+\frac{1}{2}+80\right)\left(k+\frac{1}{2}\right)\right]; 0 \le k < 160 .$$

(4.13.4-3)

8   ## 4.13.5 MDCT Residual Line Spectrum Quantization

9   The MDCT coefficients, refered to as the residual line spectrum, are quantized in a similar manner to the FCB factorial

10 codebook of 4.11.8.3. Basically, factorial coding of $N = {}^n FPC_m$ possible combinations can be achieved given that the

11 length n vector $\mathbf{v}$ has the properties $m = \sum_{i=0}^{n-1} |v_i|$ and all elements v_i are integral valued. That is, the sum of the

12 absolute value of the integer elements of $\mathbf{v}$ is equal to m. For this case, we wish to code an energy scaled version of X_k

13 such that:

14

$$m = \sum_{k=0}^{143} \left| \text{round}\{\gamma_m X_k\} \right| ,$$

(4.13.5-1)

15 where γ_m is a global scale factor, and the range 0 to 143 corresponds to the frequency range 0 to 3600 Hz. For this case,

16 m can be either 28 for NB or 23 for WB inputs. The value of γ_m used to achieve the above objective is determined

17 iteratively (for non-zero $\|X_k\|^2$) according to the following pseudo-code:

18 /* Initialization */

19 $e_{min} = -100, \ e_{max} = 20$

20 $e = \max\left\{e_{min}, -10\log_{10}\left(\|X_k\|^2\right)/1.2\right\}$

21 $s = +1, \ \Delta_e = 8$

22

23 /* main loop */

24 do {

1 $$\gamma_m = 10^{e/20}$$

2 $$m' = \sum_{k=0}^{143} \left| \text{round}\{\gamma_m X_k\} \right|$$

3   if $(m' == m)$ then break

4   else if $(m' > m$ and $s == +1)$ then $s = -1,\ \Delta_e = \Delta_e / 2$

5   else if $(m' < m$ and $s == -1)$ then $s = +1,\ \Delta_e = \Delta_e / 2$

6   end

7 $$e = e + s \cdot \Delta_e$$

8        $\}$ while $e \le e_{max}$ and $\Delta_e \ge \Delta_{min}$

9

10 The quantized residual line spectrum X_{cc} is then calculated as:

11 $$X_{cc}(k) = \begin{cases} \text{round}\{\gamma_m X_k\} & ; 0 \le k < 144 \\ 0 & ; 144 \le k < 160 \end{cases},$$ (4.13.5-2)

12 If, on the rare occasion, the values of m and m' are different, the line spectrum shall be modified by adding or subtracting
13 unit values to the quantized line spectrum X_{cc}. This guarantees that the resulting line spectrum can be reliably coded
14 using the factorial coding method. The output index representing the line spectrum X_{cc} is designated *RLSIDX*. This
15 index comprises 131 bits for the $^{144}FPC_{28}$ case and 114 bits for the $^{144}FPC_{23}$ case.

16 In order to address complexity issues associated with encoding and decoding vector X_{cc}, a low resolution combinatorial
17 approximation function $F'(n, r)$ shall be used in place of the standard combinatorial relation $F(n,r) = {}^nC_r = n!/r!(n-r)!$.
18 In particular, both the encoder and decoder utilize a combinatorial function generator $F'(n, r)$ having the properties
19 $F'(n,r) \ge F(n,r)$ and $F'(n,r) \ge F'(n-1,r) + F'(n-1,r-1)$, which are sufficient to uniquely encode/decode vector X_{cc}.
20 The function $F'(n,r)$ is given as:

21 $$F'(n,r) = R'\left(\sum_{i=n-r+1}^{n} P'(i) - Q'(r) \right),$$ (4.13.5-3)

22 where $P'(i)$ and $Q'(r)$ are 32 bit lookup tables given as:

23 $$P'(i) = 2^{-21} \left\lfloor 2^{21} \log_2(i) + 1 \right\rfloor, \quad i \in [1, 2, ..., 144]$$ (4.13.5-4)

24 and

25 $$Q'(r) = \begin{cases} 0, & r = 1 \\ \sum_{j=2}^{r} 2^{-14} \left\lfloor 2^{14} \log_2(j) - 1 \right\rfloor & r \in [2, ..., 28] \end{cases},$$ (4.13.5-5)

26 and where $R'(k)$ is a multi-precision integer approximation of the function $R'(k) \approx 2^k$, given as:

27 $$R'(k) = \left\lfloor 2^{k_i - 19} \left\lfloor 2^{19} K_f \right\rfloor \right\rfloor,$$ (4.13.5-6)

where $k = k_i + k_f$ is broken down into integer and fractional components of k, and $K_f = 2^{k_f}$ is a Taylor series expansion of the fractional component of k. These operations significantly reduce the complexity necessary in calculating the combinatorial expressions by replacing multi-precision multiply and divide operations with 32 bit additions and a low complexity Taylor series approximation of 2^k followed by a multiprecision shift operation. All other components of the encoding/decoding operations are identical to that in 4.11.8.3.

4.13.6 Global Scale Factor Quantization

The log domain scale factor is first recalculated based on the normalized cross correlation between the quantized line spectrum X_{cc} and the residual line spectrum X_k, or:

$$e_{sf} = -20 \log_{10} \left[\frac{\sum_{k=0}^{143} X_{cc}(k) X_k}{\sum_{k=0}^{143} X_{cc}^2(k)} \right] . \tag{4.13.6-1}$$

The log domain scale factor e_{sf} is then quantized using a uniform distribution scalar quantizer to produce e_q, such that the global scale factor index is obtained by minimizing the following:

$$GSFIDX = \underset{0 \le i \le 127}{\arg\min} \left\{ \left| e_{sf} - \left(e_{q\min} + e_{bias} + i\Delta_{eq} \right) \right| \right\} , \tag{4.13.6-2}$$

where $\Delta_{eq} = (e_{q\max} - e_{q\min})/127$ is the quantizer step size, $e_{q\min} = -95$ is the minimum quantizer level, $e_{q\max} = -25$ is the maximum quantizer level, and e_{bias} is the energy bias term given by:

$$e_{bias} = 10 \log_{10} \left(\sum_{k=0}^{143} X_{cc}^2(k) \right) . \tag{4.13.6-3}$$

The quantized scale factor is ultimately given as:

$$e_q = e_{q\min} + e_{bias} + GSFIDX \cdot \Delta_{eq} , \tag{4.13.6-4}$$

which is then applied to the quantized residual line spectrum X_{cc} to obtain the coded residual line spectrum X_c, given as:

$$X_c = 10^{-e_q/20} X_{cc} . \tag{4.13.6-5}$$

4.13.7 Noise Injection

The noise injection process provides improved audio quality by reducing the perception of tonal artifacts that may occur during undermodeling of the residual line spectrum. The noise injection process substitutes noise for zero valued elements of the coded residual line spectrum within a predetermined frequency range. The noise injection level (gain) is based on the ratio of the uncoded energy in the original residual line spectrum X_k to the total energy of the original line spectrum, and is coded using only 2 bits. This processed is described below.

A zero detection mask is first produced to control whether or not regions of the scaled line spectrum shall be injected with noise. The mask is represented as a 160 element column vector $\mathbf{z}_d$ that is defined as:

$$z_d(k) = \begin{cases} 1 ; & 40 \le k \le 143 \text{ and } X_c(k) == 0 \\ 0 ; & \text{otherwise} \end{cases} . \tag{4.13.7-1}$$

The noise injection gain γ_{ni} is then calculated using the zero detection control mask and the original residual line spectrum X_k, such that:

1

$$\gamma_{ni} = 0.9 \sqrt{\frac{\sum_{k=40}^{143} z_d(k) X_k^2}{\sum_{k=40}^{143} X_k^2}} \;.$$

(4.13.7-2)

2 The noise injection gain is then quantized using a 2 bit lookup table, defined as $\gamma_{ni\_table} = [0.15, 0.30, 0.45, 0.60]$, so that:

3

$$NILIDX = \underset{0 \le i \le 3}{\arg\min} \left\{ \left\| \gamma_{ni} - \gamma_{ni\_table}[i] \right\| \right\},$$

(4.13.7-3)

4 where *NILIDX* is the noise injection level index. The noise injection gain is then redefined as $\gamma_{ni} \leftarrow \gamma_{ni\_table}[NILIDX]$,

5 and subsequently used to generate the noise injected residual line spectrum X_k' as:

6

$$X_k' = \begin{cases} \gamma_{ni} 10^{-e_q/20} rand() \;; & 40 \le k \le 143 \ and \ z_d(k) == 1 \\ X_c(k) \;; & otherwise \end{cases},$$

(4.13.7-4)

7 where *rand*() is a uniform random number spanning [-0.5, 0.5]. To ensure the encoder and decoder are synchronized, the
8 random number generator is seeded using the residual line spectrum index, *RLSIDX*.

9 ## 4.13.8 Inverse Modified Discrete Cosine Transform (IMDCT)

10 Encoder signal reconstruction is required to properly update the RCELP state memories. More specifically, the Adaptive
11 Codebook and Zero Input Response filter require update in the event that next frame is an RCELP Rate 1 or Rate 1/2
12 frame type. Therefore, the following processes are required to generate the residual domain excitation signals.

13 The formant weighted residual estimate is obtained by applying the inverse MDCT to X_k' as:

14

$$y_n = \frac{2}{\sqrt{320}} \sum_{k=0}^{159} X_k' \cos\left[\frac{\pi}{160}\left(n + \frac{1}{2} + 80\right)\left(k + \frac{1}{2}\right)\right]; 0 \le k < 320 \,.$$

(4.13.8-1)

15 and then widowing and overlap-adding the previous frame's IMDCT output to produce the reconstructed formant
16 weighted residual:

17

$$\varepsilon_f'(n) = \begin{cases} y_{ov}(n) + w(n) y_n(n+40) & ; 0 \le n < 80 \\ y_n(n+40) & ; 80 \le n < 160 \end{cases}.$$

(4.13.8-2)

18 In the event that the previous frame was not a Generic Audio frame type, then the IMDCT overlap-add buffer is set to all
19 zeros. The IMDCT overlap-add buffer is updated for the next frame as:

20

$$y_{ov}(n) = w(n+160) y_n(n+200) \quad ; 0 \le n < 80 \,.$$

(4.13.8-3)

21 ## 4.13.9 Formant Deemphasis

22 The formant deemphasis filter removes the perceptual weighting previously put on the low frequency formant regions of
23 the short-term residual signal. The transfer function of the filter is given as:

24

$$H_d(z) = H_f^{-1}(z) = \frac{\dot{A}_q(\gamma_f^{-1} z)}{1 + \mu_f z^{-1}}$$

(4.13.9-1)

25 The filter is applied to the reconstructed formant weighted residual $\varepsilon_f'(n)$ to produce the reconstructed short-term
26 residual signal $\varepsilon_q'(n); 0 \le n < 160$. In the event that the previous frame was not a Generic Audio frame type, then the

formant deemphasis filter memory is set to all zeros. The signal $\varepsilon'_q(n); 0 \le n < 160$ is then used to update the RCELP state memories described in 4.11.4.15.

4.14 Encoding at Rate 1/4 Unvoiced for SO 68 and SO 73 NB and Rate 1/2 Unvoiced for SO 70 and SO 73 WB

NELP is used for Rate 1/4 unvoiced frames in SO 68 and Rate 1/2 unvoiced frames in SO 70 and SO 73 WB. A pseudo-random noise is colored using a set of filters, and scaled appropriately by gains in 10 subframes of 16 samples each to model the excitation.
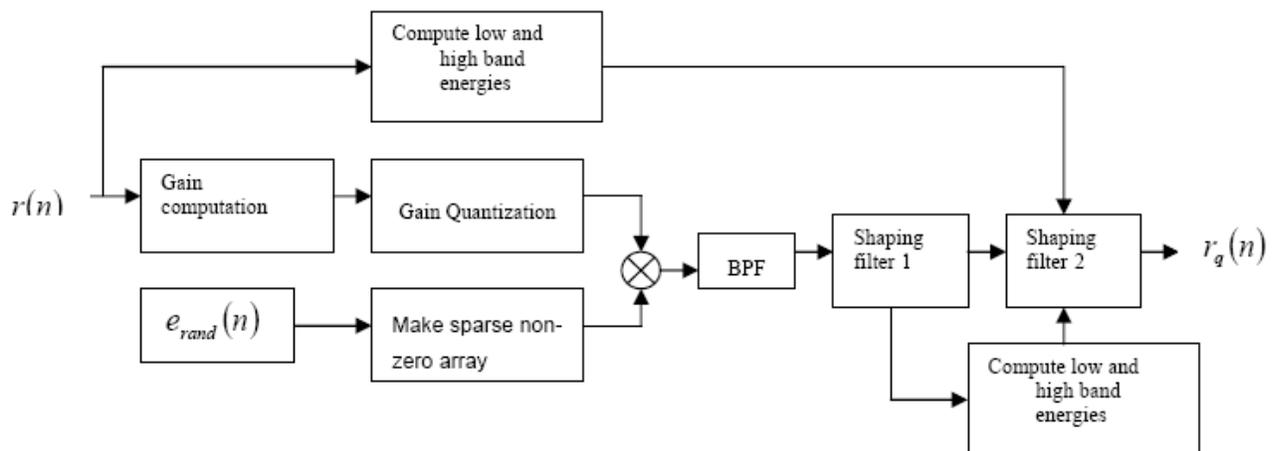
Figure 4.14-1 Excitation Coding for Rate 1/4 Unvoiced Frames for SO 68 and Rate 1/2 Unvoiced Frames for SO 70 and SO 73 WB

4.14.1 Computation and Quantization of Gains

Input:

- Residual, $r(n)$

- Quantized LSFs, $lsf_1^q(i)$

Output:

- Quantized excitation, $r_q(n)$

The energy E_i of the residual signal is computed for each of the 10 subframes $(0 \le i < 10)$ as

$$E_i = \frac{1}{16} \sum_{n=16i}^{16i+15} r^2(n)$$

(4.14.1-1)

The 10 subframe gains G_i are computed from the above energies as

$$G_i = \sqrt{E_i}$$

(4.14.1-2)

Two normalization factors for two splits $(0 \le j < 2)$ of the above 10 gains are determined by

$$G'[j] = \sqrt{\frac{1}{5} \sum_{i=0}^{4} G_{j*5+i}^2}$$

(4.14.1-3)

The above two normalization factors are vector quantized using a two dimensional codebook $UVG1CB[32][2]$ to provide an index $i\hat{G}'$ as

$$i\hat{G}' = \arg\min_k \left(\sum_{j=0}^{1} (\log 10(G'[j]) - UVG1CB[k][j])^2 \right), \quad 0 \le k < 32$$

(4.14.1-4)

The quantized normalization factors are then computed for $(0 \le j < 2)$ using

$$\hat{G}'[j] = 10^{UVG1CB[i\hat{G}'][j]}$$

(4.14.1-5)

The 10 subframe gains are then normalized using the quantized normalization factors according to

$$G''[2j+i] = \frac{G[2j+i]}{\hat{G}'[j]}; \ 0 \le i < 5, 0 \le j < 2$$

(4.14.1-6)

The 10 normalized gains are then vector quantized using a 3-dimensional codebook $UVG2CB[2][64][5]$ to provide two indices $iG''[0]$, and $iG''[1]$ such that

$$i\hat{G}''[j] = \arg\min_k \left(\sum_{i=0}^{4} (G''[2j+i] - UVG2CB[j][k][i])^2 \right); \ 0 \le k < 64; 0 \le j < 2$$

(4.14.1-7)

The quantized, normalized gains are then given by

$$\hat{G}''[2j+i] = UVG2CB[j][i\hat{G}''][i]; \quad 0 \le i < 5, 0 \le j < 2$$

(4.14.1-8)

The quantized gains used to scale the NELP excitation then are

$$\hat{G}[2j+i] = \hat{G}'[j]\hat{G}''[2j+i], \quad 0 \le i < 5, 0 \le j < 2$$

(4.14.1-9)

4.14.2 Random Number Generation

A pseudo-random vector e_{rand} of 160 samples is generated as follows,

$$e_{rand}(n) = \begin{cases} NELPseed \ , & where \\ NELPseed = (521 \times NELPseed + 259)/32768 \end{cases}; \quad 0 \le n < 160$$

(4.14.2-1)

For Rate 1/4, the seed, *NELPseed*, is initialized at the beginning of each frame to a signed 16-bit number that is derived from the bits representing the LSF.

4.14.3 Creation of Sparse Nonzero Excitation

The above pseudo-random vector $e_{rand}(n)$ is divided into 10 subframes, and each subframe is made sparse by retaining the 4 largest absolute magnitude samples, and zeroing out the remaining 12. Each of these 10 subframes of the sparse vectors is denoted by $e_{sparse}(i,n)$, $0 \le i < 10$; $0 \le n < 16$. These sparse vectors are then scaled by their appropriate gains to create 160 samples of a sparse, excitation signal

$$r_1(16i+n) = \sqrt{3}\hat{G}(i)e_{sparse}(i,n), \quad 0 \le i < 10; 0 \le n < 16$$

(4.14.3-1)

4.14.4 Shaping the Excitation

The above excitation $r_1(n)$ is filtered by a 12th order bandpass filter, in order to remove the low band and high components to yield the signal $r_2(n)$. This bandpass filter is given by the following transfer function

$$BP(z) = \frac{BP_{n0} + BP_{n1}z^{-1} + ... + BP_{n12}z^{-12}}{1 + BP_{d1}z^{-1} + ... + BP_{d12}z^{-12}}$$

(4.14.4-1)

where the coefficients BP_{ni}, and BP_{di} are given in the tables bp1_num_coef, and bp1_den_coef respectively in the C-code.

$r_2(n)$ is then filtered by the 10th order shaping filter $SF1(z)$ to give the signal $r_3(n)$. This shaping filter is given by the following transfer function

$$SF1(z) = \frac{SF1_{n0} + SF1_{n1}z^{-1} + ... + SF1_{n12}z^{-10}}{1 + SF1_{d1}z^{-1} + ... + SF1_{d12}z^{-10}}$$

(4.14.4-2)

where the coefficients $SF1_{ni}$, and $SF1_{di}$ are given in the tables shape1_num_coef, and shape1_den_coef respectively in the C-code.

$r_3(n)$ is then scaled according to the following equation to yield

$$r_4(n) = \left(\frac{\sqrt{\sum_{n=0}^{159}[r(n)]^2}}{\sqrt{\sum_{n=0}^{159}[r_3(n)]^2}} \right) r_3(n)$$

(4.14.4-3)

$r(n)$ and $r_4(n)$ are filtered by the 10th order lowpass filter $LP(z)$ to give $r_L(n)$ and $r_{4L}(n)$. Similarly, $r(n)$ and $r_4(n)$ are filtered by the 10th order highpass filter $HP(z)$ to give $r_H(n)$ and $r_{4H}(n)$.

$$LP(z) = \frac{LP_{n0} + LP_{n1}z^{-1} + ... + LP_{n12}z^{-10}}{1 + LP_{d1}z^{-1} + ... + LP_{d12}z^{-10}}$$

(4.14.4-4)

$$HP(z) = \frac{HP_{n0} + HP_{n1}z^{-1} + ... + HP_{n12}z^{-10}}{1 + HP_{d1}z^{-1} + ... + HP_{d12}z^{-10}}$$

(4.14.4-5)

where the coefficients LP_{ni}, and LP_{di} are given in the tables txlpf1_num_coef, and txlpf1_den_coef respectively in the C-code, and the coefficients HP_{ni}, and HP_{di} are given in the tables txhpf1_num_coef, and txhpf1_den_coef respectively in the C-code.

Two factors R_L, and R_H are computed as follows

$$R_L = 10 \log 10 \left(\frac{\sum_{n=0}^{n=159} [r_L(n)]^2}{\sum_{n=0}^{n=159} [r_{4L}(n)]^2} \right)$$

(4.14.4-6)

and

$$R_H = 10 \log 10 \left(\frac{\sum_{n=0}^{n=159} [r_H(n)]^2}{\sum_{n=0}^{n=159} [r_{4H}(n)]^2} \right)$$

(4.14.4-7)

A filter index F_{idx} is computed as follows from R_L, and R_H, and sent to the decoder.

$$F_{idx} = \begin{cases} 1 & ; & R_L < -3 \\ 2 & ; & R_L \geq -3 \text{ and } R_H < -3 \\ 0 & ; & otherwise \end{cases}$$

(4.14.4-8)

The shaping filter $SF(z)$ is determined from F_{idx} as follows

$$SF(z) = \begin{cases} SF1(z) & ; & F_{idx} = 0 \\ SF1(z)SF2(z) & ; & F_{idx} = 1 \\ SF1(z)SF3(z) & ; & F_{idx} = 2 \end{cases}$$

(4.14.4-9)

where, $SF2(z)$, $SF3(z)$ are

$$SF2(z) = \frac{SF2_{n0} + SF2_{n1} z^{-1} + ... + SF2_{n12} z^{-10}}{1 + SF2_{d1} z^{-1} + ... + SF2_{d12} z^{-10}}$$

(4.14.4-10)

$$SF3(z) = \frac{SF3_{n0} + SF3_{n1} z^{-1} + ... + SF3_{n12} z^{-10}}{1 + SF3_{d1} z^{-1} + ... + SF3_{d12} z^{-10}}$$

(4.14.4-11)

where the coefficients $SF2_{ni}$, $SF2_{di}$, $SF3_{ni}$, and $SF3_{di}$ are given in the tables shape2_num_coef, shape2_den_coef, shape3_num_coef, and shape3_den_coef respectively in the C-code.

Finally, $r_3(n)$ is filtered by $SF(z)$ to provide the quantized formant residual $r_q(n)$ for the Rate 1/4 frame.

4.14.5 Perceptual Weighting Filter Update

Although there is no closed-loop search at Rate 1/4 unvoiced coding, it is still necessary to update the memory of the perceptual weighting filter with the new excitation determined in 4.6.7 for each subframe. Update the weighted synthesis

filter memory by filtering the excitation vector, E(n), through the weighted synthesis filter, Hwq(z), which is given in Equation 4.11.4-5.

4.15 Encoding at Rate 1/8

Inputs: The inputs to Rate 1/8 encoding are:

- The unquantized LSPs from the current and previous frames, $\Omega(m)$ and $\Omega(m\text{-}1)$

- The quantized LSPs from the previous frame, $\Omega_q(m\text{-}1)$

- The short-term prediction residual, $\varepsilon(n)$

Outputs: The outputs of Rate 1/8 encoding are:

- The quantized LSPs for the current frame, $\Omega_q(m)$

- The quantization index corresponding to these LSPs, *LSPIDX(k)*

- The vector quantized frame energy index, *FGIDX*

State Variables Affected:

- The accumulated shift counter, τ_{acc}, is set to 0

- The pointer to the last element in shifted residual, n_m, is set to 0

Processing: Rate 1/8 encoding shall comprise 4.15.1 to 4.15.11.

4.15.1 LSP Quantization

Vector quantize the unquantized LSPs, $\Omega(m)$, using the procedure found in Sections 4.9 and 4.10 for Rate 1/8.

4.15.2 Interpolation of LSP Parameters

Interpolate the quantized LSPs over the three subframes, m', in the current frame, m, as defined in 4.6.2.1.

4.15.3 LSP to LPC Conversion

Convert the quantized, interpolated LSPs, $\dot{\Omega}_q(m)$, to quantized, interpolated LPC parameters, $\left\{ \dot{a}_q \right\}$, as described in Section 4.6.2.2 for each subframe.

4.15.4 Impulse Response Computation

Calculate the unweighted impulse response, *h(n)*, of $1/\dot{A}_q(z)$ to 54 terms for each subframe, where $\dot{A}_q(z)$ is defined as:

$$\dot{A}_q(z) = 1 - \sum_{i=1}^{10} \dot{a}_q(i) z^{-i}$$

(4.15.4-1)

4.15.5 Calculation of the Frame Energy Gain for SO 3

The frame energy gain is defined as the ratio of the energy of the impulse response to the mean of the residual for each subframe m':

$$\gamma(m') = \frac{E_\varepsilon(m')}{E_h(m')},$$

(4.15.5-1)

where the energy of the impulse response is given by:

$$E_h(m') = \sqrt{\sum_{i=0}^{L-1} h^2(n)},$$

(4.15.5-2)

and the mean $E_\varepsilon(m')$ of the residual signal $\varepsilon(n)$ is defined as:

$$E_\varepsilon(m') = \max\left\{1, \frac{1}{L}\sum_{i=0}^{L-1}|\varepsilon(n)|\right\},$$

(4.15.5-3)

where L is the subframe size (53 for subframes 0 and 1, 54 for subframe 2), $n = 0$ is defined as the index of the first sample in the current subframe, and $\varepsilon(n)$ is the residual at the current subframe.

4.15.6 Gain Quantization for SO 3

The gain vector, $\gamma(m')$, shall be quantized to 8 bits using a vector quantizer. The quantizer will assign one index for the best three-element gain vector corresponding to $\gamma(m')$. The best vector is found by calculating the error vector, $e_g(k)$, as defined by:

$$e_g(k) = \sum_{m'=0}^{2}\left(\log_{10}\left(\gamma(m')\right) - q_{\log}\left(m',k\right)\right)^2,$$

(4.15.6-1)

where $q_{log}(m',k)$ is an entry in the gain quantization codebook found in Table 8-18. The best codebook index, *FGIDX*, is defined as the index, k, at which $e_g(k)$ is minimized. This codebook index, *FGIDX*, shall be used to determine a set of quantized gains $\gamma_q(m')$:

$$\gamma_q(m') = 10^{q\log(m',FGIDX)}; \quad 0 \le m' < 3.$$

(4.15.6-2)

4.15.7 Calculation of the frame energy gain

The mean $E\varepsilon(m')$ of the residual signal $\varepsilon(n)$ is defined as:

$$E_\varepsilon(m') = \max\left\{1, \frac{1}{L}\sum_{i=0}^{L-1}|\varepsilon(n)|\right\},$$

(4.15.7-1)

where L is the subframe size (53 for subframes 0 and 1, 54 for subframe 2), n = 0 is defined as the index of the first sample in the current subframe, and $\varepsilon(n)$ is the residual at the current subframe.

4.15.8 Gain Quantization for SO 68, SO 70, and SO 73 NB Mode

The gain shall be quantized to 6 bits using a scalar quantizer. The best gain index is found by minimizing the error, eg, as defined by:

$$FGIDX = \arg\min_i \left(\log\left(\max_{m'}\left(E_t(m')\right)\right) - \left(MINLOGE + i \cdot \left(\frac{MAXLOGE - MINLOGE}{NUMQLEVELS} \right)\right)\right) \qquad (4.15.8\text{-}1)$$

where $0 \le i < NUMQLEVELS$, the indices of gain quantization, and MINLOGE is 0.2, MAXLOGE is 2.6, and NUMQLEVELS is 64.

This codebook index, FGIDX, shall be used to determine a frame gain γ_q:

$$\gamma_q(m') = 10^{q\log(m',FGIDX)}; \quad 0 \le m' < 3 \qquad (4.15.8\text{-}2)$$

$$\gamma_q = 10^{MINLOGE + FGIDX \cdot \left(\frac{MAXLOGE - MINLOGE}{NUMQLEVELS} \right)} \qquad (4.15.8\text{-}3)$$

And the set of 3 subframe gains are given by

$$\gamma_q(m') = \begin{cases} \dfrac{2}{3}\gamma_q^{last} + \dfrac{1}{3}\gamma_q; & m' = 0 \\[2mm] \dfrac{1}{3}\gamma_q^{last} + \dfrac{2}{3}\gamma_q; & m' = 1 \\[2mm] \gamma_q; & m' = 2 \end{cases} \qquad (4.15.8\text{-}4)$$

4.15.9 Gain Quantization for SO 70 and SO 73 WB Mode

The gain shall be quantized to 5 bits using a scalar quantizer. The best gain index is found by minimizing the quantization error, as defined by:

$$FGIDX = \arg\min_i \left(\frac{1}{2}\log\left(1 + \frac{1}{4}\sum_{n=80}^{239}\left(s'(n)\right)^2 \right) - \left(MINLOGE_{wb} + i \cdot \left(\frac{MAXLOGE_{wb} - MINLOGE_{wb}}{NUMQLEVELS_{wb}} \right)\right)\right),$$

$$(4.15.9\text{-}1)$$

where $0 \le i < NUMQLEVELS_{wb}$, the indices of gain quantization, and $MINLOGE_{wb} = 2.0$, $MAXLOGE_{wb} = 12.0$, and $NUMQLEVELS_{wb} = 32$. The frame gain used for subsequent Eighth-Rate encoder processing is $\gamma_q = 0$.

4.15.10 Generation of Rate 1/8 Excitation

The excitation for each subframe at Rate 1/8 is generated by using a zero-mean, unit variance pseudo-Gaussian white noise sequence which is scaled by the quantized frame energy gain $\gamma_q(m')$ for each subframe m':

$$E(n) = \gamma_q ran\_g\{seed\}; \quad 0 \le n < L, \qquad (4.15.10\text{-}1)$$

where ran_g{*seed*} is the unit variance pseudo-random Gaussian white noise generator (see 4.16.2) and *seed* is a unique seed value defined at reset of the system.

4.15.11 Perceptual Weighting Filter Update

Although there is no closed-loop search at Rate 1/8, it is still necessary to update the memory of the perceptual weighting filter with the new excitation determined in 4.15.10 for each subframe. Update the weighted synthesis filter memory by filtering the excitation vector, $E(n)$, through the weighted synthesis filter, $H_{wq}(z)$, which is given in Equation 4.11.4-5.

4.16 Random Number Generation

Zero mean, unit variance gaussian pseudo-random numbers are obtained by generating uniform pseudo-random numbers and appropriately transforming them. The following two sub-sections describe the uniform pseudo-random number generator and the algorithm for effecting the uniform to gaussian transformation.

4.16.1 Uniform Pseudo-Random Number Generation Algorithm

The algorithm for generating the uniform pseudo-random numbers is initialized with a seed value, and produces a new seed value with each successive invocation, as well as producing the desired pseudo-random number. The state of the uniform random number generator is captured by the current value of the seed. Different modules making use of the random number generator should maintain their own seeds. Uniform pseudo-random numbers are generated as follows:

Inputs: The input to the uniform number generator is:

- The seed value, *seed0*

Outputs: The outputs of the uniform pseudo-random number generator are:

- The uniform pseudo-random number, *ran0*

- The modified seed value, *newseed0*

Processing: Uniform pseudo-random number generation is accomplished as described in the following pseudo-code:

$newseed0 = seed0 \oplus 23148373$

$temp = \text{trunc}(newseed0 \, / \, 127773)$

$newseed0 = 16807 * (newseed0 - temp * 127773) - 2836 * temp$

if($newseed0 < 0$)

$newseed0 = newseed0 + 2147483647$

$ran0 = newseed0 \, / \, 2147483647$

$newseed0 = seed0 \oplus 23148373$

4.16.2 Gaussian Pseudo-Random Number Generator

The gaussian pseudo-random number generator makes use of the uniform pseudo-random number generator described in 4.16.1. Gaussian pseudo-random numbers are generated in pairs. While the algorithm that performs the transformation from uniform to gaussian does not itself have any memory, the values produced by it are a function of the seed value, which is used in invocations of the uniform pseudo-random number generator described in 4.16.1. Consequently, it is important for each module that makes use of the random number generator to maintain its own seed value. The transformation from uniform to gaussian is described as follows:

Inputs: The input to the gaussian pseudo-random number generator is

- The seed value, *seed*

Outputs: The outputs of the gaussian pseudo-random number generator are:

- The pair of gaussian pseudo-random numbers generated, *ran_g0* and *ran_g1*

- The new value of the seed, *newseed*.

Processing: Generation of gaussian pseudo-random numbers is described in the following pseudocode. Note that *ran_u0* and *ran_u1* are the uniform pseudo-random numbers produced by two different invocations of the algorithm of 4.16.1, and *newseed0* and *newseed1* are the new seed values produced by each of these respective invocations. Each invocation of the uniform pseudo-random number generator uses the current value of *seed* as its input.

do {

    $v1 = 2.0 * ran\_u0 - 1.0$

    $seed = newseed0$

    $v2 = 2.0 * ran\_u1 - 1.0$

    $seed = newseed1$

    $rsq = v1 * v1 + v2 * v2$

} while ($rsq >= 1.0$ OR $rsq == 0.0$)

$fac = (-2.0 * \log(rsq) / rsq)^{1/2}$

$ran\_g0 = v1 * fac$

$ran\_g1 = v2 * fac$

4.17 Discontinuous Transmission (DTX)

DTX implementation of EVRC-A follows the requirements in [N2].

EVRC-B and, EVRC-WB, and EVRC-NW have a native Discontinous Transmission or DTX scheme. When DTX is enabled, the encoder doesn't not transmit Rate 1/8 frame every 20 ms during VAD = 0, but only sends Eighth-Rate frame infrequently. At the end of a talk spurt (defined by the use of a Rate 1 or Rate 1/2 or Rate 1/4 frame), the leading first "Eighth-Rate Hangover" [N2] Rate 1/8 frames are transmitted. SubsequentlyIn case of EVRC-B and EVRC-WB, all following Rate 1/8 frames are blanked or suppressed and not transmitted. In case of EVRC-NW, such Rate 1/8 frames are classified as Rate 1/8 Non Critical and may or may not be blanked by multiplex sublayer. If the multiplex sublayer decides to transmit all of the Rate 1/8 Non Critical packets, then the system is equivalent to operate in non-DTX mode. Otherwise, if the multiplex sublayer suppresses some or all the the Rate 1/8 Non Critical packets, then the system operates in DTX mode. If there is an appreciable change in the Rate 1/8 frame indicating a significant change in the background noise characteristic, another Rate 1/8 frame is sent and this frame type is called Silence Insertion Description (SID) [N2]. This change is calculated by comparing gain of the Eighth-Rate frame with a running average of Eighth-Rate frame energies. If the change is larger than a threshold, then an Eighth-Rate frame is transmitted. Similarly, the LSPs of the Eighth-Rate frame are compared with the LSPs of the last transmitted Eighth-Rate frame. If the change is greater than a threshold an Eighth-Rate frame is transmitted, otherwise an Eighth-Rate frame is transmitted according to the "DTX min" and "DTX max" parameters in [N2]. In addition to the above mentioned adaptive decision to transmit an Eighth-Rate frame, the DTX implementation of EVRC-B and EVRC-WB also follows the requirements in [N2]

including the default values. When a decision is made to transmit an Eighth-Rate frame, a smoothed prototype of the Eighth-Rate frame is transmitted rather than the Rate 1/8 frame computed according to Section 4.15. The gains and the LSPs are statistically smoothed in that the smoothed prototype contains the gain and LSPs that have been most often chosen in the recent past.

EVRC-NW always operates in DTX mode, with the same DTX mechanism as EVRC-B and EVRC-WB. For circuit switched call, however, the Rate 1/8 frames between two valid speech frames, which include Rate 1, Rate 1/2, Rate 1/4, Eighth Rate Hangover, and SID, are classified as Rate 1/8 Non-Critical and are presented to multiplex sublayer, and may or may not be blanked or suppressed by multiplex sublayer. The "Rate 1/8" packet type in Table 2.1.1-1 denotes either Eighth Rate Hangover or SID. A Rate 1/8 packet (16 bits) itself has no indication if this packet is Rate 1/8 Non-Critical or not.

4.18 Coding the High-Frequency Band of Wideband Speech

4.18.1 Declick Algorithm

The basic idea of the click-removing algorithm is to detect energy bursts that exist in the high band but not in the low band, and then briefly reduce the level of the high-band signal during each of these energy bursts.

Figure 4.18.1-1 shows the algorithm. It operates on frames of 20 milliseconds, and the low-band is sampled at 8 kHz and the high-band is sampled at 7 kHz. The low-band input is filtered with a shaping filter of the form

$$F_{LB}(z) = \frac{1 + 0.96z^{-1} + 0.96z^{-2} + z^{-3}}{1 - 0.5z^{-1}},$$

which removes very low and high frequencies from the low-band signal. The high band is filtered with a shaping filter of the form

$$F_{HB}(z) = \frac{0.5 + z^{-1} + 0.5z^{-2}}{1 + 0.5z^{-1} + 0.3z^{-2}},$$

which removes frequencies around 4 kHz.
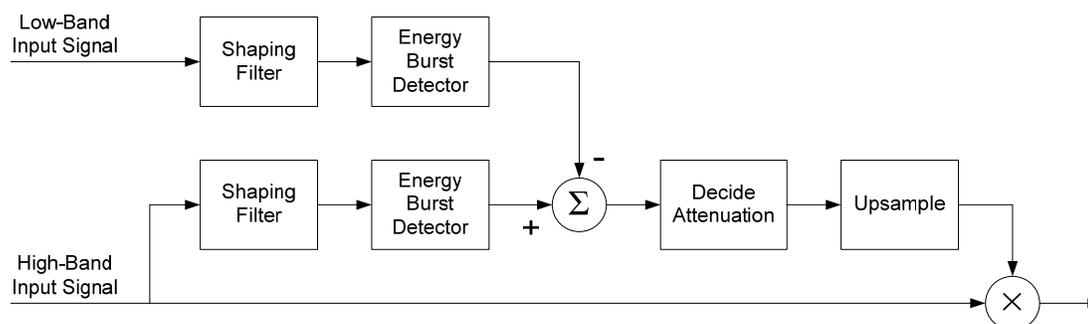
Figure 4.18.1-1 Pre-processing of the High Band to Remove Clicks

In the next step, the shaped low band and high band are both processed by a energy burst detector, depicted in Figure 4.18.1-2. The energy burst detector combines a forward-moving and a backward-moving edge detector. The forward-moving edge detector first computes the square of each sample to obtain the energy signal. Then a smoothing operation is performed according to

$$S(n) = \begin{cases} P(n), & \text{if } P(n) \geq S(n-1) \\ \alpha S(n-1) + (1-\alpha)P(n), & \text{if } P(n) < S(n-1), \end{cases}$$

where $P(n)$ is the energy signal, $S(n)$ is the smoothed energy signal, and α is a decay factor such that the decay time of the smoother is about 5 milliseconds. The smoothed energy signal is subsequently down-sampled to a 1 kHz sampling rate, and converted to a logarithmic scale. The resulting signal is reduced by a delayed version of itself, where the delay is about 4 milliseconds, to obtain the output of the forward-moving edge detector. The same procedure is followed to find the backward-moving edge detector signal, with the difference that all operations are reversed in time: processing starts a few milliseconds beyond the end of the current frame and progresses to the beginning of the current frame. Finally, we combine the two edge detectors by clipping the negative values ($out = \max(in, 0)$) and taking the mean of these clipped signals. The arithmetic mean ($\frac{1}{2}(a+b)$) is used by the energy burst detector operating on the low band, whereas the high band energy burst detector uses the more conservative geometric mean ($\sqrt{a \cdot b}$). The output signal indicates the time location and strength of energy bursts in the input signal, on a logarithmic scale, and is always equal to or larger than zero.

Next, the outputs of the low-band and high-band energy burst detectors are combined by subtracting the former from the latter to produce a signal that indicates where energy burst exist in the high band that do not occur (or are weaker) in the low band. This signal is used to decide how much to attenuate the high band using the formula

$$A_{dB} = \begin{cases} 0, & \text{if } D_{dB} < T_{dB} \\ 20\left(1 - \dfrac{2}{1 + \exp(D_{dB}/10)}\right), & \text{if } D_{dB} > T_{dB}, \end{cases}$$

where D_{dB} is the difference between high-band and low-band energy burst detectors, T_{dB} is a threshold above which energy bursts will be attenuated (we use 8 dB as the threshold), and A_{dB} is the attenuation that will be applied to the high-band signal. The attenuation signal, being derived from the energy burst detector outputs, is sampled at 1 kHz, and is up-sampled to a sampling rate equal to that of the high band (7 kHz) using windowed sinc functions. Finally, we multiply the high-band signal by the up-sampled attenuation signal to obtain the click-removed high-band signal.
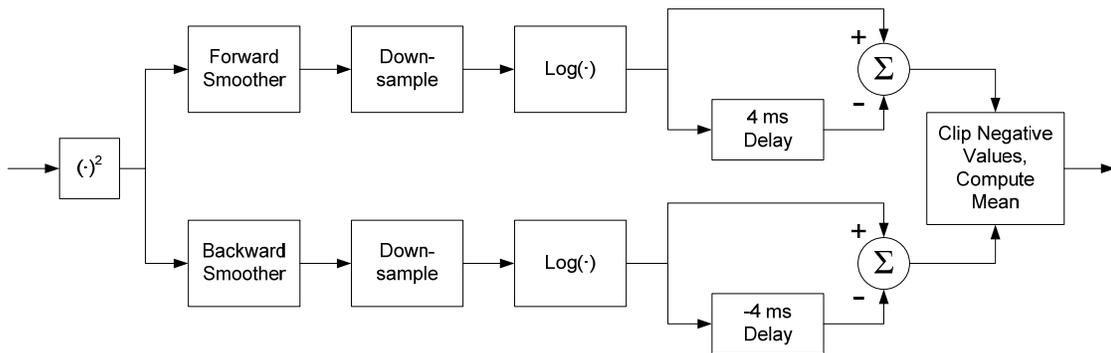
Figure 4.18.1-2 Energy Burst Detector

4.18.2 High-Band LPC Analysis and Quantization

Six spectral parameters are obtained by a traditional linear prediction (LPC) analysis. For efficient coding, these prediction coefficients are converted to line spectral frequencies (LSFs) and quantized using a vector quantizer using 8 bits.

The LSF quantizer is improved by incorporating temporal noise shaping. For each frame, the LSF quantization error vector is computed and multiplied by a scale factor less than 1.0. The following frame, this scaled quantization error is added to the LSF vector before quantization. The scale factor is adjusted dynamically depending on the amount of fluctuations already present in the unquantized LSF vectors: when the difference between the current and previous LSF vectors is large, the scale factor is close to zero and almost no noise shaping results. When the current LSF vector differs little from the previous one, the scale factor is close to 1.0. The resulting LSF quantization minimizes spectral distortion when the speech signal is changing, and minimizes spectral fluctuations when the speech signal is relatively constant from one frame to the next.

4.18.3 High-Band Excitation Generation

The high-band excitation signal is derived from the low-band excitation in the form of the excitation signal.
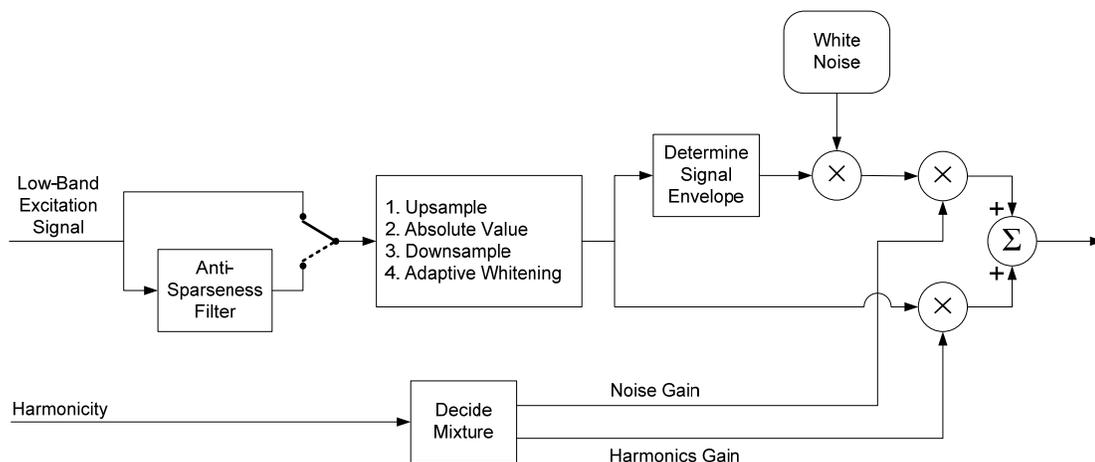
Figure 4.18.3-1 Generation of the High-Band Excitation Signal

Figure 4.18.3-1 depicts the process that generates the high-band excitation from the narrowband excitation signal. The excitation signal is first run through an all-pass filter of the form

$$H(z) = \frac{-0.7 + z^{-4}}{1 - 0.7z^{-4}} \cdot \frac{0.6 + z^{-6}}{1 + 0.6z^{-6}}.$$

The decision to use the output of the filter or the original input signal is derived from the quantized narrowband adaptive codebook gain and first reflection coefficient: unvoiced signals are characterized by a low adaptive codebook gain and a first reflection coefficient that is close to zero or positive (indicating a flat or upwards-tilted spectrum).

Next is the nonlinear function. This module up-samples the signal to 16 kHz before applying the absolute value non-linear function. From here, a 8 kHz-sampled signal is produced using the same high-band analysis filter that was used to

split the input signal in a low and a high band in Section 4.5. The result is spectrally flattened with an adaptive 4th order linear prediction filter, to create the harmonically-extended excitation signal.

A modulated noise signal is generated by multiplying a unit-variance white noise signal with the envelope of the harmonically-extended excitation signal. This envelope is obtained by taking the squared value of each sample, smoothing with a first order IIR low-pass filter and taking the square-root of each smoothed sample.

The modulated noise and harmonically-extended excitation signals are now mixed together to create a signal with the right amount of harmonic and noise contents. The gains of the noise and harmonics parts are set adaptively to match the ratio of harmonic to noise energy in the high band of the original speech signal. Typically, the noise gain is between 0.1 and 0.7, and the harmonics gain is derived from the noise gain as $G_{harm} = \sqrt{1 - G_{noise}^2}$, so that the energy of the output signal is not affected by changing the gains.

Note that the high-band excitation is generated in both encoder and decoder. The encoder determines the side information for the gain factors by comparing the synthesized high band with the original high band, and the decoder then uses these gains to properly scale the synthesized high band. For the decoding process to be consistent with the encoding process, it is important that identical high-band excitations be generated in encoder and decoder. This is achieved by (1) using the *quantized* narrowband excitation, which is available to both encoder and decoder, and (2) having the same state for the random white noise generator in encoder and decoder at all times.

4.18.4 Computation and Quantization of Gains for High-Band Excitation

Five temporal gain parameters each represent the scale factor of a 4 millisecond subframe. The gain parameters are found by comparing the energies in each subframe of the input high-band signal with the unscaled, synthesized high-band excitation signal. For each frame, a frame-level gain is first computed. This gain is then used to normalize the 5 subframe gains. The subframe gain parameters are also quantized with a vector quantizer, and the quantization index is transmitted to the decoder.

For voiced speech, frame-level gain is quantized using 4 bits look-up table quantizer. The subframe gains are vector quantized using 4 bits.

For unvoiced speech, frame-level gain is quantized using 7 bit scalar quantizer. The subframe gains are vector quantized using a multistage VQ with 4 bits per stage.

4.19 Packet Formatting for SO 3

After encoding, the encoded speech packets shall be formatted as described in Table 4.19-1. For each parameter, bit index 0 corresponds to the most significant bit. The packet buffer bit index 1 indicates the bit position corresponding to the beginning of the packet buffer. Parameter notations are defined as follows:

- *LPCFLAG*: spectral change indicator,

- *LSPIDX(k)[j]*: the *j*-th bit of the *k*-th LSP codebook for the entire frame, (4 codebooks for Rate 1, 3 for Rate 1/2, and 2 for Rate 1/8),

- *DELAY[j]*: the *j*-th bit of the pitch delay estimate for the entire frame,

- *DDELAY[j]*: the *j*-th bit of the delay difference for the entire frame,

- *ACBGIDX(m′)[j]*: the *j*-th bit of the adaptive codebook gain index for the *m′*-th subframe,

1 ● *FCBSIDX*(*m′,k*)[*j*]: the *j*-th bit of the *k*-th fixed codebook shape index for the *m′*-th subframe, (4 codebooks for Rate
2 1, 1 for Rate 1/2),

3 ● *FCBGIDX*(*m′*)[*j*]: the j-th bit of the fixed codebook gain index for the *m′*-th subframe,

4 ● *FGIDX*[*j*]: the *j*-th bit of the frame energy gain.

5 In the case where a Rate 1/8 packet with all bits set to '1' (null Traffic Channel data) is generated, *FGIDX*[0] is set to
6 zero.

7

8 **Table 4.19-1 Packet Formats**

Bit index	Rate 1 packet bits	Rate 1/2 packet bits	Rate 1/8 packet bits
1	*LPCFLAG*	*LSPIDX*(1)[0]	*LSPIDX*(1)[0]
2	*LSPIDX*(1)[0]	*LSPIDX*(1)[1]	*LSPIDX*(1)[1]
3	*LSPIDX*(1)[1]	*LSPIDX*(1)[2]	*LSPIDX*(1)[2]
4	*LSPIDX*(1)[2]	*LSPIDX*(1)[3]	*LSPIDX*(1)[3]
5	*LSPIDX*(1)[3]	*LSPIDX*(1)[4]	*LSPIDX*(2)[0]
6	*LSPIDX*(1)[4]	*LSPIDX*(1)[5]	*LSPIDX*(2)[1]
7	*LSPIDX*(1)[5]	*LSPIDX*(1)[6]	*LSPIDX*(2)[2]
8	*LSPIDX*(2)[0]	*LSPIDX*(2)[0]	*LSPIDX*(2)[3]
9	*LSPIDX*(2)[1]	*LSPIDX*(2)[1]	*FGIDX*[0]
10	*LSPIDX*(2)[2]	*LSPIDX*(2)[2]	*FGIDX*[1]
11	*LSPIDX*(2)[3]	*LSPIDX*(2)[3]	*FGIDX*[2]
12	*LSPIDX*(2)[4]	*LSPIDX*(2)[4]	*FGIDX*[3]
13	*LSPIDX*(2)[5]	*LSPIDX*(2)[5]	*FGIDX*[4]
14	*LSPIDX*(3)[0]	*LSPIDX*(2)[6]	*FGIDX*[5]
15	*LSPIDX*(3)[1]	*LSPIDX*(3)[0]	*FGIDX*[6]
16	*LSPIDX*(3)[2]	*LSPIDX*(3)[1]	*FGIDX*[7]
17	*LSPIDX*(3)[3]	*LSPIDX*(3)[2]	
18	*LSPIDX*(3)[4]	*LSPIDX*(3)[3]	
19	*LSPIDX*(3)[5]	*LSPIDX*(3)[4]	
20	*LSPIDX*(3)[6]	*LSPIDX*(3)[5]	
21	*LSPIDX*(3)[7]	*LSPIDX*(3)[6]	
22	*LSPIDX*(3)[8]	*LSPIDX*(3)[7]	
23	*LSPIDX*(4)[0]	*DELAY*[0]	
24	*LSPIDX*(4)[1]	*DELAY*[1]	
25	*LSPIDX*(4)[2]	*DELAY*[2]	
26	*LSPIDX*(4)[3]	*DELAY*[3]	
27	*LSPIDX*(4)[4]	*DELAY*[4]	
28	*LSPIDX*(4)[5]	*DELAY*[5]	
29	*LSPIDX*(4)[6]	*DELAY*[6]	
30	*DELAY*[0]	*ACBGIDX*(0)[0]	
31	*DELAY*[1]	*ACBGIDX*(0)[1]	
32	*DELAY*[2]	*ACBGIDX*(0)[2]	

Bit index	Rate 1 packet bits	Rate 1/2 packet bits	Rate 1/8 packet bits
33	*DELAY*[3]	*FCBSIDX*(0,0)[0]	
34	*DELAY*[4]	*FCBSIDX*(0,0)[1]	
35	*DELAY*[5]	*FCBSIDX*(0,0)[2]	
36	*DELAY*[6]	*FCBSIDX*(0,0)[3]	
37	*DDELAY*[0]	*FCBSIDX*(0,0)[4]	
38	*DDELAY*[1]	*FCBSIDX*(0,0)[5]	
39	*DDELAY*[2]	*FCBSIDX*(0,0)[6]	
40	*DDELAY*[3]	*FCBSIDX*(0,0)[7]	
41	*DDELAY*[4]	*FCBSIDX*(0,0)[8]	
42	*ACBGIDX*(0)[0]	*FCBSIDX*(0,0)[9]	
43	*ACBGIDX*(0)[1]	*FCBGIDX*(0)[0]	
44	*ACBGIDX*(0)[2]	*FCBGIDX*(0)[1]	
45	*FCBSIDX*(0,0)[0]	*FCBGIDX*(0)[2]	
46	*FCBSIDX*(0,0)[1]	*FCBGIDX*(0)[3]	
47	*FCBSIDX*(0,0)[2]	*ACBGIDX*(1)[0]	
48	*FCBSIDX*(0,0)[3]	*ACBGIDX*(1)[1]	
49	*FCBSIDX*(0,0)[4]	*ACBGIDX*(1)[2]	
50	*FCBSIDX*(0,0)[5]	*FCBSIDX*(1,0)[0]	
51	*FCBSIDX*(0,0)[6]	*FCBSIDX*(1,0)[1]	
52	*FCBSIDX*(0,0)[7]	*FCBSIDX*(1,0)[2]	
53	*FCBSIDX*(0,1)[0]	*FCBSIDX*(1,0)[3]	
54	*FCBSIDX*(0,1)[1]	*FCBSIDX*(1,0)[4]	
55	*FCBSIDX*(0,1)[2]	*FCBSIDX*(1,0)[5]	
56	*FCBSIDX*(0,1)[3]	*FCBSIDX*(1,0)[6]	
57	*FCBSIDX*(0,1)[4]	*FCBSIDX*(1,0)[7]	
58	*FCBSIDX*(0,1)[5]	*FCBSIDX*(1,0)[8]	
59	*FCBSIDX*(0,1)[6]	*FCBSIDX*(1,0)[9]	
60	*FCBSIDX*(0,1)[7]	*FCBGIDX*(1)[0]	
61	*FCBSIDX*(0,2)[0]	*FCBGIDX*(1)[1]	
62	*FCBSIDX*(0,2)[1]	*FCBGIDX*(1)[2]	
63	*FCBSIDX*(0,2)[2]	*FCBGIDX*(1)[3]	
64	*FCBSIDX*(0,2)[3]	*ACBGIDX*(2)[0]	
65	*FCBSIDX*(0,2)[4]	*ACBGIDX*(2)[1]	
66	*FCBSIDX*(0,2)[5]	*ACBGIDX*(2)[2]	
67	*FCBSIDX*(0,2)[6]	*FCBSIDX*(2,0)[0]	
68	*FCBSIDX*(0,2)[7]	*FCBSIDX*(2,0)[1]	
69	*FCBSIDX*(0,3)[0]	*FCBSIDX*(2,0)[2]	
70	*FCBSIDX*(0,3)[1]	*FCBSIDX*(2,0)[3]	
71	*FCBSIDX*(0,3)[2]	*FCBSIDX*(2,0)[4]	
72	*FCBSIDX*(0,3)[3]	*FCBSIDX*(2,0)[5]	
73	*FCBSIDX*(0,3)[4]	*FCBSIDX*(2,0)[6]	
74	*FCBSIDX*(0,3)[5]	*FCBSIDX*(2,0)[7]	

Bit index	Rate 1 packet bits	Rate 1/2 packet bits	Rate 1/8 packet bits
75	*FCBSIDX*(0,3)[6]	*FCBSIDX*(2,0)[8]	
76	*FCBSIDX*(0,3)[7]	*FCBSIDX*(2,0)[9]	
77	*FCBSIDX*(0,3)[8]	*FCBGIDX*(2)[0]	
78	*FCBSIDX*(0,3)[9]	*FCBGIDX*(2)[1]	
79	*FCBSIDX*(0,3)[10]	*FCBGIDX*(2)[2]	
80	*FCBGIDX*(0)[0]	*FCBGIDX*(2)[3]	
81	*FCBGIDX*(0)[1]		
82	*FCBGIDX*(0)[2]		
83	*FCBGIDX*(0)[3]		
84	*FCBGIDX*(0)[4]		
85	*ACBGIDX*(1)[0]		
86	*ACBGIDX*(1)[1]		
87	*ACBGIDX*(1)[2]		
88	*FCBSIDX*(1,0)[0]		
89	*FCBSIDX*(1,0)[1]		
90	*FCBSIDX*(1,0)[2]		
91	*FCBSIDX*(1,0)[3]		
92	*FCBSIDX*(1,0)[4]		
93	*FCBSIDX*(1,0)[5]		
94	*FCBSIDX*(1,0)[6]		
95	*FCBSIDX*(1,0)[7]		
96	*FCBSIDX*(1,1)[0]		
97	*FCBSIDX*(1,1)[1]		
98	*FCBSIDX*(1,1)[2]		
99	*FCBSIDX*(1,1)[3]		
100	*FCBSIDX*(1,1)[4]		
101	*FCBSIDX*(1,1)[5]		
102	*FCBSIDX*(1,1)[6]		
103	*FCBSIDX*(1,1)[7]		
104	*FCBSIDX*(1,2)[0]		
105	*FCBSIDX*(1,2)[1]		
106	*FCBSIDX*(1,2)[2]		
107	*FCBSIDX*(1,2)[3]		
108	*FCBSIDX*(1,2)[4]		
109	*FCBSIDX*(1,2)[5]		
110	*FCBSIDX*(1,2)[6]		
111	*FCBSIDX*(1,2)[7]		
112	*FCBSIDX*(1,3)[0]		
113	*FCBSIDX*(1,3)[1]		
114	*FCBSIDX*(1,3)[2]		
115	*FCBSIDX*(1,3)[3]		
116	*FCBSIDX*(1,3)[4]		

3GPP2 C.S0014-D v3.0

Bit index	Rate 1 packet bits	Rate 1/2 packet bits	Rate 1/8 packet bits
117	FCBSIDX(1,3)[5]		
118	FCBSIDX(1,3)[6]		
119	FCBSIDX(1,3)[7]		
120	FCBSIDX(1,3)[8]		
121	FCBSIDX(1,3)[9]		
122	FCBSIDX(1,3)[10]		
123	FCBGIDX(1)[0]		
124	FCBGIDX(1)[1]		
125	FCBGIDX(1)[2]		
126	FCBGIDX(1)[3]		
127	FCBGIDX(1)[4]		
128	ACBGIDX(2)[0]		
129	ACBGIDX(2)[1]		
130	ACBGIDX(2)[2]		
131	FCBSIDX(2,0)[0]		
132	FCBSIDX(2,0)[1]		
133	FCBSIDX(2,0)[2]		
134	FCBSIDX(2,0)[3]		
135	FCBSIDX(2,0)[4]		
136	FCBSIDX(2,0)[5]		
137	FCBSIDX(2,0)[6]		
138	FCBSIDX(2,0)[7]		
139	FCBSIDX(2,1)[0]		
140	FCBSIDX(2,1)[1]		
141	FCBSIDX(2,1)[2]		
142	FCBSIDX(2,1)[3]		
143	FCBSIDX(2,1)[4]		
144	FCBSIDX(2,1)[5]		
145	FCBSIDX(2,1)[6]		
146	FCBSIDX(2,1)[7]		
147	FCBSIDX(2,2)[0]		
148	FCBSIDX(2,2)[1]		
149	FCBSIDX(2,2)[2]		
150	FCBSIDX(2,2)[3]		
151	FCBSIDX(2,2)[4]		
152	FCBSIDX(2,2)[5]		
153	FCBSIDX(2,2)[6]		
154	FCBSIDX(2,2)[7]		
155	FCBSIDX(2,3)[0]		
156	FCBSIDX(2,3)[1]		
157	FCBSIDX(2,3)[2]		
158	FCBSIDX(2,3)[3]		

Bit index	Rate 1 packet bits	Rate 1/2 packet bits	Rate 1/8 packet bits
159	*FCBSIDX*(2,3)[4]		
160	*FCBSIDX*(2,3)[5]		
161	*FCBSIDX*(2,3)[6]		
162	*FCBSIDX*(2,3)[7]		
163	*FCBSIDX*(2,3)[8]		
164	*FCBSIDX*(2,3)[9]		
165	*FCBSIDX*(2,3)[10]		
166	*FCBGIDX*(2)[0]		
167	*FCBGIDX*(2)[1]		
168	*FCBGIDX*(2)[2]		
169	*FCBGIDX*(2)[3]		
170	*FCBGIDX*(2)[4]		
171	TTY Baud Rate Bit		

4.20 Packet Formatting for SO 68

The parameters are packed MSB first. That is, in the notation, Ps-Pe used in "Bit index in Packet" column, Ps referes to the MSB of the parameter, and Pe refers to the LSB of the parameter.

Table 4.20-1 Packet Formatting for Full-Rate CELP

Parameter (value)	Number of bits	Bit index in packet
ModeBit (0)	1	P0-P0
DelayIndex	7	P1-P7
LSP_IDX0	6	P8-P13
LSP_IDX1	6	P14-P19
LSP_IDX2	9	P20-P28
LSP_IDX3	7	P29-P35
DDELAY_IDX	5	P36-P40
ACBG_IDX_SF0	3	P41-P43
ACBG_IDX_SF1	3	P44-P46
ACBG_IDX_SF2	3	P47-P49
FCB_PULSE_IDX0_SF0	16	P50-P65
FCB_PULSE_IDX1_SF0	16	P66-P81
FCB_PULSE_IDX2_SF0	3	P82-P84
FCB_PULSE_IDX0_SF1	16	P85-P100
FCB_PULSE_IDX1_SF1	16	P101-P116
FCB_PULSE_IDX2_SF1	3	P117-P119
FCB_PULSE_IDX0_SF2	16	P120-P135
FCB_PULSE_IDX1_SF2	16	P136-P151
FCB_PULSE_IDX2_SF2	3	P152-P154
FCBG_IDX_SF0	5	P155-P159
FCBG_IDX_SF1	5	P160-P164

Parameter (value)	Number of bits	Bit index in packet
FCBG_IDX_SF2	5	P165-P169
RESERVED (0)	1	P170

1

2

<p align="center">**Table 4.20-2 Packet Formatting for Full-Rate PPP**</p>

Parameter (value)	Number of bits	Bit index in packet
ModeBit (1)	1	P0-P0
DelayIndex	7	P1-P7
LSP_IDX0	6	P8-P13
LSP_IDX1	6	P14-P19
LSP_IDX2	9	P20-P28
LSP_IDX3	7	P29-P35
ROTATION_IDX	7	P36-P42
AMP_IDX0	6	P43-P48
AMP_IDX1	6	P49-P54
AMP_IDX2	8	P55-62
POWER_IDX	8	P63-70
ALIGN_IDX0	5	P71-P75
ALIGN_IDX1	5	P76-P80
ALIGN_IDX2	5	P81-P85
ALIGN_IDX3	6	P86-P91
ALIGN_IDX4	6	P92-P97
ALIGN_IDX5	6	P98-P103
ALIGN_IDX6	6	P104-P109
ALIGN_IDX7	6	P110-P115
ALIGN_IDX8	6	P116-P121
ALIGN_IDX9	6	P122-P127
ALIGN_IDX10	6	P128-P133
ALIGN_IDX11	6	P134-P139
ALIGN_IDX12	6	P140-P145
ALIGN_IDX13	6	P146-P151
ALIGN_IDX14	6	P152-P157
ALIGN_IDX15	6	P158-P163
ALIGN_IDX16	6	P164-P169
RESERVED (0)	1	P170

3

4

<p align="center">**Table 4.20-3 Packet formatting for Half-Rate CELP**</p>

Parameter (value)	Number of Bits	Bit index in packet
DelayIndex	7	P0-P6

Parameter (value)	Number of Bits	Bit index in packet
LSP_IDX0	7	P7-P13
LSP_IDX1	7	P14-P20
LSP_IDX2	8	P21-P28
ACBG_IDX_SF0	3	P29-P31
ACBG_IDX_SF1	3	P32-P34
ACBG_IDX_SF2	3	P35-P37
FCB_PULSE_IDX0_SF0	10	P38-P47
FCB_PULSE_IDX0_SF1	10	P48-P57
FCB_PULSE_IDX0_SF2	10	P58-P67
FCBG_IDX_SF0	4	P68-P71
FCBG_IDX_SF1	4	P72-P75
FCBG_IDX_SF2	4	P76-P79

1

2

Table 4.20-4 Packet Formatting for Special Half-Rate CELP

Parameter (value)	Number of bits	Bit index in packet
SpecialHalfCelpFlag0(0x6E)	7	P0-P6
SpecialHalfCelpFlag1(0x00)	2	P7-P8
ModeBit (0)	1	P9-P9
LSP_IDX0	6	P10-P15
LSP_IDX1	6	P16-P21
LSP_IDX2	9	P22-P30
LSP_IDX3	7	P31-P37
ACBG_IDX_SF0	3	P38-P40
ACBG_IDX_SF1	3	P41-P43
ACBG_IDX_SF2	3	P44-P46
DelayIndex	7	P47-P53
RESERVED (all 0s)	26	P54-P79

3

4

Table 4.20-5 Packet Formatting for Special Half-Rate PPP

Parameter (value)	Number of bits	Bit index in packet
SpecialHalfPPPFlag0(0x6E)	7	P0-P6
SpecialHalfPPPFlag1(0x00)	2	P7-P8
ModeBit (1)	1	P9-P9
LSP_IDX0	6	P10-P15
LSP_IDX1	6	P16-P21
LSP_IDX2	9	P22-P30
LSP_IDX3	7	P31-P37
ROTATION_IDX	7	P38-P44
AMP_IDX0	6	P45-P50

3GPP2 C.S0014-D v3.0

Parameter (value)	Number of bits	Bit index in packet
AMP_IDX1	6	P51-P56
AMP_IDX2	8	P57-64
POWER_IDX	8	P65-72
DelayIndex	7	P73-P79

1

2

Table 4.20-6 Packet Formatting for Special Half-Rate NELP

Parameter (value)	Number of bits	Bit index in packet
SpecialHalfNelpFlag0(0x6E)	7	P0-P6
SpecialHalfNelpFlag1(0x01)	2	P7-P8
LSP_IDX0	8	P9-P16
LSP_IDX1	8	P17-P24
NELP_GAIN_IDX0	5	P25-P29
NELP_GAIN_IDX1	6	P30-P35
NELP_GAIN_IDX2	6	P36-P41
NELP_FILTER_ID	2	P42-P43
RESERVED (all 0s)	36	P44-P79

3

4

Table 4.20-7 Packet Formatting for Quarter-Rate NELP

Parameter (value)	Number of bits	Bit index in packet
ModeBit (0)	1	P0-P0
LSP_IDX0	8	P1-P8
LSP_IDX1	8	P9-P16
NELP_GAIN_IDX0	5	P17-P21
NELP_GAIN_IDX1	6	P22-P27
NELP_GAIN_IDX2	6	P28-P33
NELP_FILTER_ID	2	P34-P35
RESERVED (0)	4	P36-P39

5

6

Table 4.20-8 Packet Formatting for Quarter-Rate PPP

Parameter (value)	Number of bits	Bit index in packet
ModeBit (1)	1	P0-P0
LSP_IDX0	10	P1-P10
LSP_IDX1	6	P11-P16
DELTA LAG IDX	4	P17-P20
AMP_IDX0	6	P21-P26
AMP_IDX1	6	P27-P32
POWER_IDX	6	P33-P38
RESERVED (0)	1	P39-P39

Table 4.20-9 Packet Formatting for Eighth-Rate

Parameter (value)	Number of bits	Bit index in packet
LSP_IDX0	4	P0-P3
LSP_IDX1	4	P4-P7
GAIN IDX	6	P8-P13
RESERVED0 (0)	1	P14-P14
RESERVED1 (0)	1	P15-P15

The ModeBit in Table 4.20-1, Table 4.20-2, Table 4.20-4, Table 4.20-5, Table 4.20-7, and Table 4.20-8, indicates if a packet is a PPP packet or a CELP/NELP packet. If a packet is a PPP packet, then ModeBit is set to '1'. Otherwise, it is set to '0'.

4.20.1 Special Half NELP Frame

This frame format carries the NELP frame in a Half-Rate packet. This packet is used in Half-Rate max mode.

4.20.2 Packet Level Signaling Interworking Function

Table 4.20-4 and Table 4.20-5 show special Half-Rate CELP and special Half-Rate PPP frames formats.

A packet level signaling interworking function that needs to dim a Full-Rate packet CELP packet to a special Half-Rate CELP packet should receive a packet in the frame in the format described in Table 4.20-1 and convert it to the format described in Table 4.20-4 by relocating some of the parameters (those that are common between Table 4.20-1 and Table 4.20-4) and setting the SpecialHalfRateCELP flag.

A packet level signaling interworking function that needs to dim a Full-Rate packet PPP packet to a special Half-Rate PPP packet should receive a packet in the frame in the format described in Table 4.20-2 and convert it to the format described in Table 4.20-5 by relocating some of the parameters (those that are common between Table 4.20-2 and Table 4.20-5) and setting the SpecialHalfRateCELP flag.

4.21 Packet Formatting for SO 70

The parameters are packed MSB first. That is, in the notation, Ps-Pe used in "Bit index in Packet" column, Ps referes to the MSB of the parameter, and Pe refers to the LSB of the parameter.

Table 4.21-1 Packet Formatting for Full-Rate CELP in SO 68 Interoperable Mode

Parameter (value)	Number of bits	Bit index in packet
ModeBit (0)	1	P0-P0
DelayIndex	7	P1-P7
LSP_IDX0	6	P8-P13
LSP_IDX1	6	P14-P19
LSP_IDX2	9	P20-P28

Parameter (value)	Number of bits	Bit index in packet
LSP_IDX3	7	P29-P35
DDELAY_IDX	5	P36-P40
ACBG_IDX_SF0	3	P41-P43
ACBG_IDX_SF1	3	P44-P46
ACBG_IDX_SF2	3	P47-P49
FCB_PULSE_IDX0_SF0	16	P50-P65
FCB_PULSE_IDX1_SF0	16	P66-P81
FCB_PULSE_IDX2_SF0	3	P82-P84
FCB_PULSE_IDX0_SF1	16	P85-P100
FCB_PULSE_IDX1_SF1	16	P101-P116
FCB_PULSE_IDX2_SF1	3	P117-P119
FCB_PULSE_IDX0_SF2	16	P120-P135
FCB_PULSE_IDX1_SF2	16	P136-P151
FCB_PULSE_IDX2_SF2	3	P152-P154
FCBG_IDX_SF0	5	P155-P159
FCBG_IDX_SF1	5	P160-P164
FCBG_IDX_SF2	5	P165-P169
EVRCB_IO_FLAG (0)	1	P170

1

2

Table 4.21-2 Packet Formatting for Full-Rate Wideband CELP Frame

Parameter (value)	Number of bits	Bit index in packet
DelayIndex	7	P0-P6
LSP_IDX0	6	P7-P12
LSP_IDX1	6	P13-P18
LSP_IDX2	9	P19-P27
LSP_IDX3	7	P28-P34
ACBG_IDX_SF0	3	P35-P37
ACBG_IDX_SF1	3	P38-P40
ACBG_IDX_SF2	3	P41-P43
FCB_PULSE_IDX_DELAY_ADJUST	94	P44-P137
FCBG_IDX_SF0	5	P138-P142
FCBG_IDX_SF1	5	P143-P147
FCBG_IDX_SF2	5	P148-P152
HIGHBAND_LSP_IDX	8	P153-P160
HIGHBAND_GAIN_SHAPE_IDX	4	P161-P164
HIGHBAND_GAIN_FRAME_IDX	4	P165-P168
MODEBIT (CELP/ MDCT) (0)	1	P169
EVRCB_IO_FLAG (1)	1	P170

3

Table 4.21-3 Packet Formatting for Full-Rate Wideband MDCT Frame

Parameter (value)	Number of bits	Bit index in packet
LSP_IDX	28	P0-P27
MDCT_INDEX	114	P28-P141
FRAME_GAIN_INDEX	7	P142-P148
NOISE_GAIN_INDEX	2	P149-P150
HIGHBAND_LSP_IDX	8	P151-P158
HIGHBAND_GAIN_SHAPE_IDX	4	P159-P162
HIGHBAND_GAIN_FRAME_IDX	4	P163-P166
RESERVED	1	167
NB_WB_MODE_BIT (1)	1	P168
MODEBIT (CELP/ MDCT/) (1)	1	P169
EVRCB_IO_FLAG (1)	1	P170

Table 4.21-4 Packet Formatting for Full-Rate Narrowband MDCT Frame

Parameter (value)	Number of bits	Bit index in packet
LSP_IDX	28	P0-P27
MDCT_INDEX	131	P28-P158
FRAME_GAIN_INDEX	7	P159-P165
NOISE_GAIN_INDEX	2	P166-P167
NB_WB_MODE_BIT (0)	1	P168
MODEBIT (CELP/ MDCT) (1)	1	P169
EVRCB_IO_FLAG (1)	1	P170

Table 4.21-5 Packet Formatting for Half-Rate Wideband NELP Frame

Parameter (value)	Number of bits	Bit index in packet
WB_HALFRATE_IDENTIFIER (0x3F)	6	P0-P5
LSP_IDX	28	P6-P33
NELP_GAIN_INDEX	17	P34-P50
NELP_FILTER_ID	2	P51-P52
HIGHBAND_LSP_IDX	12	P53-P64
HIGHBAND_GAIN_SHAPE_IDX	8	P65-P72
HIGHBAND_GAIN_FRAME_IDX	7	P73-P79

Table 4.21-6 Packet Formatting for Half-Rate CELP in SO 68 Interoperable Mode

Parameter (value)	Number of bits	Bit index in packet
DelayIndex	7	P0-P6
LSP_IDX0	7	P7-P13
LSP_IDX1	7	P14-P20

Parameter (value)	Number of bits	Bit index in packet
LSP_IDX2	8	P21-P28
ACBG_IDX_SF0	3	P29-P31
ACBG_IDX_SF1	3	P32-P34
ACBG_IDX_SF2	3	P35-P37
FCB_PULSE_IDX0_SF0	10	P38-P47
FCB_PULSE_IDX0_SF1	10	P48-P57
FCB_PULSE_IDX0_SF2	10	P58-P67
FCBG_IDX_SF0	4	P68-P71
FCBG_IDX_SF1	4	P72-P75
FCBG_IDX_SF2	4	P76-P79

Table 4.21-7 Packet Formatting for Eighth-Rate in SO 68 Interoperable Mode

Parameter (value)	Number of bits	Bit index in packet
LSP_IDX0	4	P0-P3
LSP_IDX1	4	P4-P7
GAIN IDX	6	P8-P13
RESERVED0 (0)	1	P14-P14
RESERVED1 (0)	1	P15-P15

Table 4.21-8 Packet Formatting for Eighth-Rate in Wideband Mode

Parameter (value)	Number of bits	Bit index in packet
LSP_IDX	10	P0-P9
GAIN IDX	5	P10-P14
WIDEBAND_FLAG (1)	1	P15-P15

Table 4.21-9 Packet Formatting for Special Half-Rate Wideband CELP Frame (for Packet Level Signaling)

Parameter (value)	Number of bits	Bit index in packet
SpecialHalfCelpFlag0(0x7B)	7	P0-P6
LSP_IDX	28	P7-P34
ACBG_IDX_SF0	3	P35-P37
ACBG_IDX_SF1	3	P38-P40
ACBG_IDX_SF2	3	P41-P43
DelayIndex	7	P44-P50
~~EVRCWB_SPL_HALF_MODE (0x00)~~	~~2~~	~~P51-P52~~
RESERVED (all 0s)	2~~9~~7	P51~~3~~-P79

Table 4.21-10 Packet Formatting for Special Half-Rate CELP in SO 68 Interoperable Mode

Parameter (value)	Number of bits	Bit index in packet
SpecialHalfCelpFlag0(0x6E)	7	P0-P6
SpecialHalfCelpFlag1(0x00)	2	P7-P8
ModeBit (0)	1	P9-P9
LSP_IDX0	6	P10-P15
LSP_IDX1	6	P16-P21
LSP_IDX2	9	P22-P30
LSP_IDX3	7	P31-P37
ACBG_IDX_SF0	3	P38-P40
ACBG_IDX_SF1	3	P41-P43
ACBG_IDX_SF2	3	P44-P46
DelayIndex	7	P47-P53
RESERVED (all 0s)	26	P54-P79

Table 4.21-11 Packet Formatting for Special Half-Rate NELP

Parameter (value)	Number of bits	Bit index in packet
SpecialHalfNelpFlag0(0x6E)	7	P0-P6
SpecialHalfNelpFlag1(0x01)	2	P7-P8
LSP_IDX0	8	P9-P16
LSP_IDX1	8	P17-P24
NELP_GAIN_IDX0	5	P25-P29
NELP_GAIN_IDX1	6	P30-P35
NELP_GAIN_IDX2	6	P36-P41
NELP_FILTER_ID	2	P42-P43
RESERVED (all 0s)	36	P44-P79

Table 4.21-12 Packet Formatting for Special Half-Rate Wideband and Narrowband MDCT Frame (for Packet Level Signaling)

Parameter (value)	Number of bits	Bit index in packet
SpecialHalfMDCTFlag0(0x7A)	7	P0-P6
LSP_IDX	28	P7-P34
FRAME_GAIN_INDEX	7	P35-P41
NOISE_GAIN_INDEX	2	P42-43
RESERVED (all 0s)	36	P44-P79

The ModeBit in Table 4.21-1indicates if a packet is a PPP packet or a CELP/NELP packet. As none of SO 70 packets is coded with PPP scheme, ModeBit is always '0' for a SO 70 packet.

If EVRCB_IO_FLAG in Table 4.21-1 – Table 4.21-4 is set to '1', the packet is a SO 70 COP 0 full rate packet. Otherwise, the packet is a full rate packet of SO 70 COP 4 or SO 68 COP 0 – COP 6.

P15 of an eighth rate packet of SO 70 COP 0 is set to '1', like in Table 4.21-8. Otherwise, the eighth rate packet belongs to SO 70 COP 4 or COP 7, or SO 68.

The MODEBIT (CELP/MDCT) of a SO 70 COP 0 full rate packet, as appeared in Table 4.21-2 – Table 4.21-4, is an indication if the packet is a CELP packet ('0') or a MDCT packet ('1').

The NB_WB_MODE_BIT of a SO 70 COP 0 full rate MDCT packet, as appeared in Table 4.21-3 and Table 4.21-4, is an indication if the packet is a NB MDCT packet ('0') or a WB MDCT packet ('1').

4.21.1 Packet Level Signaling Interworking Function

A packet level signaling interworking function that needs to dim a wideband Full-Rate packet CELP packet to a special Half-Rate CELP packet should receive a packet in the frame in the format described in Table 4.21-2 and convert it to the format described in Table 4.21-9 by relocating some of the parameters (those that are common between Table 4.21-2 and Table 4.21-9) and setting the SpecialHalfRateCELP flag. The highband parameters are dropped and the decoder handles the highband part as in Frame Erasure processing.

A packet level signaling interworking function that needs to dim a EVRC-B interoperable Full-Rate packet CELP packet to a EVRC-B interoperable special Half-Rate CELP packet should receive a packet in the frame in the format described in Table 4.21-1 and convert it to the format described in Table 4.21-10 by relocating some of the parameters (those that are common between the two tables) and setting the SpecialHalfRateCELP flag.

4.22 Packet Formatting for SO 73

The packet formatting for SO 73 includes these for both SO 68 and SO 70.

1

2

3

4

5

6   This page intentionally left blank.

1 # 5 SPEECH DECODER

2
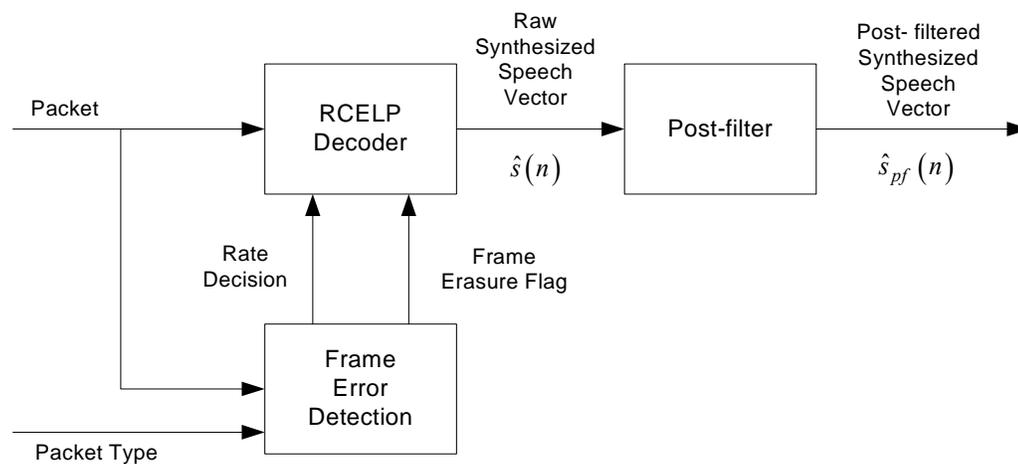
3 **Figure 5-1 SO 3 Speech Decoder Top-Level Diagram**

4

5 Figure 5-1 presents a top-level view of the EVRC speech decoder for SO 3. A top-level view of the EVRC-B speech
6 decoder for SO 68 is presented in Figure 4.2-2. A top-level view of the EVRC-WB and EVRC-NW speech decoder for
7 SO 70 and SO 73 is presented in Figure 4.3-3. The inputs to the decoder are the received speech packet, and a packet
8 type indicator from the multiplex sub-layer. The frame error detection module uses the packet type indicator to determine
9 the data rate and whether or not there was a frame error detected by the multiplex sub-layer. The decoder also applies
10 rules to detect some channel errors not detected by the multiplex sub-layer.

11 The decoder uses the parameters contained in the received packet to re-synthesize the speech frame based on the rate
12 decision. It uses the frame erasure flag to trigger frame error recovery logic. The raw synthesized speech is then post-
13 filtered and output.

14 ## 5.1 Frame Error Detection

15 **Inputs:** The inputs to frame error detection are:

16 • The packet type supplied by the multiplex sublayer

17 • The delay transmission code, *DELAY*

18 **Outputs:** The outputs of frame error detection are:

19 • The frame erasure flag, *FER_FLAG*(m)

20 • The rate of operation for the decoder, *Rate*

21 **Processing:** The frame error detector shall comprise 5.1.1 through 5.1.6.

22 **Initialization:**

23 • The last valid rate of operation, *last_valid_rate*, is initialized to Rate 1/8

1 • The frame erasure flag, *FER_FLAG*(*m*) = FALSE ; *m* = 0

5.1.1 Received Packet Type Processing for SO 3

3 The received packet type from the multiplex sublayer (see 2.1.2) is used to generate the decoder rate of operation, *Rate*,
4 as well as the frame erasure flag, *FER_FLAG*(*m*). *FER_FLAG*(*m*), is defined for each received packet type in Table
5 5.1.1-1.

6 **Table 5.1.1-1 Received Packet Type Decoding for SO 3**

Packet type	*FER_FLAG*(*m*)
Rate 1	FALSE
Rate 1/2	FALSE
Rate 1/4	TRUE
Rate 1/8	FALSE
Blank	TRUE
Insufficient frame quality (erasure)	TRUE
Rate 1/8 with all bits set to '1'	TRUE
Rate 1 with all bits set to '0'	TRUE
Rate 1/2 with all bits set to '0'	TRUE
~~Rate 1/4 with all bits set to '0'~~	~~TRUE~~
~~Rate 1/8 with all bits set to '0'~~	~~TRUE~~

7

8 The decoder rate of operation for the current frame, *Rate*, is then defined by the pseudo-code:

9   if (*FER_FLAG*(*m*) == TRUE) {

10   if (*last_valid_rate* == Rate 1/8) {

11   *Rate* = Rate 1/8

12   }

13   else {

14   *Rate* = Rate 1

15   }

16   }

17   if ((*Rate* == Rate 1/8) and (*last_valid_rate* == Rate 1) and

18   (*FER_FLAG*(*m*-1) == FALSE)) {

19   *FER_FLAG*(*m*) = TRUE

20   *Rate* = Rate 1

21   }

22   if (*FER_FLAG*(*m*) == FALSE) {

1   *Rate* = Rate from received packet type

2   *last_valid_rate* = *Rate*

3   }

5.1.2 Received Packet Type Processing for SO 68

5 The received packet type from the multiplex sublayer (see 2.1.2) is used to generate the decoder rate of operation, *Rate*,
6 as well as the frame erasure flag, *FER_FLAG*(*m*). *FER_FLAG*(*m*), is defined for each received packet type in Table
7 5.1.2-1.

Table 5.1.2-1 Received Packet Type Decoding for SO 68

Packet Type	FER_FLAG(m)
Rate 1	FALSE
Rate 1/2	FALSE
Rate 1/4	FALSE
Rate 1/8	FALSE
Blank with last valid received rate not equal to Rate 1/8	TRUE
Blank with last valid received rate Rate 1/8	FALSE
Insufficient frame quality (erasure)	TRUE
Rate 1/8 with all bits set to '1'	TRUE
Rate 1 with all bits set to '0'	TRUE
Rate 1/2 with all bits set to '0'	TRUE
Rate 1/4 with all bits set to '0'	TRUE
Rate 1/8 with all bits set to '0'	TRUE

10 The decoder rate of operation for the current frame, *Rate*, is then defined by the pseudo-code:

11   if (*FER_FLAG*(*m*) == TRUE) {

12   *Rate = last_valid_rate*

13   ~~if (*last_valid_rate* == Rate 1/8) {~~

14   ~~*Rate* = Rate 1/8~~

15   ~~}~~

16   ~~else {~~

17   ~~*Rate* = Rate 1~~

18   ~~}~~

19   }

20   if (((*Rate* == Rate 1/8) and (*last_valid_rate* == Rate 1) and

21   (*FER_FLAG*(*m*-1) == FALSE)) {

22   *FER_FLAG*(*m*) = TRUE

Rate = Rate 1

}

if (*FER_FLAG*(*m*) == FALSE) {

if (Rate == Blank) {

Rate = *last_valid_rate* (also same as Rate 1/8)

}

else {

Rate = Rate from received packet type

last_valid_rate = *Rate*

}

}

The same pseudo-code applies to SO 70 and SO 73.

5.1.3 Received Packet Type Processing for SO 70 ~~and SO 73 (in WB Mode)~~

Service Option~~s~~ 70 ~~and 73 (in WB mode)~~ treat~~s~~ received packets similar to service option 68 as described in Section 5.1.2. *FER_FLAG*(*m*), is defined for each received packet type in Table 5.1.3-1. ~~, with the exception of insufficient frame quality frames (erasures). Erasures are first checked to determine if they are suppressed Eighth-Rate frames produced by DTX.~~

~~Before the decoder rate of operation for the current frame, *Rate*, is determined by the pseudo-code in Section 5.1.2, the following pseudo-code examines erasure frames for possible suppressed Eighth-Rate frames:~~

~~if (*Rate* == Erasure) {~~

~~if ((*last_valid_rate* == Rate 1/8) or (*dtx_likely_flag* == TRUE)) {~~

~~*Rate* = Suppressed Rate 1/8~~

~~}~~

~~}~~

Table 5.1.3-1 Received Packet Type Decoding for SO 70

Packet Type	FER_FLAG(m)
Rate 1	FALSE
Rate 1/2	FALSE
Rate 1/4	TRUE
Rate 1/8	FALSE
Blank with last valid received rate not equal to Rate 1/8	TRUE
Blank with last valid received rate Rate 1/8	FALSE
Insufficient frame quality (erasure)	TRUE
Rate 1/8 with all bits set to '1'	TRUE
Rate 1 with all bits set to '0'	TRUE

Packet Type	FER_FLAG(m)
Rate 1/2 with all bits set to '0'	TRUE
Rate 1/8 with all bits set to '0'	TRUE

Further, if the decoder operates in wideband mode, *FER_FLAG(m) = FALSE,* and the current packet is Rate 1 (or Rate 1/8) with P170 = 0 (or P15 = 0), then *FER_FLAG(m)* = TRUE and *Rate = last_valid_rate*. If the next packet is again Rate 1 (or Rate 1/8) with P170 = 0 (or P15 = 0) with *FER_FLAG(m+1) = FALSE*, then the decoder decodes $(m+1)^{th}$ packet as a narrowband packet and operates in narrowband mode. Similar procedure applies to the decoder operation mode transition from narrowband to wideband. Here is the pseudo-code to illustrate this process:

```
if ( MODE == MODE_WB) {
    if ( FER_FLAG(m) == FALSE ) {
        if (Rate == Rate 1 (or Rate 1/8)) {
            if (P170 (or P15) == 0) {
                if (counter_wb>0) {
                    MODE = MODE_NB
                    counter_wb = 0
                }
                else {
                    counter_wb = counter_wb+1
                    Rate = last_valid_rate
                    FER_FLAG(m) == TRUE
                }
            }
            else {
                counter_wb = 0
            }
        }
        else {
            counter_wb = 0
        }
    }
}
```

Similarly,

```
if ( MODE == MODE_NB) {
    if ( FER_FLAG(m) == FALSE ) {
```

```
            if (Rate == Rate 1 (or Rate 1/8)) {
                if (P170 (or P15) == 1) {
                    if (counter_nb>0) {
                        MODE = MODE_WB
                        counter_nb = 0
                    }
                    else {
                        counter_nb = counter_nb+1
                        Rate = last_valid_rate
                        FER_FLAG(m) == TRUE
                    }
                }
                else {
                    counter_nb = 0
                }
            }
            else {
                counter_nb = 0
            }
        }
    }
```

This transition in decoder does not cause any change of capacity operating point in encoder.

5.1.4 Received Packet Type Processing for SO 73

Service Option 73 treats received packets similar to service option 68 as described in Section 5.1.2. *FER_FLAG(m)*, is defined for each received packet type in Table 5.1.4-1.

Table 5.1.4-1 Received Packet Type Decoding for SO 73

Packet Type	FER_FLAG(m)
Rate 1	FALSE
Rate 1/2	FALSE
Rate 1/4	FALSE (NB mode) TRUE (WB mode)
Rate 1/8	FALSE
Blank with last valid received rate not equal to Rate 1/8	TRUE

Packet Type	FER_FLAG(m)
Blank with last valid received rate Rate 1/8	FALSE
Insufficient frame quality (erasure)	TRUE
Rate 1/8 with all bits set to '1'	TRUE
Rate 1 with all bits set to '0'	TRUE
Rate 1/2 with all bits set to '0'	TRUE
Rate 1/4 with all bits set to '0'	TRUE
Rate 1/8 with all bits set to '0'	TRUE

1 Further, if the decoder operates in wideband mode, *FER_FLAG(m) = FALSE,* and the current packet is Rate 1 (or Rate
2 1/8) with P170 = 0 (or P15 = 0), then *FER_FLAG(m)* = TRUE and *Rate = last_valid_rate*. If the next packet is again
3 Rate 1 (or Rate 1/8) with P170 = 0 (or P15 = 0) with *FER_FLAG(m+1) = FALSE*, then the decoder decodes $(m+1)^{th}$
4 packet as a narrowband packet and operates in narrowband mode. Similar procedure applies to the decoder operation
5 mode transition from narrowband to wideband if wideband mode is supported and activated. This process is illustrated
6 by the same pseudo-code as in 5.1.3 . This transition in decoder does not cause any change of capacity operating point in
7 encoder.

5.1.4 5.1.5 Packet Sanity Check for Bad Rate Detection for SO 3

5.1.4.1 5.1.5.1 Delay Parameter Checking

10 If *Rate* is determined to be Rate 1/2 or 1, and the delay transmission code, *DELAY*, is greater than 100, the
11 *FER_FLAG*(*m*) flag shall be set to TRUE.

5.1.4.2 5.1.5.2 Delta Delay Parameter Checking

13 If *Rate* is determined to be Rate 1, the delta delay transmission code, DDELAY, shall be sanity checked. The last
14 frame's delay shall be computed according to Equation 5.2.2-5. If the computed delay is less than DMIN or greater than
15 DMAX, then the *FER_FLAG(m)* flag shall be set to TRUE.

5.1.5 5.1.6 Packet Sanitycheck for Badrate Detection for SO 68

5.1.5.1 5.1.6.1 Delay Parameter Checking

18 If Rate is determined to be Rate 1/2 or 1, and the delay transmission code, DELAY, is greater than 100, and not equal to
19 120 or 115, the FER_FLAG (m) flag shall be set to TRUE.

5.1.5.2 5.1.6.2 Delta Delay Parameter Checking

21 If Rate is determined to be Rate 1, the delta delay transmission code, DDELAY, shall be sanity checked. The last frame's
22 delay shall be computed according to Equation 5.2.2-5. If the computed delay is less than DMIN or greater than DMAX,
23 then the FER_FLAG(m) flag shall be set to TRUE.

~~5.1.5.3~~ 5.1.6.3 LSP Checking

If Rate is determined to be Rate 1/2, decode the LSP parameters, $\Omega q(m)$, as defined in Section 4.10, using the decoded LSP indices, LSPIDX(k). If MAX($\Omega q(0)$, $\Omega q(1)$, $\Omega q(2)$) >= MIN($\Omega q(6)$, $\Omega q(7)$, $\Omega q(8)$, $\Omega q(9)$), the FER_FLAG (m) flag shall be set to TRUE.

If Rate is determined to be Rate 1, decode the LSP parameters, $\Omega q(m)$, as defined in Section 4.10, using the decoded LSP indices, LSPIDX(k). If MAX($\Omega q(0)$, $\Omega q(1)$) > MIN($\Omega q(4)$, $\Omega q(5)$, $\Omega q(6)$), the FER_FLAG (m) flag shall be set to TRUE.

~~5.1.5.4~~ 5.1.6.4 Unused Bit Checking

If Rate is determined to be Rate 1/8, and bit 15 of the received packet is 1, the FER_FLAG (m) flag shall be set to TRUE.

If Rate is determined to be Rate 1/4 and frame is a NELP frame, and if any of the bits 36, 37, 38, 39 of the received packet is 1, the FER_FLAG (m) flag shall be set to TRUE.

If Rate is determined to be Rate 1/4 and frame is a PPP frame, and if bit 39 of the received packet is 1, the FER_FLAG (m) flag shall be set to TRUE.

~~5.1.5.5~~ 5.1.6.5 Invalid Filter ID

If Rate is determined to be Rate 1/4 and frame is a NELP frame, and the shaping filter ID is equal to 3, the FER_FLAG (m) flag shall be set to TRUE.

~~5.1.5.6~~ 5.1.6.6 Invalid Transition

If Rate is determined to be Rate 1/4 and the frame is a PPP frame, and if the previous frame is a Rate 1/8, or a Rate 1/4 NELP frame, the FER_FLAG (m) flag shall be set to TRUE.

If Rate is determined to be Rate 1 and the frame is a PPP frame, and if the previous frame is a Rate 1/8, or a Rate 1/4 NELP frame, the FER_FLAG (m) flag shall be set to TRUE.

5.2 Rate 1/2 and 1 CELP Decoding for SO 3, SO 68, SO 70, and SO 73

Inputs: The inputs to Rate 1/2 and 1 CELP decoding are:

- The quantized LSP indices from the current frame, *LSPIDX*(*k*)

- The quantized LSPs from the previous frame, $\Omega_q(m\text{-}1)$

- The adaptive codebook gain index, *ACBGIDX*(*m'*), for each subframe, *m'*,

- The fixed codebook shape indices, *FCBSIDX*(*m'*, *k*), for each subframe, *m'*,

- The fixed codebook gain indices, *FCBGIDX*(*m'*, *k*), for each subframe, *m'*,

- The delay transmission code, *DELAY*

- The spectral transition indicator, *LPCFLAG*, Rate 1 only

- The delay difference, *DDELAY*, Rate 1 only

- The frame erasure flag, *FER_FLAG*(*m*), for the current frame, *m*

Outputs: The outputs from Rate 1/2 and 1 CELP decoding are:

- The post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$

State Variables:

- The adaptive codebook excitation, $E(n)$

Initialization:

- The adaptive codebook excitation memory, $E(n) = 0$; $-128 \le n < 64$

- The memory for the last valid adaptive codebook buffer, $\{E_{lv}(n)\} = 0$; $0 \le n < 128$

- The delay, $\tau(m) = 40$; $m = 0$

- The quantized LSPs, $\Omega_q(m,k) = 0.048k$; $1 \le k \le 10, m = 0$

- The codebook and pitch gains for the last valid frame (see 5.2.3.5 and 5.2.3.6)

    $g_c(m') = g_p(m')= 0$; $0 \le m' \le 2$

- The fader scaler variable, $\alpha_f = 1$

Processing: Rate 1/2 and 1 decoding shall comprise 5.2.1 to 5.2.3

5.2.1 Decoding of the LSP Parameters

If (*FER_FLAG(m) = TRUE*), use the LSP parameters from the last frame to generate bandwidth expanded LSPs for the current frame as follows: $\Omega_q(m) = 0.875\ \Omega_q(m-1) + 0.125\ \Omega_{spread}$ where Ω_{spread} are the initial values of the LSPs, as defined in Section 5.2.

Otherwise, decode the LSP parameters, $\Omega_q(m)$, as defined in 4.9.5, using the decoded LSP indices, *LSPIDX(k)*. In the event that the decoded LSPs are not strictly ascending, *FER_FLAG(m)* shall be set to *TRUE*, where "strictly ascending" is defined as $\Omega_q(m, i) < \Omega_q(m, i+1)$; $1 \le i \le 9$.

5.2.2 Delay Decoding and Frame Erasure Delay Contour Reconstruction

5.2.2.1 Delay Decoding

The delay parameter, $\tau(m)$, for the current frame is defined as:

$$\tau(m) = \begin{cases} \tau(m-1) & ; FER\_FLAG(m) = TRUE, \\ DELAY + 20 & ; otherwise, \end{cases} \tag{5.2.2-1}$$

where *DELAY* is the pitch delay transmission code.

5.2.2.2 Frame Erasure Delay Contour Reconstruction for Rate 1 for SO 3

If (*FER_FLAG(m) = FALSE*) and (*FER_FLAG(m-1) = TRUE*) and (*Rate* = Rate 1), the delay contour shall be reconstructed (for Rate 1 operation only) by warping the last valid adaptive codebook memory buffer, $E_{lv}(n)$, using a recovered delay contour. $E_{lv}(n)$ is first copied to the current adaptive codebook excitation memory, $E(n)$:

$$E(n\text{-}128) = E_{lv}(n); \quad 0 \le n < 128 \tag{5.2.2-2}$$

where $E_{lv}(n)$ was determined in 5.2.3.9 for the last valid frame, and E(n) is the adaptive codebook excitation memory.

5.2.2.3 Frame Erasure Delay Contour Reconstruction for Rate 1 Good Frame Following an Erased Frame for SO 68

If (FER_FLAG(m) = FALSE) and (FER_FLAG(m-1) = TRUE) and (Rate = Rate 1) and if the previous frame was neither Silence nor Rate 1/4 NELP, the delay contour shall be reconstructed (for Rate 1 operation only) by warping the last valid adaptive codebook memory buffer, $E_{lv}(n)$, using a recovered delay contour. Elv(n) is first copied to the current adaptive codebook excitation memory, E(n):

$$E(n\text{-}128) = E_{lv}(n); \quad 0 \le n{<}128$$

(5.2.2-3)

where $E_{lv}(n)$ was determined in Section 5.2.3.10 for the last valid frame, and E(n) is the adaptive codebook excitation memory.

5.2.2.3.1 Delay Reconstruction

Calculate a delay, τ', which corresponds to the delay of the last valid frame:

$$\tau' = \tau(m\text{-}1),$$

(5.2.2-4)

and recover the last frame's delay as computed in the encoder by using the delay difference:

$$\tau(m\text{-}1) = \tau(m) - DDELAY - 16,$$

(5.2.2-5)

where *DDELAY* is the received delay difference transmission code, and $\tau(m)$ is the current frame delay parameter.

Limit τ' by:

$$\tau' = \begin{cases} \tau(m\text{-}1) & ; |\tau(m\text{-}1) - \tau'| > 15, \\ \tau' & ; otherwise, \end{cases}$$

(5.2.2-6.)

E(n) is then warped using τ' and $\tau(m\text{-}1)$ in 5.2.2.3.2 and 5.2.2.3.3 for each subframe $0 \le m' < 3$.

5.2.2.3.2 Reconstruction of the Delay Contour

The reconstructed delay contour, $\tau_c(n)$, is generated as defined in 4.11.5.1 of the encoder, using a set of delay interpolations computed by:

$$\overset{\bullet}{d}(m', j) = \begin{cases} \tau(m-1) & ; |\ \tau(m-1) - \tau'\ | > 15, \\ (1 - f(m' + j))\,\tau(m-1) + f(m' + j)\tau' & ; otherwise, \end{cases} \quad 0 \le j < 3,$$

(5.2.2-7)

where *m* is the current frame, *m'* is the subframe index, and *m'+j* is an index into an array of interpolator coefficients, f, defined by:

$$f = \{\ 0.0, 0.3313, 0.6625, 1.0, 1.0\ \}.$$

(5.2.2-8)

5.2.2.3.3 Warping of the Adaptive Codebook Memory

The adaptive codebook memory, E(n), is then mapped to the delay contour, $\tau_c(n)$, as described in 4.11.5.2 of the encoder.

5.2.2.4 Frame Erasure Delay Contour Reconstruction for Rate 1 Good Frame Following a QPPP frame Which Follows an Erased Frame in SO 68

If (FER_FLAG(m) = FALSE) and (FER_FLAG(m-1) = FALSE) and (FER_FLAG(m-2) = TRUE) and (Rate = Rate 1) and if the previous frame was QPPP, the delay contour shall be reconstructed (for Rate 1 operation only) by warping the valid prior to the last valid adaptive codebook memory buffer, Elvlv(n), using a recovered delay contour. Elvlv(n) is first copied to the current adaptive codebook excitation memory, E(n):

$$E(n-128) = E_{lvlv}(n); \qquad 0 \le n < 128$$

(5.2.2-9)

where Elvlv(n) was determined in Section 5.2.3.10 for the last valid frame, and E(n) is the adaptive codebook excitation memory.

5.2.2.4.1 Delay Reconstruction

Recover the prior to the last frame's delay as computed in the encoder by using the delay difference:

$$\tau'' = \tau\,(m\text{-}2)\,,$$

(5.2.2-10)

and recover the prior to the last frame's delay as computed in the encoder by using the delay difference:

$$\tau\,(m\text{-}2) = \tau\,(m) - DELAY - 16 - DELTADELAY,$$

(5.2.2-11)

where DELTADDELAY is the received delay difference transmission code, and τ (m) is the current frame delay parameter.

E(n) is then warped using τ'' and τ (m-2) in 5.2.2.3.2 and 5.2.2.3.3 for each subframe $0 \le m' < 3$.

E(n) is then further warped using τ (m-2) and τ (m-1) in 5.2.2.3.2 and 5.2.2.3.3 for each subframe $0 \le m' < 3$ to obtain an adaptive codebook buffer using a recovered delay contour.

5.2.2.4.2 Reconstruction of the Delay Contour

The reconstructed delay contour, $\tau c(n)$, is generated as defined in Section 4.11.5.1 of the encoder, using a set of delay interpolations computed by:

$$\dot{d}(m',j) = \begin{cases} \tau(m-1) & ;\left| \ \tau(m-1)-\tau' \ \right| > 15, \\ \big(1-f(m'+j)\big)\,\tau(m-1)+f(m'+j)\tau' & ;otherwise, \end{cases} \qquad 0 \le j < 3,$$

(5.2.2-12)

where m is the current frame, m' is the subframe index, and m'+j is an index into an array of interpolator coefficients, f, defined by:

$$f = \{\ 0.0,\ 0.3313,\ 0.6625,\ 1.0,\ 1.0\ \}.$$

(5.2.2-13)

5.2.2.4.3 Warping of the Adaptive Codebook Memory

The adaptive codebook memory, E(n), is then mapped to the delay contour, $\tau c(n)$, as described in 4.11.5.2 of the encoder. For the first subframe, m', the mapped adaptive codebook memory is scaled by a value g given by:

$$g = \begin{cases} MIN\big(g_{avg},0.95\big); & for\ m' = 0 \\ 0.95; & otherwise \end{cases}$$

where gpavg is the average adaptive codebook gain value.

5.2.2.5 Smoothing of the Decoded Delay

The delay of the previous frame, $\tau(m\text{-}1)$, shall be smoothed for delay interpolation in 5.2.3.4 by:

$$\tau(m-1) = \begin{cases} \tau(m) & ; \mid \tau(m) - \tau(m-1) - \tau' \mid \; > 15, \\ \tau(m-1) & ; otherwise, \end{cases}$$

(5.2.2-14)

5.2.3 Rates 1/2 and 1 Subframe Decoding

Compute the decoded synthesized speech signal for each subframe, $0 \le m' < 3$, as described in 5.2.3.1 through 5.2.3.10. The subframe size, *L*, is 53 for subframes 0 and 1, and 54 for subframe 2.

5.2.3.1 Interpolation of LSP Parameters

Interpolate the quantized LSPs over the three subframes, *m'*, in the current frame, *m*. The form of the quantized, interpolated LSP vector is:

$$\dot{\Omega}_q(m') = (1 - \mu_{m'})\Omega_q(m-1) + \mu_{m'}\Omega_q(m),$$

(5.2.3-1)

where the subframe interpolator constants are defined as μ = { 0.1667, 0.5, 0.8333 }.

5.2.3.2 LSP to LPC Conversion

Convert the quantized, interpolated LSPs, $\dot{\Omega}_q(m)$, to quantized, interpolated LPC parameters, $\left\{ \dot{a}_q \right\}$, as described in 4.6.2.2 for each subframe.

5.2.3.3 Bandwidth Expansion

If the decoded *LPCFLAG* transmission code = *TRUE*, and (*FER_FLAG(m* - 1) = *TRUE*), the interpolated LPC parameters, $\left\{ \dot{a}_q \right\}$, shall be bandwidth expanded using:

$$\dot{a}_q(k) = (0.75)^K \dot{a}_q(k); \qquad 1 \le k < 10.$$

(5.2.3-2)

5.2.3.4 Interpolated Delay Estimate Calculation.

Calculate the set of interpolated delay estimates, $\dot{d}(m', j)$, as in 4.11.4.3 of the encoder. If (*FER_FLAG(m)* = *TRUE*), and the average adaptive codebook gain (see 5.2.3.5) for the last valid frame, ($g_{pavg} < 0.3$), the interpolated delay estimates are defined as:

$$\dot{d}(m', j) = d_{md}(m' + j); \qquad 0 \le j < 3$$

(5.2.3-3)

where $d_{rnd}(k)$ is the kth element of:

$$d_{md} = \{55.0, 80.0, 39.0, 71.0, 33.0\},$$

(5.2.3-4)

5.2.3.5 Calculation of the Adaptive Codebook Contribution

The average adaptive codebook gain, g_{pavg} is computed by:

5.2.3.5.1 SO 3

$$
g_{pavg} = \begin{cases} g_{pavg}(m\text{-}1) & ; FER\_FLAG(m) = TRUE \ and \ FER\_FLAG(m\text{-}1) = \text{FALSE}, \\ 0.75 g_{pavg}(m\text{-}1) & ; FER\_FLAG(m) = TRUE \ and \ FER\_FLAG(m\text{-}1) = \text{TRUE}, \\ \dfrac{1}{3}\sum_{i=0}^{2} g_p(i) & ; FER\_FLAG(m) = \text{FALSE}, \end{cases}
\tag{5.2.3-5}
$$

5.2.3.5.2 SO 68

$$
g_{pavg} = \begin{cases} g_{pavg}(m-1) & ; FER\_FLAG(m) = TRUE \ and \ FER\_FLAG(m-1) = \text{FALSE} \\ 0.75 g_{pavg}(m-1) & ; FER\_FLAG(m) = TRUE \ and \ FER\_FLAG(m-1) = TRUE \\ \dfrac{g_p(0) + 2g_p(1) + 3g_p(2)}{6} & ; FER\_FLAG(m) = \text{FALSE} \end{cases}
$$

where $g_{pcb}(k)$ is an entry in the adaptive codebook gain quantization Table 4.11.4.9-1, and $ACBGIDX(m')$ is the decoded adaptive codebook gain index.

The adaptive codebook gain, $g_p(m')$, shall be defined as:

$$
g_p(m') = \begin{cases} g_{pavg}(m) & ; FER\_FLAG(m) = TRUE, \\ g_{pcb}(ACBGIDX(m')) & ; FER\_FLAG(m) = FALSE, \end{cases}
\tag{5.2.3-6}
$$

For SO 68, if the previous frame was erased, and was processed as a silence or a NELP frame, then the current frame adaptive codebook gain, $g_p(m')$ is set to zero.

Calculate the adaptive codebook excitation, E(n), as described in 4.11.5 of the encoder.

5.2.3.6 Calculation of the Fixed Codebook Gain

The fixed codebook gain, $g_c(m')$, is defined as:

$$
g_c(m') = \begin{cases} g_{cavg} & ; FER\_FLAG(m) = TRUE, \\ g_{ccb}(FCBGIDX(m')) & ; otherwise, \end{cases}
\tag{5.2.3-7}
$$

where the average fixed codebook gain, g_{cavg}, is defined by:

$$
g_{cavg} = \frac{1}{3}\sum_{i=0}^{2} g_c(i),
\tag{5.2.3-8}
$$

calculated over $g_c(i)$ for the last valid frame; $g_{ccb}(k)$ is an entry in the fixed codebook gain quantization Table 8-15, and $FCBGIDX(m')$ is the fixed codebook gain index.

5.2.3.7 Computing of the Reconstructed ACELP Fixed Codebook Excitation for SO 3

If ($FER\_FLAG(m) = TRUE$), the ACELP contribution is $c(n) = 0$; $0 \le n \le 54$, otherwise generate the fixed codebook contribution as follows:

For Rate 1 frames, the position of the two-single-pulse tracks, q, is determined by using *FCBSIDX(m′,3)[9,10]* as given in Table 4.11.7.2.2-1. The remaining three tracks correspond to double-pulse tracks.

The single and double-pulse track positions in the decimated domain, for $0 \le i \le 3$, are determined by:

$$\lambda_0 = \left\lfloor \frac{FCBSIDX(m',i)[0,...,6]}{11} \right\rfloor ,$$
(5.2.3-9)

$$\lambda_1 = FCBSIDX(m',i)[0,...,6] - 11\lambda_0 .$$
(5.2.3-10)

The sign of the three double-pulse tracks is determined by using the relationship between λ_0 and λ_1 as well as *FCBSIDX(m′, i)* [7] for i = 0, 1, 2. If $\lambda_0 < \lambda_1$, then the two pulses have the same sign, as specified by *FCBSIDX(m′, i)* [7]. If $\lambda_0 < \lambda_1$, then the two pulses have different signs, and *FCBSIDX(m′, i)* [7] specifies the sign of the pulse at position λ_0 as described in 4.11.7.3. The sign of the two single-pulse tracks is given by the codeword *FCBSIDX(m′, 3)* [7] and *FCBSIDX(m′, 3)* [8], accordingly.

Next, referring to Equations 4.9.7.3-1 and 4.9.7.3-2 and Table 4.11.7.1-1 and Table 4.11.7.2.2-1, the pulse positions in the decimated domain shall be converted to the undecimated domain.

For Rate 1/2, *FCBSIDX(m′, 0)* [9] indicates the sign of the first and the third pulse track, and the opposite sign of the 2nd pulse track, and *FCBSIDX(m′, 0)* [0,...,8] indicates the positions of three pulses in the three tracks accordingly, as given in the Table 4.11.7.4-1.

The received algebraic codebook index is used to extract the positions and amplitudes (signs) of the excitation pulses and to construct the algebraic codevector, c, according to Equation 4.9.7.1-1. If the pitch delay, τ, is less than the subframe size 55, the pitch sharpening processing shall be applied which modifies $c(n)$ by $c(n)=c(n)+g_P c(n\text{-}\tau)$, where g_P is the decoded pitch gain bounded by [0.2, 0.9].

5.2.3.8 Computing of the Reconstructed Fixed Codebook Excitation in SO 68

If (FER_FLAG(m) = TRUE), the FACTORIAL CODEBOOK contribution is c(n) = 0 ; $0 \le n \le 54$, otherwise generate the fixed codebook contribution as follows:

For Rate 1 frames, the Fixed codebook codeword is converted to the fixed codebook vector, c(n) as described in Section 4.11.4.11.

For Rate 1/2, the Fixed codebook codeword is converted to the fixed codebook vector, c(n) as described in Section 4.11.4.11.

If the pitch delay, τ, is less than the subframe size 55, the pitch sharpening processing shall be applied which modifies c(n) by c(n)=c(n)+gPc(n-τ) , where gP is the decoded pitch gain bounded by [0.2, 0.9].

5.2.3.9 Decoder Total Excitation Generation

Add the fixed and adaptive codebook contributions to obtain the total excitation, $E_T(n)$, for the current frame as defined by:

$$E_T(n) = \begin{cases} g_p E(n); & FER\_FLAG(m) = TRUE , \\ g_p E(n) + g_c\, c(n); & otherwise, \end{cases} ;0 \le n < L .$$
(5.2.3-11)

5.2.3.10 Adaptive Codebook Memory Update

Update the adaptive codebook excitation memory with the combined excitation for the current subframe, $E_T(n)$, excluding any random excitations added in 5.2.3.10:

$$E_T(n) = \begin{cases} E(n+L); & -128 \leq n < -L \\ E_T(n+L); & -L \leq n < 0. \end{cases} \quad . \tag{5.2.3-12}$$

If ($FER\_FLAG(m) = FALSE$), and the current subframe, $m' = 2$, update the last valid excitation memory buffer, $E_{lv}(n)$ with the adaptive codebook memory:

$$E_{lv}(n) = E(n-128); \quad 0 \leq n < 128 . \tag{5.2.3-13}$$

5.2.3.11 Additional Excitation Frame Processing

A fade scaling variable, α_f, shall be calculated every subframe as defined by:

$$\alpha_f = \begin{cases} \alpha_f\text{-}0.05; & FER\_FLAG(m) = TRUE \\ \alpha_f\text{+}0.2; & otherwise, \end{cases} \tag{5.2.3-14}$$

where α_f is bounded by $0 \leq \alpha_f \leq 1$, and is used to scale the combined excitation:

$$E_T(n) = \alpha_f E_T(n); \quad 0 \leq n < L . \tag{5.2.3-15}$$

If ($FER\_FLAG(m) = TRUE$), or if the frame is a special packet-level signaling Half-Rate CELP frame for SO 68, and the average adaptive codebook gain, ($g_{pavg} < 0.4$), a random fixed codebook excitation is added to the combined excitation:

$$E_T(n) = E_T(n) + 0.1 g_{cavg} \text{ran\_g}\{seed\}; \quad 0 \leq n < L , \tag{5.2.3-16}$$

where g_{cavg} is calculated in 5.2.3.6 for the last valid frame, and ran_g{$seed$} is a unit variance pseudo-random Gaussian white noise sequence (see 4.7.2).

5.2.3.12 Frame Expansion or Compression in CELP Frames in SO 68

To interface to dynamic jitter buffer mechanisms used in Packet Switched networks, CELP decoder In SO 68 can output more than or less than 160 samples. When it is needed to output less than 160 samples, CELP decoder generates 160-DELAY samples of total excitation as described below.

$$E_T(n) = \begin{cases} \dfrac{(merge\_stop - n)}{merge\_stop} E_T(n) + \dfrac{n}{merge\_stop} E_T(n + merge\_end); & 0 \leq n < merge\_stop \\ E_T(n + merge\_end); & merge\_stop \leq n < 160 - merge\_end \end{cases}$$

where,

$merge\_end = DELAY$, and

$$merge\_stop = \begin{cases} merge\_end; & merge\_end \leq 80 \\ 160 - merge\_end; & merge\_end > 80 \end{cases}$$

When more than 160 samples are needed, CELP generates 160+DELAY samples of total excitation as described below.

1

$$E_T(n) = \begin{cases} E_T(n), & 0 \le n < \text{merge\_end} \\ \dfrac{(merge\_stop - n)}{merge\_stop} E_T(n) + \dfrac{n}{merge\_stop} E_T(n - merge\_end), & \text{merge\_end} \le n < merge\_end + merge\_stop \\ E_T(n - merge\_end), & \text{merge\_end} + \text{merge\_stop} \le n < 160 + \text{merge\_end} \end{cases}$$

2   where,

3    $merge\_end = DELAY$, and

4    $$merge\_stop = \begin{cases} merge\_end; & \text{merge\_end} \le 80 \\ 160 - merge\_end; & \text{merge\_end} > 80 \end{cases}$$

5.2.3.13 PPP Smoothing of CELP Frame After Erasure in SO 68

In order to smooth some phase discontinuities between a past erasure frame and a good CELP frame, PPP synthesis is employed to smooth such a transition.

Using a PPP DFS, $CURRP\_Q$, obtained from the last L (pitch lag of current frame) values of the current constructed total excitation, and the previous DFS, $PREV\_PPP$, obtained from the last pitch lag (of previous frame) of samples from excitation of the previous frame we generate the 160 samples of the current frame excitation, ET(m), similar to the method described in Section 4.12.30.4.

The above mentioned PPP smoothing is disabled if there is a significant change (either increase or decrease) between the energies of $CURRP\_Q$, and $PREV\_PPP$ referred to in this section.

5.2.3.14 Pitch Prefilter (SO 70 and SO 73 WB Mode)

The combined excitation vector, $E_T(n)$, for the current subframe, m' is filtered through a pitch enhancement filter prior to input to the LPC synthesis filter and high band synthesis processing. The pitch enhancement filter is a time varying filter of the form:

$$P(z,n) = \frac{1}{1 - 0.4 g_p z^{-\tau_c(n)}} ; \quad 0 \le n < L , \tag{5.2.3-17}$$

where $\tau_c(n)$ is the delay contour obtained from interpolating the delay adjusted endpoints, $\dot{d}(m',0)$ to $\dot{d}(m',1)$, as in 4.11.5.1. The filter uses an 8x oversampling fractional pitch interpolation similar to that used for generating the adaptive codebook contribution. This notation, however, is a mix of time domain and z-transform symbols, where the z^{-1} operator is to be viewed as a unit delay operator rather than a z-transform. The state of this filter is reset to all zeros on subframe $m' = 0$ when the previous frame was not RCELP Rate 1 or Rate 1/2, or if the previous frame was an erasure. In addition, the energy of the output of the pitch enhancement filter is normalized to the input energy. The output of the filter overwrites the previous values of $E_t(n)$, therefore it is critical to update the adaptive codebook memory prior to running this filter. This filter is not run if $FER\_FLAG(m) = TRUE$, and the state is reset to all zeros.

5.2.3.15 Synthesis of the Decoder Output Signal

Filter the combined excitation, ET(m), through the synthesis filter using the interpolated LPCs generated in Section 5.2.3.2, creating the synthesized speech signal, $\hat{s}(n)$. The synthesis filter for the decoder, Hq(z), is defined by:

$$H_q(z) = \frac{1}{A_q(z)} = \frac{1}{1 - \sum_{k=1}^{10} a_q(k) z^{-k}} . \tag{5.2.3-18}$$

The signal, $\hat{s}(n)$, shall be post-filtered according to Section 5.8 for SO 3 and Section 5.9 for SO 68, SO 70, and SO 73, producing the post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$.

5.2.3.16 Synthesis of the Decoder Output Signal

Filter the combined excitation, $E_t(m)$, through the synthesis filter using the interpolated LPCs generated in 5.2.3.2, creating the synthesized speech signal, $\hat{s}(n)$. The synthesis filter for the decoder, $H_q(z)$, is defined by:

$$H_q(z) = \frac{1}{A_q(z)} = \frac{1}{1 - \sum_{k=1}^{10} a_q(k)z^{-k}}. \tag{5.2.3-19}$$

The signal, $\hat{s}(n)$, shall be post-filtered according to 5.4, producing the post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$.

5.3 Rate 1 and Rate 1/4 PPP Decoding (FPPP and QPPP) for SO 68

5.3.1 Decoding of the LSP Parameters in FPPP

If (FER_FLAG(m) = TRUE), use the LSP parameters from the last frame to generate bandwidth expanded LSPs for the current frame as follows: $\Omega q(m) = 0.875\ \Omega q(m-1) + 0.125\ \Omega\text{spread}$ where Ωspread are the initial values of the LSPs, as defined in Section 5.2.

Otherwise, decode the LSP parameters, $\Omega q(m)$ using the decoded LSP indices, LSPIDX(k). In the event that the decoded LSPs are not strictly ascending, FER_FLAG(m) shall be set to TRUE, where "strictly ascending" is defined as $\Omega q(m, i) < \Omega q(m, i+1)$; $1 \le i \le 9$.

5.3.2 Decoding of the LSP Parameters in QPPP

A second order moving average (MA) predictor is used for quantizing the LSPs in Quarter-Rate voiced frame processing.

The MA predictive quantizer ensures that the effect of frame errors discontinues after 2 frames. The quantized LSP vector at the mth frame is given as

$$\hat{\mathbf{L}}_m = 0.4\mathbf{C}_m + 0.4\mathbf{C}_{m-1} + 0.2\mathbf{C}_{m-2} \tag{5.3.2-1}$$

where $\mathbf{C}_m$ is the codebook vector for the mth frame is constructed as follows:

$$C_m(i) = \begin{cases} CB1[i_1^*][i]; & 0 \le i < 6 \\ CB2[i_2^*][i-6]; & 6 \le i < 10 \end{cases}$$

where i_1^*, and i_2^* be the LSP codebook indices obtained from the QPPP packet for the two splits.

The LSP parameters, $\Omega q(m)$, are are elements of the vector L.

5.3.3 Delay Decoding in FPPP

The delay parameter, L, for the current frame is defined as:

$$L(m) = DELAY + 20 \tag{5.3.3-1}$$

where DELAY is the pitch delay transmission code.

5.3.4 Delay Decoding in QPPP

The delay parameter, L, for the current frame is defined as:

$$L(m) = DELTADELAY - 7 + PREV\_DELAY \tag{5.3.4-1}$$

where DELTADELAY is the pitch delay transmission code, and PREV_DELAY is the decoded delay parameter of the previous frame (same as L(m-1)).

5.3.5 Previous Frame Delay PPP (QPPP and FPPP)

The delay value, PL, of the previous frame is defined as:

$$PL = \begin{cases} L(m) & ; if\ previous\ frame\ was\ Rate\ 1/4\ unvoiced\ or\ Rate\ 1/8 \\ L(m-1) & ; otherwise \end{cases}$$

PL is further conditioned as follows

$$PL = \begin{cases} PL/2; & if\ PL > 1.85L \\ PL \cdot 2; & if\ PL \le 0.54L\ \ AND\ PL < 127/2 \\ PL; & otherwise \end{cases}$$

5.3.6 Interpolation of LSP Parameters and LSP to LPC Conversion

Interpolate the quantized LSPs over the three subframes, m′, in the current frame, m. The form of the quantized, interpolated LSP vector is:

$$\Omega_q(m') = (1 - \mu_{m'})\Omega_q(m-1) + \mu_{m'}\Omega_q(m), \tag{5.3.6-1}$$

where the subframe interpolator constants are defined as μ = { 0.1667, 0.5, 0.8333 }.

Convert the quantized, interpolated LSPs, $\Omega_q(m)$, to quantized, interpolated LPC parameters, $\left\{ \dot{a}_q \right\}$, as described in

Section 4.6.2.1, and Section 4.6.2.2 for each subframe.

The quantized interpolated LPC parameters at the end of the previous frame, and the current frames are used in the synthesis of the current frame of residual.

5.3.7 Restoring PPP Memories If Last Frame Is Non-PPP or Full-Rate PPP

The past-frame memory required for PPP processing is restored as was done in the encoder described in Section 4.12.25.

¹ **5.3.8 ERB Amplitude Dequantization of Full-Rate PPP**

² The ERB amplitudes are created from the Amplitude Quantization Indices as described in Section 4.12.29 to obtain the
³ current PPP DFS $CURRP\_Q_p$ in the polar form.

⁴ **5.3.9 Phase Dequantization of Full-Rate PPP or Special Half-Rate PPP**
⁵   **(Packet Level Signaling) Frame**

⁶ The phases of DFS, $CURRP\_Q_p$,is set to zero as follows:

⁷
$$CURRP\_Q_p.b(k) = 0; \qquad\qquad 0 \le k \le \lfloor L/2 \rfloor \tag{5.3.9-1}$$

⁸ $CURRP\_Q_p$, is converted to Cartesian representation as shown in Section 4.12.9, and then converted to the
⁹ perceptually weighted domain by pole-filtering using A_w , the weighted LPC filter, $A_w(k) = A(k)(0.8^k), \ 0 \le k < 9$.
¹⁰ Let's identify the resulting DFS' by $CURRP\_Q_S$.

¹¹ $CURRP\_Q_S$ is then shifted by the negative of the global alignment, *alignment* , obtained from the received packet as
¹² described in Section 4.12.6.

¹³ The band alignments from the received packets are then applied to $CURRP\_Q_S$, according to the following equations:

$$CURRP\_Q_S.a(k) = \left(\begin{array}{l} CURRP\_Q_S.a(k)\cos\left(-k\dfrac{2\pi}{L}\eta\left(-\dfrac{16}{\eta} + band\_alignment(j) \right) \right) - \\[2ex] CURRP\_Q_S.b(k)\sin\left(-k\dfrac{2\pi}{L}\eta\left(-\dfrac{16}{\eta} + band\_alignment(j) \right) \right) \end{array} \right)$$

$$CURRP\_Q_S s.b(k) = \left(\begin{array}{l} CURRP\_Q_S.a(k)\sin\left(-k\dfrac{2\pi}{L}\eta\left(-\dfrac{16}{\eta} + band\_alignment(j) \right) \right) + \\[2ex] CURRP\_Q_S.b(k)\cos\left(-k\dfrac{2\pi}{L}\eta\left(-\dfrac{16}{\eta} + band\_alignment(j) \right) \right) \end{array} \right) ;$$

$$0 \le k \le \left\lfloor \frac{L}{2} \right\rfloor$$

$$0 \le j < 17$$
$$-\frac{16}{\eta} \le n < \frac{16}{\eta}; n \ increasing \ in \ steps \ of \ 1$$
$$for \ every \ j,$$
$$k \in \left[0, \left\lfloor \frac{L}{2} \right\rfloor \right] \ such \ that \ lband(j) \le k\frac{8000}{L} < hband(j)$$

$$where, \eta = \begin{cases} 1; & j < 3 \\ 0.5; & 3 \le j < 17 \end{cases}$$
$$and, lband(j) = F\_BAND[j]$$
$$hband(j) = F\_BAND[j+1]$$

¹⁴

¹⁵ If the frame is a special HPPP frame, the band alignments used in the above equation are not present in the input packet
¹⁶ but instead are given by:

$$band\_alignment(j) = \begin{cases} 16; & j < 3 \\ 32; & otherwise \end{cases}$$

$CURRP\_Q_S$ is the then zero filtered using the procedure in Section 4.12.5, and A_w, the weighted LPC filter,

$A_w(k) = A(k)(0.8^k)$, $0 \le k < 9$ to obtain the quantized Full-Rate PPP DFS, $CURRP\_Q$

5.3.10 Obtaining the Dequantized Current DFS in QPPP

The ERB amplitudes of the QPPP frames are decoded as described in Section 4.12.28.3. The amplitude dequantized current PPP DFS, $CURRP\_Q_p$, is then obtained as described in Section 4.12.28.5 $CURRP\_Q_p$.

5.3.11 Copying the Phase Spectrum of Previous PPP DFS

In Quarter-Rate PPP, the phase of the current DFS is set to be equal to the phase of the previous DFS after it is extended into the current frame. That is, the phase of $CURRP\_Q_p$ (which is in polar form) is obtained as follows:

$$CURRP\_Q_p.b(k) = ACB_p.b(k); \qquad 0 \le k \le \lfloor L/2 \rfloor \qquad (5.3.11\text{-}1)$$

where ACB_p is the polar representation of ACB.

The Cartesian representation of yields the DFS of the current PPP, $CURRP\_Q$, that is to be used in the synthesis of the whole frame of excitation.

5.3.12 PPP Synthesis

Now that we have the quantized current PPP DFS, $CURRP\_Q$, and the previous DFS, $PREV\_PPP$, we generate the 160 samples of the current frame excitation, ET(m), similar to the method described in Section 4.12.30.4.

5.3.12.1 Frame Expansion or Compression in PPP Synthesis

To interface to dynamic jitter buffer mechanisms used in Packet Switched networks, Rate 1/4 PPP and Rate 1 PPP synthesis can output more than or less than 160 samples. When it is needed to output less than 160 samples, PPP generates 160-L samples. When more than 160 samples are needed, PPP generates 160+L samples.

5.3.13 Synthesis of the Decoder Output Signal in PPP

Filter the excitation, ET(m), through the synthesis filter using the interpolated LPCs generated in 5.2.3.2, creating the synthesized speech signal, $\hat{s}(n)$. The synthesis filter for the decoder, Hq(z), is defined by:

$$H_q(z) = \frac{1}{A_q(z)} = \frac{1}{1 - \sum_{k=1}^{10} a_q(k) z^{-k}}.$$

$$(5.3.13\text{-}1)$$

The signal, $\hat{s}(n)$, shall be post-filtered according to Section 5.8, producing the post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$.

5.4 Decoding at Rate 1 Generic Audio for SO 70 and SO 73

Inputs:

1 • The quantized LSP indices, *LSPIDX*(*k*)

2 • The quantized MDCT coefficient (residual line spectrum) index, *RLSIDX*

3 • The quantized global scale factor index, *GSFIDX*

4 • The quantized Noise Injection level indices, *NILIDX*

5 **Output:** The residual excitation signal, $\varepsilon'_q(n)$

6 **Initialization**: The following state variables shall be set to zero at initialization (frame *m* = 0):

7 • The formant emphasis and de-emphasis filter memories

8 • The IMDCT overlap-and-add buffer, $y_{ov}(n)$

9 **Processing**: Figure 5.4-1 depicts the Generic Audio decoding system that is described in the following sections. This
10 output signal $\varepsilon'_q(n)$ is substituted for the decoded RCELP Rate 1 and Rate 1/2 excitation signal $E_T(n)$ in 5.2.3.9. All
11 remaining processing follows from that point.

12
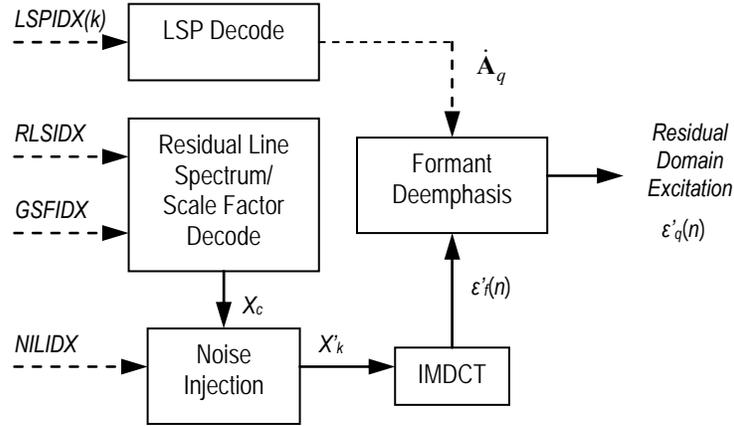
15 **Figure 5.4-1 Rate 1 Generic Audio Decoding Block Diagram**

16

17 5.4.1 MDCT Residual Line Spectrum Decoding

18 The MDCT coefficients, refered to as the residual line spectrum, are decoded from *RLSIDX* in a similar manner to the
19 FCB factorial codebook of 4.11.8.3. The decoded residual line spectrum is designated as X_{cc}.

20 5.4.2 Global Scale Factor Decoding

21 The decoded global scale factor is given as:

22 $$e_q = e_{q\min} + e_{bias} + GSFIDX \cdot \Delta_{eq},$$ (5.4.2-1)

23 where the terms are described in 0. The scale factor is then applied to the decoded residual line spectrum X_{cc} to obtain
24 the scaled residual line spectrum X_c, given as:

$$X_c = 10^{-e_q/20} X_{cc} .$$ (5.4.2-2)

5.4.3 Noise Injection Decoding

The noise injection decoding process parallels the encoding processes described in 4.13.7. A zero detection mask is first produced to control whether or not regions of the scaled line spectrum shall be injected with noise. The mask is represented as a 160 element column vector $\mathbf{z}_d$ that is defined as:

$$z_d(k) = \begin{cases} 1; & 40 \le k \le 143 \; and \; X_c(k) == 0 \\ 0; & otherwise \end{cases} .$$ (5.4.3-1)

The noise injection gain is then decoded using a lookup table, so that:

$$\gamma_{ni} = \gamma_{ni\_table}[NILIDX] ,$$ (5.4.3-2)

and subsequently used to generate the noise injected residual line spectrum X'_k as:

$$X'_k = \begin{cases} \gamma_{ni} 10^{-e_q/20} rand(); & 40 \le k \le 143 \; and \; z_d(k) == 1 \\ X_c(k); & otherwise \end{cases} ,$$ (5.4.3-3)

where all terms are previously defined.

5.4.4 Inverse Modified Discrete Cosine Transform (IMDCT)

The process of 4.13.8 is duplicated here to produce $\varepsilon'_f(n)$.

5.4.5 Formant Deemphasis

The process of 4.13.9 is duplicated here to produce $\varepsilon'_q(n)$. This signal is then substituted for the RCELP Rate 1 and Rate 1/2 excitation signal $E_T(n)$ in 5.2.3.9. All remaining processing follows from that point.

5.5 Decoding of Rate 1/4 Unvoiced Decoding or Special Rate 1/2 NELP Decoding SO 68 and Rate 1/2 Unvoiced Decoding SO 70 and SO 73

The decoding of Rate 1/4 unvoiced or the Special Rate 1/2 NELP frame is depicted in Figure 5.5-1.
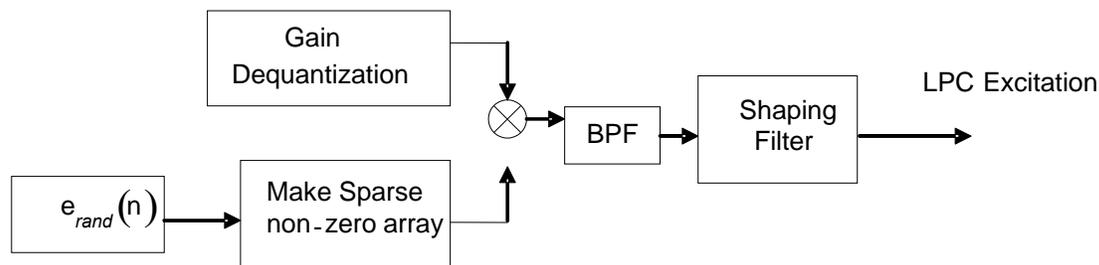
Figure 5.5-1 Excitation Decoding for Rate 1/4 NELP or Rate 1/2 Special NELP

The gain indices $i\hat{G}'$ and $i\hat{G}''[j], 0 \le j < 2$, and the shaping filter index F_{idx} are decoded from the bitstream input to the decoder. The gains are dequantized using Equations (4.14.1-5) and (4.14.1-8). A pseudo-random vector, $e_{rand}(n)$, is

generated as described in Section 4.14.2. As described in Section 4.14.3, this pseudo-random vector is used to generate a gain scaled sparse signal, $r_1(n)$, which is then shaped by the bandpass filter $BP(z)$, and the shaping filter $SF(z)$ to yield the Rate 1/4 frame LPC excitation. As shown in Equation (4.14.4-9), the index of the filter, F_{idx}, is used to establish $SF(z)$. The Rate 1/4 unvoiced frame excitation is then filtered by the LP synthesis filter and the post-processing filter to generate speech.

5.5.1 Frame Expansion or Compression in Rate 1/4 NELP Decoding

To interface to dynamic jitter buffer mechanisms used in Packet Switched networks, Rate 1/4 NELP frames can output more than or less than 160 samples. When it is needed to output less than 160 samples, Rate 1/4 NELP puts out 128 samples. When more than 160 samples are needed, Rate 1/4 NELP puts out 192 samples.

For compression, the number of subframes of Rate 1/4 NELP used are 8. Specifically, only first 8 gain values are used in the excitation generation.

For expansion, the number of subframes of Rate 1/4 NELP used is 12. Specifically, the two extra gain values used in the excitation generation are set to the last subframe's gain value in the decoded packet. That is, $\hat{G}(10) = \hat{G}(11) = \hat{G}(9)$.

5.6 Rate 1/8 Decoding **for SO 3**

Inputs: The inputs to Rate 1/8 decoding are:

- The quantized LSP indices from the current frame, $LSPIDX(k)$
- The quantized LSPs from the previous frame, $\Omega_q(m\text{-}1)$
- The vector quantized frame energy index, $FGIDX(m)$
- The vector quantized frame energy index from the last frame, $FGIDX(m - 1)$
- The frame erasure flag, $FER\_FLAG(m)$, for the current frame m

Outputs: The outputs from Rate 1/8 decoding are:

- The post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$

State Variables: None.

Processing: Rate 1/8 decoding shall comprise 5.6.1 to 5.6.3.

5.6.1 Decoding of LSP Parameters

If ($FER\_FLAG(m) = TRUE$), use the LSP parameters from the last frame, $\Omega_q(m) = \Omega_q(m\text{-}1)$. Otherwise, decode the LSP parameters, $\Omega_q(m)$, as defined in 4.9.5, using the decoded $LSPIDX(k)$. In the event that the decoded LSPs are not strictly ascending, $FER\_FLAG(m)$ shall be set to $TRUE$, where "strictly ascending" is defined as $\Omega_q(m, i) < \Omega_q(m, i+1)$;$1 \le i \le 9$.

5.6.2 Decoding of the Frame Energy Vector

If ($FER\_FLAG(m) = FALSE$), the frame energy vector, $\gamma_q(m')$, shall be defined as:

$$\gamma_q(m') = 10^{q\log(m', FGIDX(m))}; \quad 0 \le m' < 3,$$

(5.6.2-1)

1   where $q_{log}(m', FGIDX)$ is an entry in the gain quantization codebook found in Table 8-18.

2   If ($FER\_FLAG(m) = TRUE$), the frame energy vector, $\gamma_q(m')$, is calculated by:

3
$$\gamma_q(m') = \frac{1}{3}\sum_{i=0}^{2} 10^{q\log(i,FGIDX(m-1))}; \quad 0 \le m' < 3, \tag{5.6.2-2}$$

4   where $FGIDX(m\text{-}1)$ is the codebook index from the last Rate 1/8 packet in which FER_FLAG was $FALSE$.

5.6.3 Rate 1/8 Subframe Decoding

6   Compute the decoded synthesized speech signal for each subframe, $0 \le m' < 3$, as described in 5.6.3.1 to 5.6.3.4. The

7   subframe size, L, is 53 for subframes 0 and 1, and 54 for subframe 2.

5.6.3.1 Rate 1/8 Excitation Generation

9   Obtain the Rate 1/8 excitation by generating a zero-mean, unit variance pseudo-random Gaussian white noise process

10 (see 4.16.2), which is conditionally scaled by $\gamma_q(m')$:

11
$$E(n) = \gamma_q(m')\text{ran\_g}\{seed\}; \quad 0 \le n < L, \tag{5.6.3-1}$$

12 No attempt is made to synchronize the random time series generated at the receiver with that generated at the transmitter.

5.6.3.2 Interpolation of LSP Parameters

14 Interpolate the quantized LSPs as in 5.2.3.1.

5.6.3.3 LSP to LPC Conversion

16 Perform the LSP to LPC conversion as defined in 5.2.3.2.

5.6.3.4 Synthesis of Decoder Output Signal

18 Filter the Rate 1/8 excitation, E(n), through the synthesis filter as in 5.2.3.15 ~~5.2.3.11~~. The signal, $\hat{s}(n)$, shall be post-

19 filtered according to 5.8 ~~5.4~~, producing the post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$.

5.7 Rate 1/8 Decoding for SO 68, SO 70, and SO 73

5.7.1 DTX Off Mode: SO 68 and SO 70

22 **Inputs:** The inputs to Rate 1/8 decoding are:

23 • The quantized LSP indices from the current frame, $LSPIDX(k)$

24 • The quantized LSPs from the previous frame, $\Omega_q(m\text{-}1)$

25 • The vector quantized frame energy index, $FGIDX(m)$

26 • The vector quantized frame energy index from the last frame, $FGIDX(m\text{-}1)$

27 • The frame erasure flag, $FER\_FLAG(m)$, for the current frame m

28 **Outputs:** The outputs from Rate 1/8 decoding are:

1 • The post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$

2 **State Variables:** None.

3 **Processing:** Rate 1/8 decoding shall comprise 5.7.1.1 to 5.7.1.4.

5.7.1.1 Decoding of the LSP Parameters

5 If ($FER\_FLAG(m) = TRUE$), use the LSP parameters from the last frame, $\Omega_q(m) = \Omega_q(m\text{-}1)$. Otherwise, decode the LSP
6 parameters, $\Omega_q(m)$, as defined in 4.9.5, using the decoded $LSPIDX(k)$. In the event that the decoded LSPs are not strictly
7 ascending, $FER\_FLAG(m)$ shall be set to $TRUE$, where "strictly ascending" is defined as $\Omega_q(m, i) < \Omega_q(m, i+1)$
8 $;1 \le i \le 9$.

5.7.1.2 Decoding of the Frame Energy Gain for SO 68 and , SO 70, and SO 73 NB
10 **Mode**

11 If (FER_FLAG(m) = FALSE), the frame energy vector, γq(m′), is defined as in Section 4.15.8 4.15.6.

12 If (FER_FLAG(m) = TRUE), the frame energy vector, γq(m′), is calculated by:

$$\gamma_q(m') = 10^{\left(\frac{LG0+LG1+LG2}{3}\right)}$$

13 (5.7.1-1)

14 where,

$$LG2 = \sqrt{\frac{\sum_{j=-128}^{-2\times54}(E_T(j))^2}{128 - 2\times54}}$$

$$LG1 = \sqrt{\frac{\sum_{j=-54\times2-1}^{-54}(E_T(j))^2}{54}}$$

$$LG0 = \sqrt{\frac{\sum_{j=-53}^{0}(E_T(j))^2}{54}}$$

18 where E_T is the adaptive codebook vector mentioned in Section 5.2.3.10.

5.7.1.3 Decoding of the Frame Energy Gain for SO 70 and SO 73 WB Mode

20 If (FER_FLAG(m) = FALSE), the frame energy, $\gamma_q(m)$, is defined as:

$$\gamma_q(m) = \begin{cases} \exp(LOGZEROENERGY_{wb}) & ; FGIDX == 0 \\ \exp\left(MINLOGE_{wb} + FGIDX \cdot \left(\frac{MAXLOGE_{wb} - MINLOGE_{wb}}{NUMQLEVELS_{wb}}\right)\right) & ; otherwise \end{cases} ,$$

21 (5.7.1-2)

22 where $LOGZEROENERGY_{wb}$ = -2. If (FER_FLAG(m) = TRUE), the frame energy, $\gamma_q(m)$, is simply repeated from the
23 previous Rate 1/8 frame, or $\gamma_q(m) = \gamma_q(m\text{-}1)$. The frame energy is then smoothed and attenuated according to:

$$\tilde{\gamma}_q(m) = \begin{cases} \gamma_q(m) & ; m' == 0 \ and \ \left((m-k > 6) \ or \ (m < 6) \ or \ (\tilde{\gamma}_q(k) > \gamma_q(m))\right) \\ 0.97\tilde{\gamma}_q(k) + 0.0048\gamma_q(m) & ; otherwise \end{cases}$$

, (5.7.1-3)

where k is the frame index of the last Eighth-Rate frame and $\tilde{\gamma}_q(k)$ is the smoothed energy calculated during the last valid Eighth-Rate frame.

5.7.1.4 Rate 1/8 Subframe Decoding

Compute the decoded synthesized speech signal for each subframe, $0 \le m' < 3$, as described in 5.7.1.4.1 to 5.7.1.4.5. The subframe size, L, is 53 for subframes 0 and 1, and 54 for subframe 2.

5.7.1.4.1 Rate 1/8 Excitation Generation

Obtain the Rate 1/8 excitation by generating a zero-mean, unit variance pseudo-random Gaussian white noise process (see 4.16.2), which is conditionally scaled by $\gamma_q(m')$:

$$E(n) = \gamma_q(m')\mathrm{ran\_g}\{seed\}; \quad 0 \le n < \mathrm{L},$$

(5.7.1-4)

No attempt is made to synchronize the random time series generated at the receiver with that generated at the transmitter.

5.7.1.4.2 Interpolation of LSP

Interpolate the quantized LSPs as in 5.2.3.1. For SO 70 and SO 73 WB mode, the LSPs are further smoothed according to:

$$\tilde{\Omega}(m',i) = 0.9\tilde{\Omega}(m',i) + 0.1\Omega(m',i); \quad 0 \le i \le 9,$$

(5.7.1-5)

where the smoothed LSP are initialized at frame $m = 0$ to be $\tilde{\Omega}(m') = \Omega(m')$, and the smoothed LSPs are then used to replace the current set of unsmoothed LSPs, or $\Omega(m') \leftarrow \tilde{\Omega}(m')$.

5.7.1.4.3 LSP to LPC Conversion

Perform the LSP to LPC conversion as defined in 5.2.3.2.

5.7.1.4.4 Estimation of the LPC Prediction Gain (for SO 70 ~~and SO 73~~ WB Mode)

Estimate the prediction gain of the smoothed LPC filter by calculating the energy of the filter impulse response $\tilde{h}(n)$, and then apply the inverse of the gain to the Rate 1/8 excitation:

$$E(n) \leftarrow \sqrt{\dfrac{\tilde{\gamma}_q(m)}{\sum_{i=0}^{L-1}\left(\tilde{h}(i)\right)^2}}E(n); \quad 0 \le n < L,$$

(5.7.1-6)

The excitation is now properly gain scaled such that the smoothed frame energy is obtained after application of the subsequent LPC synthesis filter in the next section.

5.7.1.4.5 Synthesis of Decoder Output Signal

Filter the Rate 1/8 excitation, E(n), through the synthesis filter as in 5.2.3.15 ~~5.2.3.11~~. The signal, $\hat{s}(n)$, shall be post-filtered according to 5.9 ~~5.4~~, producing the post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$.

5.7.2 DTX On Mode: SO 68, SO 70, and SO 73

In DTX on mode, there are two mutually exclusive mechanisms which are controlled at compilation time. Both mechanisms generate comfort noise for *FER_FLAG(m) = TRUE* packets with different approaches. These two mechanisms are described in 5.7.2.1 and 5.7.2.2, respectively. Mechanism II is the default mechanism.

5.7.2.1 Mechanism I

If *FER_FLAG(m) = FALSE*, then the decoding process is identical to that described in 5.7.1.

If *FER_FLAG(m) = TRUE*, then the decoding process follows the followings:

Inputs: The frame erasure flag, FER_FLAG(m), for the current frame m

Outputs: The post-filtered synthesized speech signal, $\hat{s}_{pf}(n)$.

State Variables: The background noise estimate, $E_{bgn}(n)$

Processing: Suppressed Rate 1/8 decoding shall comprise Sections 5.7.2.1.1 and 5.7.2.1.2 . The basic structure of the decoder is modified as shown in Figure 5.7-1 to incorporate decoding of suppressed rate 1/8 frames. The received packet is decoded as defined in the previous sections when the rate is not blank or suppressed rate 1/8. When the received rate is a suppressed rate 1/8 frame, the switch connects the output of the comfort noise synthesizer and encoder to the decoder to produce the output signal.
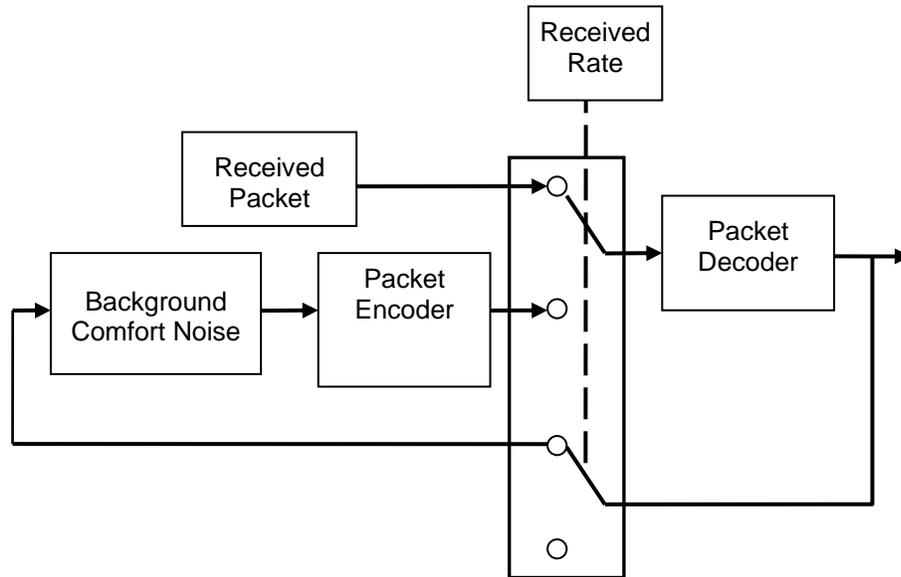
Figure 5.7-1 Comfort Noise Synthesizer Block Diagram

5.7.2.1.1 Synthesizing Comfort Noise

The comfort noise is synthesized from the background noise estimate $E_{bgn}(n)$, generated in 5.10.3. The frequency domain representation of the noise is computed from the background noise estimate. First, the background noise estimate is smoothed to produce

$$E_{bgnSm}(m,i) = \begin{cases} \beta_2 E_{bgnSm}(m-1,i) + (1-\beta_2)E_{bgn}(m,i) & ; m \geq 10 \\ \beta_{2r} E_{bgnSm}(m-1,i) + (1-\beta_{2r})E_{bgn}(m,i) & ; m < 10 \end{cases}$$

(5.7.2-1)

Next, the magnitude of noise is computed according to:

$$X_{decmag}(k) = 10^{E_{bgn}(m,i)/20} \quad ; f_L(i) \leq k \leq f_H(i), \ 0 \leq i < N_c$$

(5.7.2-2)

where $\beta_2 = 0.9925$, and $\beta_{2r} = 0.794$.

with $f_L(i)$ and $f_H(i)$, defined in Section 4.4.3.2, with $N_c = 16$. The phase is randomized to produce X_{dec}, according to:

$$\phi(k) = \cos(2\pi \cdot ran0\{seed\}) + j\sin(2\pi \cdot ran0\{seed\})$$

(5.7.2-3)

$$X_{dec}(k) = X_{decmag}(k) \cdot \phi(k)$$

(5.7.2-4)

where ran0 is a uniform pseudo-random number generator as described in 4.16.1. The noise is converted to the time domain using the inverse DFT:

$$x_{dec}(m,n) = \begin{cases} x_{dec}(m-1, n+L-D) + w_g(n) \cdot \dfrac{1}{2} \sum_{k=0}^{M-1} X_{dec}(k)e^{j2\pi nk/M} & ; 0 \leq n < D \\ w_g(n) \cdot \dfrac{1}{2} \sum_{k=0}^{M-1} X_{dec}(k)e^{j2\pi nk/M} & ; D \leq n < M. \end{cases}$$

(5.7.2-5)

with $w_g(n)$ defined as:

$$w_g(n) = \begin{cases} \sin^2(\pi(n+0.5)/2D) & ; 0 \leq n < D \\ 1 & ; D \leq n < L \\ \sin^2(\pi(n-L+D+0.5)/2D) & ; L \leq n < L+D \end{cases}$$

(5.7.2-6)

5.7.2.1.2 Generating Encoded Packet from Synthesized Noise

To better match the spectral characteristics of Rate 1/8 frames, the synthesized comfort noise $x_{dec}(m,n)$ is encoded as a Rate 1/8 frame. The enconding process is a simplified version of the one described in Section 4. First, the synthesized noise from Equation 5.7.1-5 is used to compute the LPC coefficients according to 4.6.1.1 by re-defining $s'(k)$ as:

$$s'(k) = \begin{cases} x_{dec}(m-1,k) & ; 0 \leq k < 160 \\ x_{dec}(m, k-160) & ; 160 \leq k < 320 \end{cases}$$

(5.7.2-7)

Then, the LPC coefficients are converted to LSPs according to 4.6.1.3. The LSPs are interpolated for each subframe and converted back to LPCs according to 4.6.2.1 and 4.6.2.2.

To compute the LPC gain, a short-term residual $\varepsilon(n)$ is generated according to Equation 4.6.2-10. Then, the impulse response h_{noise} of the inverse LPC filter for the current subframe m', is calculated to the subframe length L_{sf}. The frame energy gain is determined by:

$$\gamma(m') = \log 10 \left(\frac{\dfrac{2}{L_{sf}} \cdot \displaystyle\sum_{n=0}^{L_{sf}-1} |\varepsilon(n)|}{\sqrt{\displaystyle\sum_{n=0}^{L_{sf}-1} h_{noise}^2(n)}} \right)$$

$$(5.7.2-8)$$

The LSPs are vector quantized according to 4.15.1, and the frame energy gain according to 4.15.6. The encoded parameters are processed by 5.7.1.1 ~~5.6.1~~ through 5.7.1.4.5 ~~5.6.5.5~~ assuming FER_FLAG(m) is not set to produce the decoded output $\hat{s}(n)$.

5.7.2.2 Mechanism II

5.7.2.2.1 Narrowband Mode

If *FER_FLAG(m) = FALSE* and *FER_FLAG(m-1) = FALSE*, then the decoding process is identical to that described in 5.7.1. Otherwise, the decoding process follows the followings:

- If *FER_FLAG(m) = TRUE, FER_FLAG(m-1) = FALSE* and *Rate(m-1) = Rate 1/8*, and *FER_FLAG(m-2) = FALSE*, then the decoding of packet *m* is done by repeating the LSPs and frame energy gain parameters for packet *m-1*, i.e., $\Omega_q(m) = \Omega_q(m-1)$ and $\gamma_q(m) = \gamma_q(m-1)$.

- If *FER_FLAG(m) = TRUE, FER_FLAG(m-1) = FALSE* and *Rate(m-1) = Rate 1/8, FER_FLAG(m-2) =…= FER_FLAG(m-j-1) = TRUE*, and *FLAG(m-j) = FALSE* and *Rate(m-j) = Rate 1/8* with *j>2*, then the decoding of packet *m* is done by linear interpolation between the LSPs and frame energy gain parameters from packet *m-j* to *m-1*. The LSPs and frame energy gain from packet *m-j* is set to be the initial value and the LSPs and frame energy gain of packet *m-1* is set to be the final value in the linear interpolation. Linear interpolation is done over a 32 frame interval starting from the initial value to the final balue for both LSPs and frame energy gain to enable comfort noise generation for the consecutive bad (erased, suppressed, or blanked) packets which follow the current bad packet. If at any point in the middle of the linear interpolation (consecutive bad packets) another valid Rate 1/8 packet arrives, then the initial and final values of the linear interpolation are updated as follows:

  - Initial LSPs and frame energy gain = LSPs and frame energy gain of the previous bad packet,

  - Final LSPs and frame energy gain = LSPs and frame energy gain of the received Rate 1/8 packet.

  This ensures the background noise characteristics, even though not instantaneously tracked, is always smooth.

Refer to Section 5.7.1.4.1 for the description of Rate 1/8 excitation generation.

5.7.2.2.2 Wideband Mode

The silence coding for wideband mode has a LSPs and frame energy gain smoothing scheme even for regular eighth-rate frame decoding. As a result no additional LSPs and frame energy gain interpolation as discussed for the narrowband case is needed for WB mode.

If *FER_FLAG(m) = TRUE*, comfort noise generation for the *m* packet is done by simply repeating the LSPs and frame energy gain of the immediate last received valid Rate 1/8 until another valid Rate 1/8 frame is received. The LSPs and

frame energy gain smoothing which is done as a part of regular silence decoding in wideband mode ensures that the background noise characteristics, even though not instantaneously tracked, is always smooth. Refer to Sections 5.7.1.1 and 5.7.1.3 for the description of regular WB Rate 1/8 decoding. Refer to Section 5.7.1.4.1 for the description of Rate 1/8 excitation generation.

5.8 Adaptive Postfilter for SO 3

The Adaptive Postfilter function improves the perceived speech quality of the decoder output. Refer to 1.1 for guidelines concerning potential variations of the Adaptive Postfilter.

Inputs:

- The decoder synthesis output signal, $\hat{s}(n)$

- The quantized, interpolated LPC coefficients, $\dot{A}_q(n)$

- The interpolated pitch delays, $\dot{d}(m')$

Output: The post-filtered synthesis signal, $\hat{s}_{pf}(n)$

Initialization: All filter memories shall be initialized to zero at start-up.

Processing: The decoded speech signal, $\hat{s}(n)$, shall be post-filtered by applying the following operations in the sequence specified:

- A tilt-compensation filter $H_t(z)$

- A short term residual filter, $H_\varepsilon(z)$

- A long term filter, $H_p(z)$

- Gain normalization, g_s

- A short term filter, $H_s(z)$

The postfilter shall be applied once for each of the three subframes and shall take the form:

$$PF(z) = H_S(z)g_S H_p(z) H_\varepsilon(z) H_t(z).$$ (5.8-1)

The postfilter residual memory buffer is then updated with the current subframe's residual signal. Each of these steps shall be carried out as specified in Sections Tilt Compensation Filter 5.8.1 to 5.8.4.

5.8.1 Tilt Compensation Filter

Generate the tilt-compensated speech signal, $\hat{s}_t(n)$, by filtering the decoded speech signal, $\hat{s}(n)$, through the tilt compensation filter given by:

$$H_t(z) = 1 - \mu z^{-1},$$ (5.8.1-1)

where μ is the tilt coefficient calculated by determining if $\hat{s}(n)$ is voiced or un-voiced. The signal, $\hat{s}(n)$, is defined as being voiced if:

$$R = \left(\sum_{i=0}^{L-2} \hat{s}(i)\hat{s}(i+1) \right) \geq 0 .$$ (5.8.1-2)

Using the above result, the tilt coefficient, μ, is defined as:

$$\mu = \begin{cases} 0 & ; R < 0, \\ tilt \text{ (Rate)} & ; R \geq 0, \end{cases} \tag{5.8.1-3}$$

where $tilt(Rate)$ is defined in Table 5.8.1-1.

Table 5.8.1-1 Postfilter Coefficients

Rate	tilt(Rate)	γ_{p1}	γ_{p2}
1	0.20	0.57	0.75
1/2	0.35	0.50	0.75
1/8	0	0.57	0.57

5.8.2 The Short Term Residual Filter

Compute the postfilter residual, $\varepsilon_{pf}(n)$, by filtering the tilted speech signal, $\hat{s}_t(n)$, through $H_\varepsilon(z)$, given by:

$$H_\varepsilon(z) = \dot{A}_q\left(\gamma_{p1}^{-1}z\right) = 1 - \sum_{k-1}^{10} \dot{a}_q(k)\gamma_{p1}^k z^{-k}, \tag{5.8.2-1}$$

where γ_{p1} is given for each *Rate* in Table 5.8.1-1, and $\left\{\dot{a}_q\right\}$ are the interpolated LPC parameters for the current subframe.

5.8.3 The Long-Term Postfilter

Generate the long-term post-filtered signal, $\hat{s}_p(n)$, by filtering the residual signal, $\varepsilon_{pf}(n)$, through the long term postfilter given by:

$$H_p(z) = \begin{cases} 1.0 + g_p g_{lt} z^{-d_{opt}} & ; 0.5 \leq g_p < 1.0, \\ 1.0 + g_{lt} z^{-d_{opt}} & ; 1.0 \leq g_p, \\ 1.0 & ; \text{otherwise}, \end{cases} \tag{5.8.3-1}$$

where $g_{lt} = 0.5$ and where the long term predication gain, g_p, is calculated by:

$$g_p = \frac{\displaystyle\sum_{n=0}^{L-1} \varepsilon_{pf}(n)\varepsilon_{pf}(n-d_{opt})}{\displaystyle\sum_{n=0}^{L-1} \varepsilon_{pf}(n-d_{opt})\varepsilon_{pf}(n-d_{opt})}, \tag{5.8.3-2}$$

and where the long-term delay, d_{opt}, is calculated from $\varepsilon_{pf}(n)$ by finding the best integer delay around $d_I = \left(\dfrac{\dot{d}(m',0) + \dot{d}(m',1)}{2}\right)$. The decoded interpolated integer delay d_I is searched from $(d_I - 3)$ to $(d_I + 3)$ to find the best integer delay for the long-term postfilter. The best integer delay d_{opt} is computed by maximizing the correlation:

$$R(d_{opt}) = \sum_{n=0}^{L-1} \varepsilon_{pf}(n)\varepsilon_{pf}(n-d_{opt}); \quad (d_I - 3) \le d_{opt} \le (d_I + 3). \tag{5.8.3-3}$$

5.8.4 Gain Normalization and Short-Term Postfilter

Compute the temporary signal, $\hat{s}_s(n)$, by filtering the long-term postfilter output signal, $\hat{s}_p(n)$, through the short-term postfilter, $H_s(z)$, given by:

$$H_s(z) = \frac{1}{A_q\left(\gamma_{p2}^{-1}z\right)}, \tag{5.8.4-1}$$

where γ_{p2} is found in Table 5.8.1-1.

Then compute the short-term postfilter gain, g_s, by using the temporary signal, $\hat{s}_s(n)$:

$$g_s = \sqrt{\frac{\sum_{n=0}^{L-1} \hat{s}^2(n)}{\sum_{n=0}^{L-1} \hat{s}_s^{\,2}(n)}}, \tag{5.8.4-2}$$

where g_s is upper bound limited at 1.0. The signal $g_s\hat{s}_p(n)$ is then filtered through the short-term postfilter in Equation 5.8.4-1 to produce the final post-filtered speech signal, $\hat{s}_{pf}(n)$.

5.9 Adaptive Postfilter for SO 68, SO 70, and SO 73

The adaptive postfilter function improves the perceived speech quality of the decoder output.

Inputs:

- The decoder synthesis output signal, $\hat{s}(n)$

- The quantized, interpolated LPC coefficients, $\dot{A}_q(n)$

- The quantized, interpolated LSPs, $\Omega_q(m,k); 1 \le k \le 10$

- The interpolated pitch delays, $\dot{d}(m')$

Output: The post-filtered synthesis signal, $\hat{s}_{pf}(n)$.

Initialization: All filter memories shall be initialized to zero at start-up.

Processing: The decoded speech signal, $\hat{s}(n)$, shall be post-filtered by applying the following operations.

A reference set of LPCs is first obtained from the decoded speech signal as follows:

$$A_{ref}(k) = \begin{cases} -\dfrac{\sum_{n=0}^{L-2} \hat{s}(n)\hat{s}(n+1)}{\sum_{n=0}^{L-1} \hat{s}(n)\hat{s}(n)}; & k = 0 \\ \\ 0; & otherwise \end{cases}$$

A_{ref} is then converted to a reference LSPs, Ω_{ref} as described in Section 4.6.1.3.

Two sets of interpolated LSPs, Ω_3, and Ω_4, are derived from Ω_{ref}, and the input interpolated quantized LSPs, Ω_m as follows:

$$\Omega_3(i) = \begin{cases} (0.3 + 0.04i)\Omega_{ref}(i) + (0.7 - 0.04i)\Omega_m(i); & 0 \le i < 5 \\ (0.5)\Omega_{ref}(i) + (0.5)\Omega_m(i); & 5 \le i < 10 \end{cases}$$

$$\Omega_4(i) = (0.3)\Omega_{ref}(i) + (0.7)\Omega_m(i); \qquad 0 \le i < 10$$

These interpolated LSPs are then converted to interpolated LPCs A_3 and A_4 as described in Section 4.6.2.2.

The decoded speech signal, $\hat{s}(n)$, shall be post-filtered by applying the following operations in the sequence specified:

1. A short term residual filter, $H_\varepsilon(z)$

2. A long term filter, $H_p(z)$

3. Gain normalization, g_s

4. A short term filter, $H_s(z)$

The postfilter shall be applied once for each of the three subframes and shall take the form:

$$PF(z) = H_S(z)g_S H_P(z)H_\varepsilon(z) \qquad \text{(5.9-1)}$$

The postfilter residual memory buffer is then updated with the current subframe's residual signal. Each of these steps shall be carried out as specified in Sections 5.8.1 to 5.8.5.

5.9.1 The Short-Term Residual Filter

Compute the postfilter residual, εpf(n), by filtering the tilted speech signal, $\hat{s}_t(n)$, through Hε (z) which is equal to $A_3(z)$.

5.9.2 The Long-Term Postfilter

Generate the long-term post-filtered signal, $\hat{s}_p(n)$, by filtering the residual signal, εpf(n), through the long term postfilter given by:

$$H_p(z) = \begin{cases} 1.0 + g_p g_{lt} z^{-d_{opt}} & ; 0.5 \le g_p < 1.0, \\ 1.0 + g_{lt} z^{-d_{opt}} & ; 1.0 \le g_p, \\ 1.0 & ; \text{otherwise}, \end{cases}$$

$$\text{(5.9.2-1)}$$

where glt = 0.4 and where the long term predication gain, gp, is calculated by:

$$g_p = \frac{\sum_{n=0}^{L-1} \varepsilon_{pf}(n)\varepsilon_{pf}(n - d_{opt})}{\sum_{n=0}^{L-1} \varepsilon_{pf}(n - d_{opt})\varepsilon_{pf}(n - d_{opt})} , \qquad \text{(5.9.2-2)}$$

and where the long-term delay, d_{opt}, is calculated from $\varepsilon_{pf}(n)$ by finding the best integer delay

around $d_I = \left(\dfrac{\dot{d}(m',0) + \dot{d}(m',1)}{2} \right)$. The decoded interpolated integer delay d_I is searched from $(d_I - 3)$ to $(d_I + 3)$

to find the best integer delay for the long-term postfilter. The best integer delay d_{opt} is computed by maximizing the correlation:

$$R(d_{opt}) = \sum_{n=0}^{L-1} \varepsilon_{pf}(n)\varepsilon_{pf}(n - d_{opt}); \qquad (\text{dI} - 3) \leq d_{opt} \leq (\text{dI} + 3). \tag{5.9.2-3}$$

5.9.3 Gain Normalization and Short-Term Postfilter

Compute the temporary signal, $\hat{s}_s(n)$, by filtering the long-term postfilter output signal, $\hat{s}_p(n)$, through the short-term

postfilter, Hs(z), which is same as $A_4(z)$.

Then compute the short-term postfilter gain, gs, by using the temporary signal, $\hat{s}_s(n)$:

$$g_s = \sqrt{\dfrac{\sum\limits_{n=0}^{L-1} \hat{s}^2(n)}{\sum\limits_{n=0}^{L-1} \hat{s}_s^2(n)}}, \tag{5.9.3-1}$$

where gs is upper bound limited at 1.0. The signal gs $\hat{s}_s(n)$ is then filtered through the short-term postfilter Hs(z), to produce the final post-filtered speech signal, $\hat{s}_{pf}(n)$.

5.10 Background Noise Estimation (for SO 68)

To allow suppressed Rate 1/8 frames, the output of the adaptive postfilter is used to determine an estimated background noise spectrum.

Inputs:

- The post-filtered synthesis signal, $\hat{s}_{pf}(n)$

- The current frame rate, Rate

- Background noise estimate adaptation factor, w_α

Outputs: None. Only state variables affected.

State Variables: The background noise envelope estimate, $\mathbf{E}_{bgn}$.

Initialization: The following variables shall be set to zero at initialization (frame m=0):

- The overlapped portion of the input buffer (output of the postfilter), {}

The following shall be initialized to a startup value other than zero:

- The channel noise estimate, $E_{bgn}(0)$

1 • The channel energy leaky integrator's memory, $E_{Nch}(0)$

2 **Processing:** The background noise spectral estimate is only computed on frames where FER_FLAG(m) is not set and

3 Rate is not equal to Suppressed Rate 1/8. The post-filtered speech signal, $\hat{s}_{pf}(n)$, is used to determine the background

4 noise spectral estimate at the decoder. Although the frame size of the decoder is 20 ms, the background noise spectral

5 estimator uses procedures from the noise suppressor and therefore operates in blocks of 10 ms. The following

6 procedures shall be executed two times per 20 ms of decoded speech frame, and the current 10ms frame for the

7 background noise estimator shall be denoted m. Figure 5.10-1 depicts the Background Noise Estimator that is described

8 in the following sections.

9

10
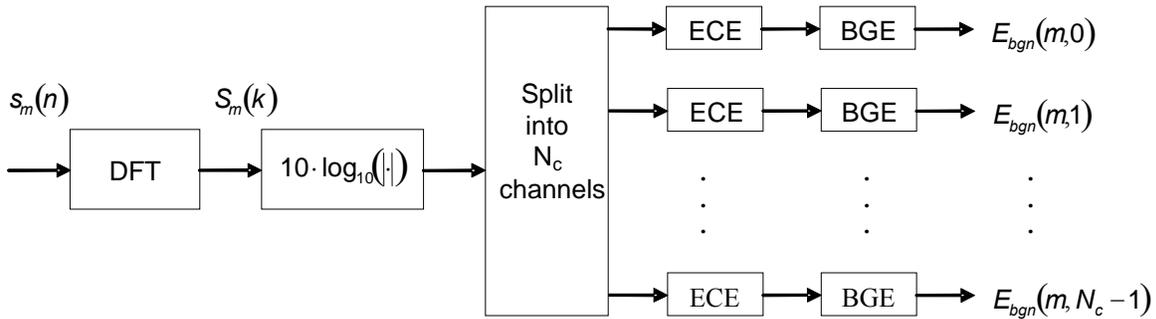
11 **Figure 5.10-1 Background Noise Estimator Block Diagram**

12 ## 5.10.1 Frequency Domain Conversion

13 The post-filtered signal $\hat{s}_{pf}(m,n)$ is windowed using a smoothed trapezoid window, in which the first D samples of the

14 noise estimator's input frame buffer $s_m(n)$ are obtained by windowing the last D samples of the previous frame

15 ($\hat{s}_{pf}(m-1,n)$). $s_m(n)$ is obtained by:

$$s_m(n) = \begin{cases} \hat{s}_{pf}(m-1, L-D+n) \cdot \sin^2(\pi(n+0.5)/2D) & ; 0 \le n < D, \\ \hat{s}_{pf}(m, n-D) & ; D \le n < L, \\ \hat{s}_{pf}(m, n-D) \cdot \sin^2(\pi(n-L+D+0.5)/2D) & ; L \le n < D+L, \\ 0 & D+L \le n < M. \end{cases} \tag{5.10.1-1}$$

17 where m is the current frame, n is the sample index to the buffer $s_m(n)$, L = 80 is the frame length, D = 24 is the

18 overlap (or delay) in samples, and M = 128 is the DFT sequence length. This results in the input buffer containing L + D

19 = 104 samples, and being zero-padded to M samples.

20 The transformation of $s_m(n)$ to the frequency domain is performed using the Discrete Fourier Transform (DFT) to

21 produce $S_m(k)$ as follows:

$$S_m(k) = \frac{2}{M} \sum_{n=0}^{M-1} s_m(n) e^{-j2\pi nk/M} \quad ; 0 \le k < M$$

<div align="right">(5.10.1-2)</div>

5.10.2 Channel Energy Estimator

Calculate the channel energy estimate $E_{Nch}(m,i)$ in decibels (dB) for the current frame, m, as:

$$E_{Nch}(m,i) = \max\left\{ E_{N\min}, \alpha_w(m) E_{Nch}(m-1,i) + (1-\alpha_w(m)) \cdot \left(\frac{10}{f_H(i) - f_L(i) + 1} \sum_{k=f_L(i)}^{f_H(i)} \log_{10}\left(|S_m(k)|^2\right) + c_m \right) \right\} \quad (5.10.2\text{-}1)$$

where $E_{N\min} = -15dB$ is the minimum allowable channel energy, $\alpha_w(m)$ is the channel energy smoothing factor (defined below), $N_c = 16$ is the number of combined channels, and $f_L(i)$ and $f_H(i)$ are the i-th elements of the respective low and high channel combining tables, which are defined in the noise suppressor (Section 4.4.3.2).

The channel energy smoothing factor, $\alpha_w(m)$, is defined as:

$$\alpha_w(m) = \begin{cases} 0 & ; m < 3 \\ 0.6 \cdot w_\alpha & ; m \ge 3, rate(m) = 1/8 \\ w_\alpha & ; otherwise \end{cases}$$

<div align="right">(5.10.2-2)</div>

and

$$c_m = \begin{cases} 3.52 & ; rate(m) \ne 1/8 \\ 0 & ; otherwise \end{cases}$$

<div align="right">(5.10.2-3)</div>

This means that $\alpha_w(m)$ assumes a value of zero for the first two frames (m < 3) and a value of 0.6 times the weight coefficient w_α for all subsequent frames. This allows the channel energy estimate to be initialized to the unfiltered channel energy of the first frame, and some control over the adaptation via the weight coefficient. The weight coefficient is varied according to:

$$w_\alpha = \begin{cases} 1.0 & ; FER\_FLAG(m) = 1 \\ 1.1 & ; FER\_FLAG(m) \ne 1 \end{cases}$$

<div align="right">(5.10.2-4)</div>

5.10.3 Background Noise Estimate

The background noise energy estimate $\mathbf{E}_{bgn}$ is updated according to the following:

If the current frame is a rate 1/8 frame,

$$E_{bgn}(m,i) = \beta_1 \cdot E_{Nch}(m-1,i) + (1-\beta_1) \cdot E_{Nch}(m,i)$$

<div align="right">(5.10.3-1)</div>

For all other frame rates:

$$E_{bgn}(m,i) = \begin{cases} E_{Nch}(m,i) & ; E_{Nch}(m,i) < E_{bgn}(m-1,i) \\ E_{bgn}(m-1,i) + \Delta_1 & ; (E_{Nch}(m,i) - E_{bgn}(m-1,i)) > E_{voice} \\ E_{bgn}(m-1,i) + \Delta_2 & ; otherwise \end{cases}$$

1   (5.10.3-2)

2   where $\Delta_1 = 0.005$, $\Delta_2 = 0.010$ $E_{voice} = 12dB$.

3   NOTE: The background update routines are only called when FER_FLAG(m) is not set and Rate is not set to Suppressed
4   Rate 1/8.

5.11 Background Noise Estimation (SO 70 and SO 73 in WB Mode)

6   To allow suppressed Rate 1/8 frames, the output of the adaptive postfilter is used to determine an estimated background
7   noise spectrum.

8   **Inputs:**

9   • The post-filtered synthesis signal, $\hat{s}_{pf}(n)$

10   • The current frame rate, Rate

11   • The current subframe interpolated LPC pcoefficients, $\dot{A}_q(n)$

12   • Background noise estimate adaptation factor, w_α

13   **Outputs:** None. Only state variables affected.

14   **State Variables:** The background noise envelope estimate, $\mathbf{E}_{bgn}$.

15   **Initialization:** The following variables shall be set to zero at initialization (frame m=0):

16   • The overlapped portion of the input buffer (output of the postfilter), {}

17   The following shall be initialized to a startup value other than zero:

18   • The channel noise estimate, $E_{bgn}(0)$

19   • The channel energy leaky integrator's memory, $E_{Nch}(0)$

20   **Processing:** The background noise spectral estimate is only computed on frames where FER_FLAG(m) is not set and
21   Rate is not equal to Suppressed Rate 1/8. The post-filtered speech signal, $\hat{s}_{pf}(n)$, is used to determine the background
22   noise spectral estimate at the decoder. The background noise spectral estimator for SO 70 and SO 73 in WB mode
23   operates on individual subframes. The following procedures shall be executed for each decoded speech subframe, and
24   the current subframe for the background noise estimator shall be denoted m. Figure 5.11-1 depicts the Background
25   Noise Estimator that is described in the following sections.

1
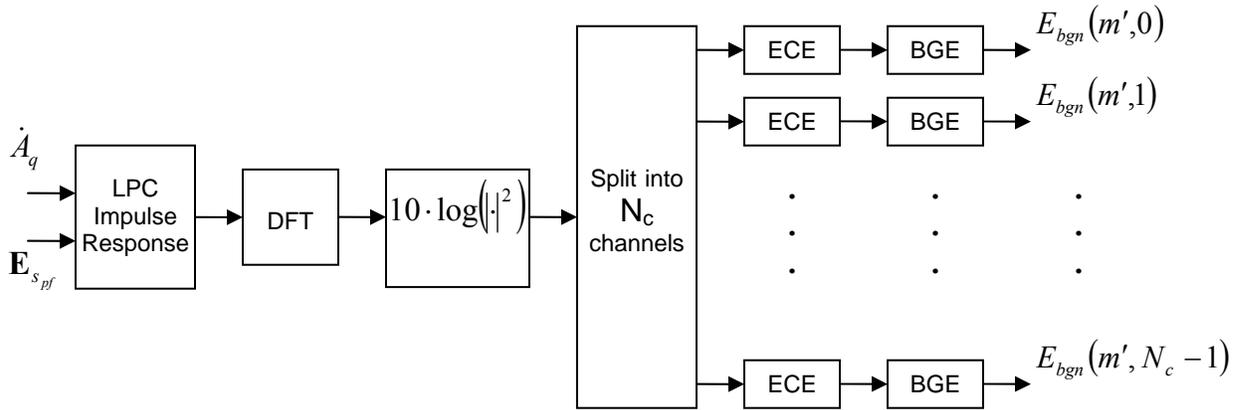
2 **Figure 5.11-1 Background Noise Estimator Block Diagram**

3 ### 5.11.1 Frequency Domain Conversion

4 Before obtaining the frequency domain representation of the current subframe, the impulse response, $h_{A_q}(n)$, of the

5 interpolated LPC coefficients $\dot{A}_q$ is computed to $M = 128$. Its energy is determined by:

6
$$E_{h_{Aq}} = \sum_{n=0}^{M-1} h^2_{A_q}(n)$$
(5.11.1-1)

7 Also, the energy of the post-filtered signal is computed as follows:

8
$$E_{s_{pf}} = \sum_{n=0}^{L-1} \hat{s}^2_{pf}(n)$$
(5.11.1-2)

9 The frequency domain representation of the current subframe is determined from the impulse response according to:

10
$$H_{A_q}(k) = \frac{2}{M} \cdot \sum_{n=0}^{M-1} h_{A_q}(n) \cdot e^{-j2\pi nk/M}$$
(5.11.1-3)

11 The energy in $H_{A_q}(k)$ needs to be adjusted to reflect the energy in the post-filtered signal $\hat{s}_{pf}(n)$. The energy

12 compensating factor is computed as:

13
$$E_{comp} = \frac{M}{L} \cdot \frac{E_{s_{pf}}}{E_{h_{aq}}}$$
(5.11.1-4)

14 or in dB as:

15
$$E_{comp\_dB} = 10 \cdot \log_{10}\left(\frac{1}{L} \cdot E_{s_{pf}}\right) - 10 \cdot \log_{10}\left(\frac{1}{M} \cdot E_{h_{aq}}\right)$$
(5.11.1-5)

16 The frequency domain magnitudes are then obtained from:

$$S_{mdB}(k) = 10 \cdot \log_{10}\left(\left|H_{A_q}(k)\right|^2\right) + E_{comp\_dB} \tag{5.11.1-6}$$

5.11.2 Channel Energy Estimator

Calculate the channel energy estimate $E_{Nch}(m',i)$ in decibels (dB) for the current subframe, m', as:

$$E_{Nch}(m',i) = \max\left\{E_{N\min}, \alpha_w(m') \cdot E_{Nch}(m'-1,i) + (1-\alpha_w(m')) \cdot \left(\frac{1}{f_H(i)-f_L(i)+1} \cdot \sum_{k=f_L(i)}^{f_H(i)} S_{mdB}(k) + c_m\right)\right\}$$

$$\tag{5.11.2-1}$$

where $E_{N\min} = -15dB$ is the minimum allowable channel energy, $\alpha_w(m')$ is the channel energy smoothing factor (defined below), $N_c = 16$ is the number of combined channels, and $f_L(i)$ and $f_H(i)$ are the i-th elements of the respective low and high channel combining tables, which are defined in the noise suppressor (Section 4.4.3.2).

The channel energy smoothing factor, $\alpha_w(m')$, is defined as:

$$\alpha_w(m') = \begin{cases} 0 & ; m < 3 \\ 0.8 \cdot w_\alpha & ; m \geq 3, rate(m) = 1/8 \\ w_\alpha & ; otherwise \end{cases} \tag{5.11.2-2}$$

and

$$c_m = \begin{cases} 3.52 & ; rate(m) \neq 1/8 \\ 0 & ; otherwise \end{cases} \tag{5.11.2-3}$$

This means that $\alpha_w(m')$ assumes a value of zero for the first two frames (m < 3) and a value of 0.8 times the weight coefficient w_α for all subsequent frames. This allows the channel energy estimate to be initialized to the unfiltered channel energy of the first frame, and some control over the adaptation via the weight coefficient. The weight coefficient is varied according to:

$$w_\alpha = \begin{cases} 1.0 & ; FER\_FLAG(m) = 1 \\ 1.1 & ; FER\_FLAG(m) \neq 1 \end{cases}$$

$$\tag{5.11.2-4}$$

5.11.3 Background Noise Estimate

The background noise energy estimate $\mathbf{E}_{bgn}$ is updated according to the following:

If the current frame is a rate 1/8 frame,

$$E_{bgn}(m',i) = \beta_1 \cdot E_{Nch}(m'-1,i) + (1-\beta_1) \cdot E_{Nch}(m',i) \tag{5.11.3-1}$$

For all other frame rates:

$$E_{bgn}(m',i) = \begin{cases} E_{Nch}(m',i) & ; E_{Nch}(m',i) < E_{bgn}(m'-1,i) \\ E_{bgn}(m'-1,i) + \Delta_1 & ; (E_{Nch}(m',i) - E_{bgn}(m'-1,i)) > E_{voice} \\ E_{bgn}(m'-1,i) + \Delta_2 & ; otherwise \end{cases}$$
(5.11.3-2)

where $\Delta_1 = 0.005$, $\Delta_2 = 0.010$, and $E_{voice} = 12dB$.

NOTE: The background update routines are only called when FER_FLAG(m) is not set and Rate is not set to Suppressed Rate 1/8.

5.11.4 Suppressed Eighth-Rate Likelihood Predictor

In some systems the rate=0 may not be defined or practical to use. In these systems suppressed Eighth-Rate frames are tagged as erased frames (rate=0xE) and there is no distinction between an error frame (erased) or an Eighth-Rate frame that was suppressed. In order to distinguish between these two cases labeled as erasures, the background energy estimator determines the likelihood of having a suppressed Eighth-Rate frame come after the current frame and sets a flag accordingly. When an rate=0xE frame is received, this information is used to determine how to handle the erasure (see Section 5.1.3).

The *dtx_likely_flag* is set according to the energy in the current subframe as compared to the estimated background noise energy using:

$$dtx\_likely\_flag = \begin{cases} true & ; energy\{\hat{s}_{pf}\} < energy\{E_{bgn}\} \\ false & ; otherwise \end{cases}$$
(5.11.4-1)

5.12 Generation of High Band Signal for 16 kHz Decoding for SO 70 and SO 73

The High band excitation is derived from the lowband excitation as described in Section 4.8.13.

This excitation is LPC filtered using the dequantized high-band LPC (obtained after dequantization and LSP to LPC conversion).

The frame gains and the subframe gains are then applied to the synthesized high-band speech signal.

When FER_FLAG(m) is set, the previous frame's high-band LSPs and gains are used to generate the high-band signal for the erased frame.

For Eighth-Rate frames, the high-band signal is using a sort of a Discontinuous Transmission scheme for the high-band. The previous frame's (the encoder ensures that there is atleast 1 Half-Rate frame unvoiced frame) high-band LSP and gain parameters are used to synthesize the Eighth-Rate frame in addition to the low-band excitaiton.

5.13 Decoding of Synthesis Filterbank for 16 kHz Decoding for SO 70 and SO 73
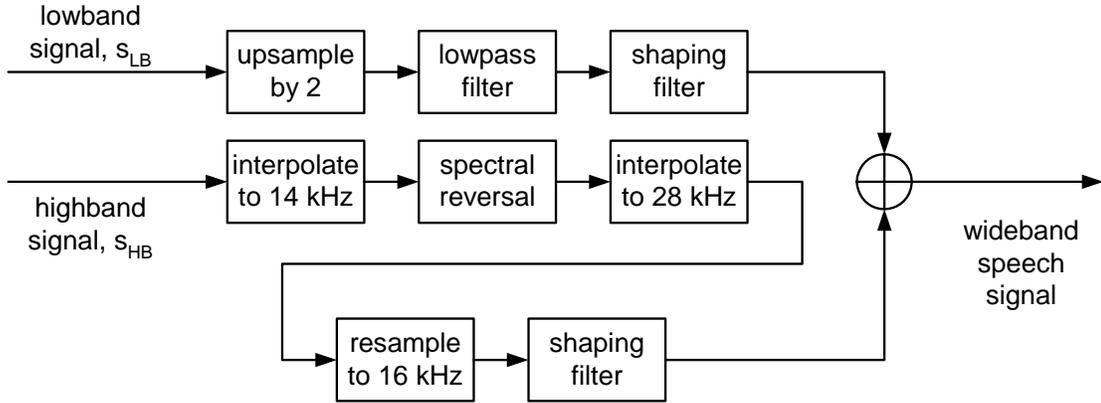
Figure 5.13-1 Synthesis Filterbank for SO 70 and SO 73 16 kHz Decoding

5.13.1 Processing the Lowband Signal, s_{LB}

5.13.1.1 Upsample-by-2 and Lowpass Filter

The upsample-by-2 and the low pass filters can be realized as a polyphase implementation.

5.13.1.2 Shaping Filter

The upsampled lowband signal, is then shaped using a 1st order pole-zero filter to yield the 16 kHz lowband signal, $s_{LB,16kHz}$.

5.13.2 Processing the High Band Signal, s_{HB}

5.13.2.1 Interpolation to 14 kHz

The 7 kHz signal s_{HB} is first upsampled to 14 kHz, $s_{14}[n]$, through the use of a 3-section polyphase implementation.

The even components of $s_{14}[n]$, for n=0, 2, 4, 6 etc are obtained by filtering $s_{HB}[n/2]$ through an allpass filter whose transfer function is given by

$$H_{up2,0} = \left(\frac{a_{up2,0,0} + z^{-1}}{1 + a_{up2,0,0} z^{-1}} \right) \left(\frac{a_{up2,0,1} + z^{-1}}{1 + a_{up2,0,1} z^{-1}} \right) \left(\frac{a_{up2,0,2} + z^{-1}}{1 + a_{up2,0,2} z^{-1}} \right)$$

And the odd components of $s_{14}[n]$, for n=1, 3, 5, 7 etc are obtained by filtering $s_{HB}[n/2]$ through an allpass filter whose transfer function is given by

$$H_{up2,1} = \left(\frac{a_{up2,1,0} + z^{-1}}{1 + a_{up2,1,0} z^{-1}} \right) \left(\frac{a_{up2,1,1} + z^{-1}}{1 + a_{up2,1,1} z^{-1}} \right) \left(\frac{a_{up2,1,2} + z^{-1}}{1 + a_{up2,1,2} z^{-1}} \right)$$

5.13.2.2 Spectral Reversal

The spectrum of $\hat{s}_{14}$ is reversed by multiplying the odd components by -1 to yield a signal s_{14}.

5.13.2.3 Interpolation to 28 kHz

The 14 kHz signal s_{HB} is then upsampled to 28, $s_{28}[n]$, through the use of a 3-section polyphase implementation.

The even components of $s_{28}[n]$, for n=0, 2, 4, 6 etc are obtained by filtering $s_{HB}[n/2]$ through an allpass filter whose transfer function is given by

$$H_{up2,0} = \left(\frac{a_{up2,0,0} + z^{-1}}{1 + a_{up2,0,0}z^{-1}} \right)\left(\frac{a_{up2,0,1} + z^{-1}}{1 + a_{up2,0,1}z^{-1}} \right)\left(\frac{a_{up2,0,2} + z^{-1}}{1 + a_{up2,0,2}z^{-1}} \right)$$

And the odd components of $s_{28}[n]$, for n=1, 3, 5, 7 etc are obtained by filtering $s_{HB}[n/2]$ through an allpass filter whose transfer function is given by

$$H_{up2,1} = \left(\frac{a_{up2,1,0} + z^{-1}}{1 + a_{up2,1,0}z^{-1}} \right)\left(\frac{a_{up2,1,1} + z^{-1}}{1 + a_{up2,1,1}z^{-1}} \right)\left(\frac{a_{up2,1,2} + z^{-1}}{1 + a_{up2,1,2}z^{-1}} \right)$$

5.13.2.4 Resampling to 16 kHz

$S_{28}[n]$ is resampled by a factor of 4/7 to obtain a 16 kHz signal, $s_{16}[n]$. This is obtained using a polyphase interpolation according to

$$s_{28}(7n+j) = \sum_{k=0}^{9} h_{32to28}(j,k)s_{32}(8n+j), \qquad n = 0,1,2...,320/8-1, j = 0,1,2,...,6$$

5.13.3 Combining the Lowband and Highband

The lowband signal, s_{LB}, and the highband signal, s_{HB}, are combined to give the final 16 kHz wideband output as

$$s_{16}(n) = s_{LB}(n) + 0.9s_{HB}(n)$$

6   TTY/TDD EXTENSION FOR SO 3

6.1 Introduction

This section provides an option to reliably transport the TTY/TDD 45.45 bps and 50 bps Baudot code, making digital wireless technology accessible to TTY/TDD users. This section is separated into two major components. Section 6.3 describes the new interface between the encoder and the decoder for transporting the TTY information. Section 6.3.4 is a description of the TTY/TDD software simulation of this section, and is offered only as a recommendation for implementation.   However, in the event of ambiguous or contradictory information, the software simulation shall be used to resolve any conflicts.

This section uses the following verbal forms: "Shall" and "shall not" identify requirements to be followed strictly to conform to the standard and from which no deviation is permitted. "Should" and "should not" indicate that one of several possibilities is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited. "May" and "need not" indicate a course of action permissible within the limits of the standard. "Can" and "cannot" are used for statements of possibility and capability, whether material, physical, or causal.

6.2 Overview

The following sections provide a method for reliably transporting the 45.45 bps and 50 bps Baudot code in the audio path, making digital wireless telephony accessible to TTY/TDD users. The following extension is robust to frame and bit errors and is completely interoperable with the pre-existing EVRC [S1] speech-coding standard.   The solution supports voice carryover/hearing carryover (VCO/HCO). VCO allows a TTY/TDD user to switch between receiving TTY and talking into the phone. Similarly, HCO allows a user to switch between transmitting TTY characters and picking up the phone to listen. When Baudot tones are not present, the vocoder operates as usual, there is no modification or added delay to the voice path when speech is present.

The TTY/TDD audio solution transports Baudot signals through the vocoder by detecting the characters, and their baud rate, that are being transmitted by the TTY/TDD in the encoder and conveying those characters to the decoder. Because one Baudot character spans at least 7 speech processing frames, the character being transmitted shall be sent a minimum of 6 times to the decoder, allowing the decoder to correctly regenerate the character despite frame errors and random bit errors in the speech packet.

The TTY characters are concealed in the speech packet in a way that interoperates with legacy vocoders that have not been modified for TTY. This is made possible because, when Baudot tones are present, the TTY information replaces the pitch lag bits for the adaptive codebook (ACB) and the ACB gain is set to zero so that an unmodified decoder ignores the TTY information.   The rest of the bits in the speech packet contain information for an unmodified decoder to reconstruct the Baudot signal with the fixed codebook and the linear prediction (LPC) filter at least as well as if the encoder was not modified. Furthermore, the encoder shall disable noise suppression, and the rate shall be set to Full-Rate when the Baudot tones are present. This further enhances the system's performance when a modified encoder is interoperating with an unmodified decoder.

A decoder modified with this extension maintains a history buffer to monitor the ACB gain and pitch lag in the speech packets. When the decoder detects that the ACB gain has been set to zero, and the pitch lag contains

information consistent with TTY, the decoder stops decoding speech and begins regenerating the Baudot tones. When the decoder stops detecting TTY information, it resumes processing speech.

When Baudot tones are not present, the modified vocoder operates on speech in exactly the same way as the unmodified vocoder. The TTY processing does not add any additional delay to the speech path.

6.3 TTY/TDD Extension

The TTY processing in the encoder shall process the received PCM one frame at a time and label each frame as NON_TTY, or as TTY_SILENCE, or as a TTY character. The vocoder will be in one of two states: TTY_MODE or NON_TTY_MODE. In the absence of Baudot tones, the encoder and decoder shall be in the NON_TTY_MODE, and the encoder and decoder shall process the frame as speech. When Baudot tones are present, the encoder and decoder shall enter TTY_MODE and process the TTY information as described below.

There shall exist a mechanism to disable the TTY/TDD extension in the vocoder, reverting the vocoder to its unmodified state.

6.3.1 TTY Onset Procedure

The TTY Onset Procedure describes the process by which the vocoder shall transition from the speech mode to the TTY mode.

6.3.1.1 Encoder TTY Onset Procedure

When the TTY encoder processing initially detects that Baudot tones are present, the encoder shall label each frame as TTY_SILENCE until it buffers enough frames to detect the character being sent. The TTY_SILENCE message shall be sent to the decoder according to the method described below. Because of the delay caused by the buffering in the encoder and decoder to detect TTY characters, it is necessary to alert the decoder to mute its output when Baudot tones are first detected. This prevents the Baudot tones from getting through the speech path before the TTY decoder processing is able to detect the TTY characters and regenerate the tones. The TTY_SILENCE message shall be sent to the decoder within 2 frames after the PCM containing the Baudot tones initially enters the encoder. However, in order to reduce the risk of false alarms, the TTY encoder may delay sending the TTY_SILENCE message for the very first character in a call. The TTY_SILENCE message shall be sent for a minimum of 4 frames and shall continue to be sent until a TTY character is detected, or until a NON_TTY frame is detected.

6.3.1.2 Decoder TTY Onset Procedure

When the decoder is in NON_TTY_MODE, the packet shall be decoded in the usual manner for speech. Because there are no bits in the packet to switch the decoder's state, the decoder shall infer the presence of TTY information from the ACB gain and pitch information. The decoder shall recognize when TTY_SILENCE messages are being sent in the packets and transition from NON_TTY_MODE to TTY_MODE before the decoder's speech path reconstructs a TTY character from the audio information in the speech packets. When the decoder makes the transition to TTY_MODE, it shall mute its output until it detects TTY characters or until it transitions back to NON_TTY_MODE. Refer to the implementation recommendation in Section 6.3.4 for an example of the TTY decoder processing.

6.3.1.3 TTY_MODE Processing

The format of the Baudot code can be found in [N11]. The Baudot code is a carrierless, binary FSK signaling scheme. A 1400 Hz. tone is used to signal a logical "1" and an 1800 Hz. tone is used to signal a logical "0". A TTY bit has a duration of 22 ± 0.4 ms for 45.45 baud and 20 ± 0.4 ms for 50 baud. A character consists of 1 start bit, 5 data bits, and $1.5 - 2$ stop bits. When a character is not being transmitted, silence, or a noisy equivalent, is transmitted. Hence, a TTY character spans a minimum of 7 speech processing frames. When the TTY encoder processing detects a character, it shall send the character, its baud rate, and its header (see Section 6.3.2 for a description of the header) to the decoder over a minimum of 6 consecutive frames and a maximum of 16 frames. Because channel impairments cause frame errors and bit errors, the decoder may not receive all of the packets sent by the encoder. The decoder shall use the redundancy to correct any corrupted TTY information. Once the decoder recognizes the TTY character being sent, the decoder's TTY repeater shall regenerate the Baudot tones corresponding to that character.

At call startup, the encoder processing may initialize its baud rate to either 45.45 baud or 50 baud. It is recommended that the default baud rate be set to the predominant baud rate of the region. Because the encoder may require several characters to determine the correct baud rate, it should be expected that the baud rate may change 3-7 characters into the TTY call.

6.3.1.4 TTY_SILENCE Processing

In order to reduce the average data rate of a TTY call, the TTY processing shall be capable of transmitting 1/8 rate packets to the decoder when the encoder is processing the silence periods between characters. Since no TTY information is in the 1/8 packet, the decoder shall infer TTY_SILENCE from an 1/8 rate packet when it is in TTY_MODE. The TTY_SILENCE message may also be sent to the decoder using a Full-Rate frame, as described in Section 6.3.2. When setting the rate to accommodate the TTY information, care shall be taken so that a Full-Rate frame is not immediately followed by an 1/8 rate frame. EVRC-A defines this to be an illegal rate transition and will force an unmodified decoder to declare a frame erasure.

6.3.2 TTY Header, Baud Rate, and Character Format

The TTY information put into the speech packet contains header, character, and baud rate information. When the encoder is transmitting a TTY character, the header shall contain a sequence number to distinguish that character from its preceding and following neighbors. The same header, character, and baud rate information shall be transmitted for each instance of a character for a minimum of 6 frames and a maximum of 16 frames. The header shall cycle through its range of valid values, one value for each instance of a character. The header and character field shall be assigned a value to correspond to the TTY_SILENCE message. TTY_SILENCE may also be conveyed by 1/8 rate packets (see Section 6.3.1.4).

The rate bit shall specify the baud rate to be regenerated by the decoder. The rate bit shall be set to '0' to denote 45.45 baud and '1' to denote 50 baud. During the TTY_SILENCE message, the baud rate bit shall be set to its last transmitted value. In the case where the baud rate has not yet been determined, the default startup baud rate shall be used. The encoder processing may initialize its baud rate to either 45.45 baud or 50 baud.

1   **Table 6.3.2-1 TTY Header and Character Fields**

Description	Range		
	Header (2 bits)	Character (5 bits)	Baud rate (1 bit)
Reserved	0	0 – 31	0 – 1
TTY Character	1 – 2	0 – 31	0– 1
TTY_SILENCE	3	4	0 – 1
Reserved	3	0 – 3, 5 – 31	01

2

3   The combinations of valid values for the TTY header, character, and baud rate fields are specified in Table 6.3.2-1.
4   Note that the value for TTY_SILENCE corresponds to the index for the maximum pitch value allowed by the
5   vocoder. All unused values are reserved for future use and shall be considered as invalid values for the purposes of
6   this annex.

7   ## 6.3.3 Transporting the TTY Information in the Speech Packet

8   In Full-Rate and Half-Rate, there are 7 bits per frame assigned to the pitch lag. In Full-Rate, the last bit of the
9   speech packet, Bit 171, is reserved by the speech coder. This bit plus the 7 bits assigned to the pitch lag shall be
10   used to convey TTY information from the encoder to the decoder using Full-Rate frames. Half-Rate packets may
11   also be used for conveying TTY information. Because Half-Rate packets have only 7 bits available, i.e., the 7 pitch
12   lag bits, Half-Rate frames shall contain the TTY header and TTY character information only and the decoder shall
13   use its last valid baud rate to regenerate TTY characters. In order to improve interoperability between a modified
14   encoder and an unmodified decoder, it is recommended to transport the TTY information in a Full-Rate packet;
15   however, a modified decoder shall be capable of detecting TTY information in both Full-Rate and Half-Rate
16   packets.

17   The TTY information replaces the pitch lag bits and the reserved Bit 171. The ACB gain shall be set to zero for
18   each subframe in packets containing TTY information. Bit 171 shall be used to convey the baud rate information
19   and the 7 pitch lag bits shall be used to convey the TTY header and character information. The 5 least significant
20   bits of the pitch bits shall be used for the 5-bit Baudot code. Two remaining pitch bits shall be used for the TTY
21   header information. The TTY information is assigned to the pitch lag bits according to. Table 6.3.3-1.

22   **Table 6.3.3-1 TTY Header and Character Bit Assignment**

PITCH LAG BIT ASSIGNMENT						
MSB						LSB
6	5	4	3	2	1	0
TTY HEADER		5 BIT BAUDOT CODE				
MSB	LSB	MSB				LSB
1	0	4	3	2	1	0

23   ## 6.3.3.1 Half-Rate TTY Mode

24   In the case where the encoder and decoder are both modified for TTY, it is possible to reduce the average data rate
25   by using Half-Rate packets to transport TTY information. Half-Rate packets should only be used to transport TTY
26   information during Dim & Burst signaling, or when it is determined that both the near-end and far-end vocoders are

TTY capable. When the near-end (far-end) decoder recognizes that the far-end (near-end) encoder is sending TTY information, the near-end (far-end) encoder may be notified by the near-end (far-end) decoder to send TTY information in Half-Rate packets. When the near-end (far-end) decoder receives a NON_TTY packet, the near-end (far-end) encoder should exit Half-Rate TTY mode. This preserves interoperability in the event of a hard handoff from a modified vocoder to an unmodified vocoder. Because Half-Rate packets do not convey the baud rate, the vocoder should return to Full-Rate for at least 4 characters after a change in baud rate before returning to Half-Rate TTY mode, whenever possible. In Dim & Burst signaling, for example, it may be more desirable to stay in Half-Rate TTY mode to complete the signaling message rather than go to Full-Rate to convey a baud rate change.

6.3.3.2 Interoperability with 45.45 Baud-Only TTY Extensions

Previous TTY extensions support 45.45 bps Baudot code only. This extension supports both 45.45 baud and 50 baud by using an additional bit to convey the baud rate. Note, the other 7 bits used for the character information and header are the same as the 45.45 baud-only TTY extensions.

Because the baud rate uses a bit in the encoded speech packet that was not previously used for TTY, some level of interoperability is achieved (See Table 6.3.3.2-1). When a 45.45 baud decoder is receiving 50 baud TTY packets, it will regenerate the characters at 45.45 baud. In this case, the regenerator may fall behind and drop characters because it is regenerating characters at a slower rate than the encoder is sending them.

Table 6.3.3.2-1 Baud Rate Interoperability Matrix

	45.45 baud-only decoder	**45.45 and 50 baud decoder**
45.45 baud-only encoder	Compatible	Either 45.45 or 50 baud will be regenerated by the decoder, depending on the value of the baud rate bit set by the encoder.
45.45 and 50 baud encoder	Baud rate is ignored and decoder always regenerates 45.45 baud.	Compatible

In the case where the encoder is 45.45 baud-only and the decoder is capable of regenerating either 45.45 or 50 baud, the 2 header bits and the 5 TTY character bits are the same, so the decoder is able to decode these bits correctly. The baud rate bit, however, has no meaning to the 45.45 baud-only encoder, so it will be set randomly, depending on the implementation, and the baud rate used by the decoder will depend on the value of the baud rate bit.

In order for 45.45 baud-only implementations to interoperate with this extension, it is recommended that the 45.45 baud-only encoder implementations set the reserved Bit 171 to zero so that the baud rate bit is set to 45.45 baud.

6.3.3.3 Reflected Baudot Tones

It is possible for the signal generated by the TTY solution to reflect back to the near-end TTY detector, either on the network side or the mobile side. Possible sources of the reflected signal are crosstalk, impedance mismatch, or hybrid echo. To prevent the detector from detecting Baudot tones, that are not intentionally originated by the TTY device, the input PCM shall be muted, before being processed by the near-end encoder, whenever Baudot tones are being regenerated. This requirement applies to both the network and mobiles. Note, the sample solution described in Section 6.3.4 does not contain a mechanism for blocking reflected Baudot tones.

6.3.4 TTY/TDD Processing Recommendation

The following describes the software simulation of this annex. It is intended as a recommendation for implementation only and is not required to satisfy compliance with this annex. However, the software shall be used to resolve ambiguous or incomplete statements that may exist in the sections above. The TTY/TDD processing is divided into 2 major components, encoder processing and decoder processing. The TTY encoder process detects the presence of Baudot tones and decodes the TTY character being transmitted. It then conveys that information to the decoder. The TTY decoder processing shall detect the presence of TTY information and regenerate the Baudot tones corresponding to that character. Refer to Figure 6.3.4-1 for a block diagram of the TTY processing.
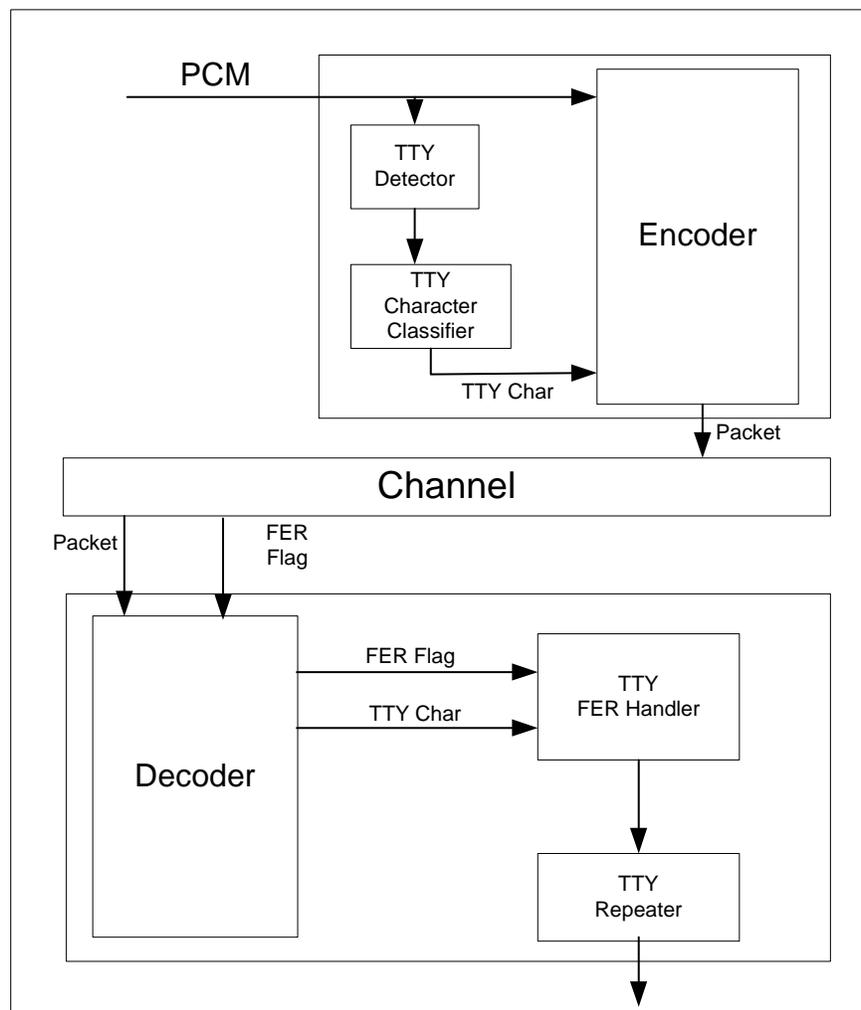
Figure 6.3.4-1 TTY/TDD Processing Block Diagram

6.3.5 TTY Encoder Processing

The TTY encoder processing takes the larger task of detecting TTY characters and divides it into a series of smaller tasks, creating different levels of detection. It is through this divide and conquer approach that the tty_enc( ) routine has low complexity in the absence of Baudot tones.

The first level of detection is to divide the 160 samples in the speech frame into 10 blocks of 16 samples. These blocks are called detection intervals, or dits. Each dit is classified as NON_TTY, LOGIC_0, or LOGIC_1.

The next level of detection is to determine if there are enough LOGIC_0 or LOGIC_1 dits in a row to form a TTY bit. The transition from a "0" bit to a "1" bit or the detection of two consecutive "0" bits signals the onset of a character and the TTY_SILENCE message is sent to the decoder. The TTY_SILENCE message shall continue to be sent until a TTY character is detected, or until a NON_TTY frame is detected. When enough bits are detected to form a character, the rate of the character is determined and the TTY character information is sent to the decoder.

6.3.5.1 TTY Encoder Inputs

- TTY character information from the previous frame (header, TTY character, and baud rate).

- 160 PCM samples from the output of the high pass filter.

6.3.5.2 Dit Classification

The 160 samples from the high pass filter's output are converted into 10 dits. For every block of 16 samples, it computes the spectral energy at the mark and space frequencies using a 16-point DFT at 1400 Hz and 1800 Hz with a rectangular window. The ratio of the maximum energy between the mark and space energy and the total energy is compared to a threshold; i.e.:

$$\frac{\max(mark\_energy, space\_energy)}{total\_energy} > \text{THRESH}$$

If that threshold is exceeded, the dit is labeled as either a mark or a space, whichever one has the greater energy. If the threshold is not met, the dit is labeled as NON_TTY.

6.3.5.3 Dits to Bits

The dits are used to form a TTY bit. Dits are classified as LOGIC_0, LOGIC_1, or UNKNOWN. A nominal bit consists of 11 dits for 45.45 baud and 10 dits for 50 baud. A bit is not required to have a continuous run of LOGIC_0's or LOGIC_1's in order to be detected. Spurious UNKNOWN detections are permitted, up to a threshold.

A TTY bit is searched by looking at the dits within a variable-length sliding window. The window varies in length from 8 dits to 13 dits. The number of LOGIC_0's, LOGIC_1's, and UNKNOWN dit detections are counted in the window. The dits in the search window must meet the following criteria in order to detect a TTY bit:

- Minimum of 6 LOGIC_0 (LOGIC_1) dits

- Maximum of 2 LOGIC_1 (LOGIC_0) dits

- Maximum of 5 UNKNOWN dits

Heuristics are also applied so that the sliding window is centered over the TTY bit being detected. If all of the thresholds are met, a "0" ("1") TTY bit is declared, and the overall length of the bit and the number of bad dits within the window are recorded. If the dits in the search window do not meet the criteria for a TTY bit, a gap between TTY bits is declared and the gap is recorded. The window is slid by one dit and the search is repeated.

The length of the search window is allowed to vary. The length is adjusted depending on the previous character's baud rate, and where the mark/space transitions occur.

Two history buffers are maintained to record the detected TTY bits and gaps. The array tty_bit_hist[ ] maintains a history of the bits that are detected and tty_bit_len_hist[ ] stores the lengths, in dits, of the gaps and TTY bits that are detected. Both arrays are 9 elements long, there is one element for the start bit, five for the data bits, one for the stop bit, and one for the memory bit, the bit before the start bit, as depicted in Table 6.3.5.3-1. The last element in the array is used to record partially detected TTY bits.

Table 6.3.5.3-1 TTY Bit History Buffer

0	1	2	3	4	5	6	7	8
Memory Bit	Start Bit	LSB Data 0	Data 1	Data 2	Data 3	MSB Data 4	Stop Bit	Next Bit

When a TTY bit or a gap is detected, its value is recorded in tty_bit_hist[ ] and its length is stored in tty_bit_len_hist[ ]. The length is used by get_tty_char( ) to threshold the length of a candidate character, and by tty_rate( ) to determine if a detected character is 45.45 baud or 50 baud. Since the length of the memory bit is irrelevant, tty_bit_len_hist[0] is used to count the number of NON_TTY and TTY_SILENCE dits that were detected within the TTY bits.

Because the speech frames may not coincide with the boundaries of the TTY bits, it is possible that a bit may straddle two speech frames. It is possible, therefore, that the sliding window may contain only a partial bit within a frame. These partial detections are recorded in element 8 of the TTY bit history buffers, labeled "Next Bit" in Table 6.3.5.3-1.

6.3.5.4 TTY Character Classification

The routine get_tty_char( ) checks if the detected bits form a TTY character. The following conditions must be met in order for a character to be declared.

- The bit preceding the start bit must not be a "0".

- The start bit must be a "0"

- The stop bit must be a "1"

- The length of the candidate character, in dits, must be within a maximum and minimum threshold.

- The number of bad dit detections within the character must be within a threshold.

If all of the conditions are met, a character is declared.

Once a character is found, the character information and its header are sent to the decoder a minimum of 6 frames and a maximum of 16 frames. The constant FRAMING_HANGOVER dictates the maximum number of times the information for the same character is sent. If a new character is framed before FRAMING_HANGOVER is reached, the information for the old character is terminated and the new information is sent to the decoder.

6.3.5.5 TTY Baud Rate Determination

After a character has been detected, tty_rate( ) determines the baud rate of the character by counting the length, in dits, of the start bit and the five data bits. The length of the stop bit is not used because its length is too variable.

A character with a length of 63 dits or greater is declared 45.45 baud, otherwise it is declared 50 baud. A hangover of three characters is maintained before tty_rate( ) will switch from one baud rate to the other. That is to say, if the

baud rate changes in the middle of a call, three consecutive characters must be detected at the new baud rate in order for tty_rate( ) to declare the new baud rate.

6.3.5.6 TTY State Machine

The routine get_tty_state( ) is responsible for changing the state of the TTY encoder processing. There are 3 states, NON_TTY_MODE, TTY_ONSET, and TTY_MODE. Get_tty_state( ) is responsible for determining NON_TTY_MODE and TTY_ONSET.

Changing TTY state from NON_TTY_MODE to TTY_ONSET requires that a "0" bit is followed by a "1" bit. This rule requires the presence of both the space tone and the mark tone and for the tones to be the correct duration. This test must be met in order to declare TTY_ONSET for the first time.

NON_TTY_MODE is declared by get_tty_state( ) whenever a non-TTY bit is detected or when a "0" bit occupies the bit preceding the start bit.

6.3.6 TTY/TDD Decoder Processing

The TTY decoder processing must recognize when TTY/TDD information is in the packet, recover from channel impairments to decode the TTY character being sent, and regenerate the Baudot tones corresponding to that character.

When the decoder is in NON_TTY_MODE, the packet is decoded in the usual manner for speech. When the decoder makes the transition to TTY_MODE, it mutes its output until it receives TTY character information or until it transitions back to NON_TTY_MODE.

6.3.6.1 TTY Decoder Inputs

- TTY Information (header, TTY character, baud rate).

- Bad frame indicator

6.3.6.2 Decoding the TTY/TDD Information

The task of detecting the presence of TTY information, recovering from frame and bit errors, and decoding the TTY character is performed in tty_dec( ). If the routine detects that TTY information is being sent, the tty_dec( ) flag is set to non-zero and the PCM buffer is filled with the appropriate Baudot tones. If TTY is not detected, the flag is set to zero and the PCM buffer is returned unmodified.

Table 6.3.6.2-1 tty_dec( ) History Buffer

Frame:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Description:	Lookahead									Current Frame	Lookback

The routine labels each frame as NON_TTY, TTY_SILENCE, FER, or a TTY character, and maintains a history buffer of these classifications for 11 frames: 9 frames of lookahead, 1 current frame, and 1 frame of lookback (see Table 6.3.6.2-1). The most recent packet enters the buffer at location 0, but the decision for the current frame is based on the contents of element 9. The buffer is updated at the end of each frame, shifting its contents to the right by one. The buffer is initialized to NON_TTY.

At the start of each frame, the most recent information is sanity checked to see if it is consistent with TTY information. If the frame erasure flag is set, the frame is labeled FER, otherwise the TTY/TDD information is checked to see if the header and TTY character fields fall within the allowed range of values. If all of the tests pass, Frame 0 is labeled with the TTY character in the history buffer.

The TTY decoder processing can reliably regenerate the TTY characters despite channel impairments because the character information is transmitted a minimum of 6 times from the encoder. Errors are corrected by a voting process. The current frame and nine frames of lookahead are used to determine the correct TTY header, character, and baud rate. Errors are replaced with the winner of the voting process.

Voting is conducted under the following conditions:

- Any time the current frame is labeled as FER.

- Every time the current frame contains information for the start of a new character. Because a new character must contain a minimum of 6 frames of the same information, a vote is taken to verify that the information is present before it will generate the tones for that character. Once a character wins the vote, any frame errors, bit errors, or other inconsistencies are corrected in the frame window where the character information is expected, i.e., the current frame and the adjacent frames of lookahead will contain the same header and character information.

- Any time the current frame contains TTY_SILENCE or a TTY character, and the frame of lookback contains NON_TTY. This makes it harder for the decoder to erroneously go into TTY_MODE, thus preventing false alarms when speech is present.

In the normal course of TTY transmissions, TTY_SILENCE messages are sent first, followed by the TTY character information. When 3 TTY_SILENCE messages are received in a window of 5 frames, the decoder's output is muted until a NON_TTY frame is received or until the voting results in a TTY character to be regenerated. If a new character is detected, it will only be regenerated if it was preceded by TTY_SILENCE. If it is not, the character information is ignored and the decoder returns to NON_TTY mode. This is done to prevent false alarms from packets that look like TTY packets but are really speech packets with the ACB gain coincidentally set to zero.

Once tty_dec( ) makes its decision on the current frame, tty_dec( ) calls tty_gen( ) to generate the appropriate PCM samples.

6.3.6.3 Baudot Generator

Once the current frame is labeled by tty_dec( ), tty_gen( ) is called to fill the PCM buffer with the appropriate Baudot tones. In the case of NON_TTY, the PCM buffer is returned unmodified. In the case of TTY_SILENCE, the PCM is muted.

Generating TTY characters is more involved because one character spans many frames, so tty_gen( ) must generate the Baudot tones one subframe at a time. When a TTY character needs to be regenerated, tty_gen( ) puts a subframe's worth of samples in the PCM buffer. It keeps track of which bit it is in the middle of generating and the number of samples left to generate for that bit, so that the next time it is called, it can pick up where it left off. Once tty_gen( ) begins to generate a character, it will generate the entire character before it will generate the next character. This is done so that the repeater will only generate valid TTY characters.

There exists logic in tty_gen( ) to detect when the next character arrives before the current one is finished. If the next character arrives before the current one can be regenerated, a minimum of 1 stop bit is generated.

There exists a provision in [N11] for the TTY/TDD device to extend its stop bit in order to prevent a TTY/TDD device from detecting its own echo. This routine will extend the stop bit a maximum of 300 ms if a TTY character is followed by silence. If a new character arrives before 300 ms has elapsed, the extended stop bit is terminated and the new character is generated immediately.

The tones themselves are generated by tone_gen( ). Before tty_gen( ) returns, it updates the decoder's lookback field in the TTY history buffer with the information corresponding to the last samples generated. For example, if tty_gen( ) finished generating a character in the middle of the subframe and started generating silence, the lookback field is updated with TTY_SILENCE.

6.3.6.4 Tone Generator

The routine tone_gen( ) is a sine wave generator. Given a frequency and the number of samples, it will generate the PCM samples by using a 2 tap marginally stable IIR filter. The filter implements the trigonometric identity:

$$\cos(k\omega) = 2 \cdot \cos(\omega) \cdot \cos((k-1)\omega) - \cos((k-2)\omega)$$

It is a zero excitation filter, using only its past 2 samples and the cosine of the frequency to be generated, to produce the next sample.

1

2

3

4

5

6 This page intentionally left blank.

7 EVRC-B, EVRC-WB, AND EVRC-NW DATA CAPABILITIES

7.1 EVRC-B, EVRC-WB, and EVRC-NW Data Capabilities

EVRC-B, EVRC-WB, and EVRC-NW provide the capability for passing data by replacing the speech allocated bits by general user data bits. This capability is called in-band or in-bitstream data. In this document, TTY/TDD Baudot Code (45.45 baud and 50 baud) and DTMF are the only data payloads that are defined and supported by EVRC-B, EVRC-WB, and EVRC-NW.

In-band TTY/TDD Baudot Code is mandatory. All EVRC-B, EVRC-WB, and EVRC-NW implementations shall include in-band TTY/TDD capability. However, in-band DTMF is not always mandatory. Depending upon applications such as wireless local loop (fixed wireless) or mobility, in-band DTMF can be optional. In any case, in-band and out-of-band DTMF shall not be transmitted simultaneously.

For mobility applications, in-band DTMF signaling shall not be used.

For fixed wireless (also known as wireless local loop) applications, sending DTMF in the forward link is common (e.g., answering machine access). In order to guarantee that in-band DTMF can be decoded correctly in both forward and reverse links, implementing the in-band DTMF decoder is mandatory in both directions (forward and reverse), whereas implementing the in-band DTMF encoder is optional.

In-band data is transported with Rate 1/2 packet with data.

The TTY/TDD and DTMF features define their own error recovery schemes. The decoder, however, is capable of automatically detecting the in-band data by decoding the packet. No out-of-band signaling is required to send in-band data from the encoder to the decoder.

When the encoder sends in-band data to the decoder, the data is packed in Rate 1/2 frames and they contain a unique bit pattern that can be identified by the decoder to indicate that the packet contains data. The decoder then reads the header bits to determine which type of data is in the frame and calls the appropriate algorithm to regenerate the signal at the decoder's output, or to send the data to an appropriate application.

The bit allocation for Rate 1/2 data frames is summarized in Table 7.1-1.

Table 7.1-1 Bit Allocation for Rate 1/2 Data Frames

# of bits	Rate 1/2 data frame
3	Data Header
70	Data Block
7	Illegal Lag Codeword

For in-band data formats, a codeword is put in the packet that is considered illegal for speech frames. A packet is identified as containing data if these fields are set to the values below. The illegal codeword for each of the in-band data formats are in Table 7.1-2.

1

Table 7.1-2 Illegal Codeword Value

Codeword	Rate 1/2 data
Illegal Lag Codeword	0x78

2 Whereas the illegal codeword identifies the packet as containing data, the Data Header is a 3-bit field, which

3 identifies the data format and payload. Table 7.1-3 contains a description of the Data Header field for Rate 1/2

4 packets.

5

Table 7.1-3 Data Header Description

Data Header value	Rate 1/2 (bits 2 to 4)
0	Reserved
1	Data
2	TTY/TDD
3	DTMF
4	Reserved
5 to 7	Reserved

6 ## 7.2 TTY/TDD Payload Format

7 The TTY/TDD payload shall be transported using the Rate 1/2 packet data format only. Rate 1 data formats shall not

8 be used for this application. For TTY/TDD payloads, the Data Header field shall be set to 2 (see Table 7.1-3) and

9 the first 8 bits of the data field shall correspond to the TTY Type field. The Baudot code is currently the only TTY

10 Type supported by this standard. Table 7.2-1 summarizes the valid values for the TTY Type field.

11

Table 7.2-1 TTY Type Field

TTY type	Description
0	Reserved
1	Baudot Code
2 to 255	Reserved

12 ## 7.3 Baudot Code Payload

13 Payloads for TTY/TDD Baudot codes at 45.45 baud and 50 baud are defined for EVRC-B, EVRC-WB, and EVRC-

14 NW. For transmitting the Baudot code, the Data Field shall be set to 2 and TTY Type shall be set to 1. For Baudot,

15 the bits in the Rate 1/2 packet are defined as in Table 7.3-1.

Table 7.3-1 Rate 1/2 Packet for TTY/TDD Baudot Code Payload

# of bits	Field	Value
3	Data Header	2
8	TTY Type	1
8	TTY Header	See Table 7.5.1-1
16	TTY Character	See Table 7.5.1-2
16	TTY Rate	See Table 7.5.1-3
22	Reserved	N/A
7	Illegal Codeword	0x78

7.4 DTMF Payload Format

The DTMF payload shall be transported using the Rate 1/2 packet data format only. Rate 1 data formats shall not be used for this application. Table 7.4-1 shows a Rate 1/2 packet with the format for a DTMF payload. The Data Header field shall be set to 3 (see Table 7.1-3) and the first 16 bits of the data field shall correspond to a 16-bit signed value DTMF digit, as specified in Table 7.4-2.

Table 7.4-1 Rate 1/2 Packet with DTMF Payload

# of bits	Field	Value
3	Data Header	3
16	DTMF Digit	See Table 7.4-2
54	Reserved	N/A
7	Illegal Codeword	0x78

Table 7.4-2 DTMF Data Values

Data field	DTMF signal
1	1
2	2
3	4
5	5
6	6
7	7
8	8
9	9
10	0
11	*
12	#
13	A
14	B

Data field	DTMF signal
15	C
16	D
-1	No signal detected

1

2 **7.5 TTY/TDD Operation**

3 The following feature provides a method for reliably transporting the 45.45 bps and 50 bps Baudot code in the
4 EVRC-B, EVRC-WB, and EVRC-NW, making digital wireless telephony accessible to TTY/TDD users. The
5 feature supports voice carryover/hearing carryover (VCO/HCO). VCO allows a TTY/TDD user to switch between
6 receiving TTY and talking into the phone. Similarly, HCO allows a user to switch between transmitting TTY
7 characters and picking up the phone to listen. When Baudot tones are not present, the vocoder operates as usual;
8 there is no modification or added delay to the voice path when speech is present.

9 The Baudot code is specified in [N11]. The Baudot code is a carrierless, binary FSK signaling scheme. A 1400 Hz
10 tone is used to signal a logical "1" and a 1800 Hz tone is used to signal a logical "0". A 45.45 bps TTY bit has a
11 duration of 22 ms $\pm$ 0.4 and a 50 bps bit has a duration of 20 ms. $\pm$ 0.4, and a character consists of 1 start bit, 5 data
12 bits, and 1.5 to 2 stop bits. When a character is not being transmitted, silence, or a noisy equivalent, is transmitted.
13 Hence, a TTY character spans a minimum of 6 speech processing frames. When the TTY encoder processing detects
14 a character, it sends the character and its header (see Section 7.5.1 for a description of the header) to the decoder
15 over a minimum of 6 consecutive frames and a maximum of 16 frames. Because of channel impairments, the
16 decoder may not receive all of the packets sent by the encoder because of frame erasures and random bit errors. The
17 decoder uses the redundancy to correct any corrupted TTY information. Once the decoder recognizes the TTY
18 character being sent, the decoder's TTY repeater regenerates the Baudot tones corresponding to that character. The
19 TTY processing in the encoder processes the received PCM input one frame at a time and labels each frame as either
20 NON_TTY, TTY_ONSET, or as a TTY character. The vocoder is in one of two states, TTY_MODE or
21 NON_TTY_MODE. In the absence of Baudot tones, the encoder and decoder are in the NON_TTY_MODE and the
22 encoder and decoder process the frame as speech. When Baudot tones are present, the encoder and decoder enter
23 TTY_MODE and process the TTY information as described in the following subsections. When the vocoder stops
24 detecting TTY information, it resumes processing speech.

25 It is recommended that a mechanism exist to disable the TTY/TDD feature in the vocoder.

26 Figure 7.5-1 depicts the TTY/TDD processing block diagram.

1
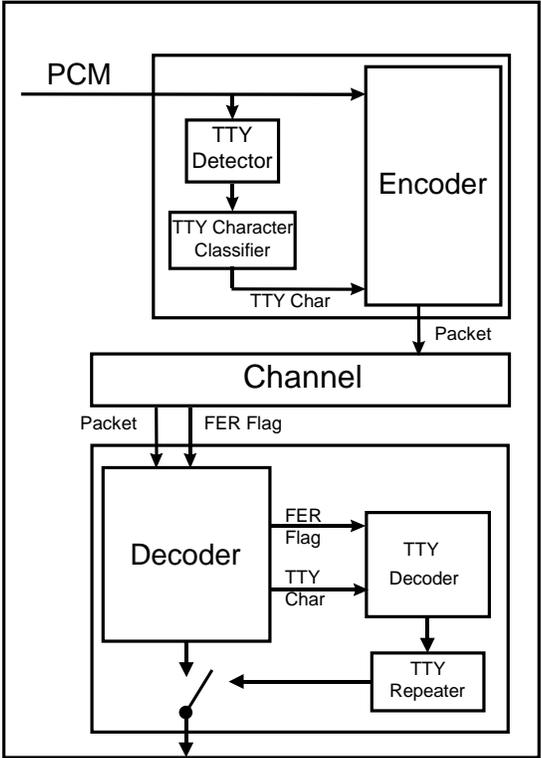
2   **Figure 7.5-1 TTY/TDD Processing Block Diagram**

3   7.5.1   TTY Header and Character Format

4   The TTY information passed from the encoder to the decoder contains a header, character information, and the baud

5   rate. When the encoder is transmitting a TTY character, the header contains a sequence number to distinguish that

6   character from the characters immediately before and after. The same TTY information is transmitted for each

7   instance of a character for a minimum of 6 frames and a maximum of 16 frames. The header cycles through its range

8   of valid values, one value for each instance of a character. During the TTY_ONSET state, the encoder sends the

9   decoder a TTY_SILENCE message in the header and character field. The TTY header uses 2 bits in an 8-bit field

10   (see Table 7.5.1-1). The TTY character uses 5 bits in a 16-bit field (see Table 7.5.1-2). The TTY baud rate uses 1 bit

11   in a 16-bit field (see Table 7.5.1-3).

12   **Table 7.5.1-1 Range of Values for TTY/TDD Baudot Header Field**

Range	Description
0	TTY_SILENCE
1 to 3	TTY Counter
4 to 255	Reserved

1

2   **Table 7.5.1-2 Range of Values for TTY/TDD Baudot Character Field**

Range	Description
4	TTY_SILENCE (when TTY Header = 0)
0 to 31	TTY Character
32 to 65535	Reserved

3

4   **Table 7.5.1-3 Range of Values for TTY/TDD Baud Rate Field**

Range	Description
0	45.45 baud
1	50 baud
2 to 65535	Reserved

5   ## 7.5.2 TTY Encoder Processing

6   The TTY encoder processing takes the larger task of detecting TTY characters and divides it into a series of smaller
7   tasks, creating different levels of detection. It is through this divide and conquer approach that the tty_enc() routine
8   has low complexity in the absence of Baudot tones.

9   The first level of detection is to divide the 160 samples in the speech frame into 10 blocks of 16 samples. These
10   blocks are called detection intervals, or dits. Each dit is classified as NON_TTY, LOGIC_0, or LOGIC_1.

11   The next level of detection is to determine if there are enough LOGIC_0 or LOGIC_1 dits in a row to form a TTY
12   bit. The transition from a "0" bit to a "1" bit is used to determine the onset of a TTY character and the
13   TTY_SILENCE message is sent to the decoder. The TTY_SILENCE message shall continue to be sent until a TTY
14   character is detected, or until a NON_TTY frame is detected. When enough bits are detected to form a character, the
15   rate of the character is determined and the TTY character information is sent to the decoder.

16   ### 7.5.2.1 TTY Encoder Inputs

17   TTY character information from the previous frame (header, TTY character, and baud rate)

18   160 PCM samples from the output of the high pass filter

19   ### 7.5.2.2 Dit Classification

20   The 160 samples from the high pass filter's output are converted into 10 dits. For every block of 16 samples, it
21   computes the spectral energy at the mark and space frequencies using a 16-point DFT at 1400 Hz and 1800 Hz with
22   a rectangular window. The ratio of the maximum energy between the mark and space energy and the total energy is
23   compared to a threshold; i.e.:

24   $$\frac{\max(mark\_energy, space\_energy)}{total\_energy} > \text{THRESH}$$

25   If that threshold is exceeded, the dit is labeled as either a mark or a space, whichever one has the greater energy. If
26   the threshold is not met, the dit is labeled as NON_TTY.

A second threshold is applied to the total energy of the dit. If the total energy is less than the threshold, the dit is classified as NON_TTY. This minimum energy threshold prevents the TTY detector from being overly sensitive to low level tones that may have been unintentionally reflected back into the TTY detector. The threshold is set at approximately -50 dBm.

7.5.2.3 Dits to Bits

The dits are used to form a TTY bit. Dits are classified as LOGIC_0, LOGIC_1, or UNKNOWN. A nominal bit consists of 11 dits for 45.45 baud and 10 dits for 50 baud. A bit is not required to have a continuous run of LOGIC_0's or LOGIC_1's in order to be detected. Spurious UNKNOWN detections are permitted, up to a threshold.

A TTY bit is searched by looking at the dits within a variable length sliding window. The window varies in length from 8 dits to 13 dits. The number of LOGIC_0's, LOGIC_1's, and UNKNOWN dit detections are counted in the window. The dits in the search window must meet the following criteria in order to detect a TTY bit:

- Minimum of 6 LOGIC_0 (LOGIC_1) dits

- Maximum of 2 LOGIC_1 (LOGIC_0) dits

- Maximum of 5 UNKNOWN dits

Heuristics are also applied so that the sliding window is centered over the TTY bit being detected. If all of the thresholds are met, a "0" ("1") TTY bit is declared, and the overall length of the bit and the number of bad dits within the window are recorded. If the dits in the search window do not meet the criteria for a TTY bit, a gap between TTY bits is declared and the gap is recorded. The window is slid by one dit and the search is repeated.

The length of the search window is allowed to vary. The length is adjusted depending on the previous character's baud rate, and where the mark/space transitions occur.

Two history buffers are maintained to record the detected TTY bits and gaps. The array tty_bit_hist[] maintains a history of the bits that are detected and tty_bit_len_hist[] stores the lengths, in dits, of the gaps and TTY bits that are detected. Both arrays are 9 elements long; there is one element for the start bit, five for the data bits, one for the stop bit, and one for the memory bit, the bit before the start bit, as depicted in Table 7.5.2.3-1. The last element in the array is used to record partially detected TTY bits.

Table 7.5.2.3-1 TTY Bit History Buffer

0	1	2	3	4	5	6	7	8
Memory Bit	Start Bit	LSB Data 0	Data 1	Data 2	Data 3	MSB Data 4	Stop Bit	Next Bit

When a TTY bit or a gap is detected, its value is recorded in tty_bit_hist[] and its length is stored in tty_bit_len_hist[]. The length is used by get_tty_char() to threshold the length of a candidate character, and by tty_rate() to determine if a detected character is 45.45 baud or 50 baud. Since the length of the memory bit is irrelevant, tty_bit_len_hist[0] is used to count the number of NON_TTY and TTY_SILENCE dits that were detected within the TTY bits.

Because the speech frames may not coincide with the boundaries of the TTY bits, it is possible that a bit may straddle two speech frames. It is possible, therefore, that the sliding window may contain only a partial bit within a

frame. These partial detections are recorded in element 8 of the TTY bit history buffers, labeled "Next Bit" in Table 7.5.2.3-1.

7.5.2.4 TTY Character Classification

The routine get_tty_char() checks if the detected bits form a TTY character. The following conditions must be met in order for a character to be declared:

- The bit preceding the start bit must not be a "0".

- The start bit must be a "0".

- The stop bit must be a "1".

- The length of the candidate character, in dits, must be within a maximum and minimum threshold.

- The number of bad dit detections within the character must be within a threshold.

If all of the conditions are met, a character is declared.

Once a character is found, the character information and its header are sent to the decoder a minimum of 6 frames and a maximum of 16 frames. The constant FRAMING_HANGOVER dictates the maximum number of times the information for the same character is sent. If a new character is framed before FRAMING_HANGOVER is reached, the information for the old character is terminated and the new information is sent to the decoder.

7.5.2.5 TTY Baud Rate Determination

After a character has been detected, tty_rate() determines the baud rate of the character by counting the length, in dits, of the start bit and the five data bits. The length of the stop bit is not used because its length is too variable.

A character with a length of 63 dits or greater is declared 45.45 baud, otherwise, it is declared 50 baud. A hangover of three characters is maintained before tty_rate() will switch from one baud rate to the other. That is to say, if the baud rate changes in the middle of a call, three consecutive characters must be detected at the new baud rate in order for tty_rate() to declare the new baud rate.

7.5.2.6 TTY State Machine

The routine get_tty_state() is responsible for changing the state of the TTY encoder processing. There are 3 states, NON_TTY_MODE, TTY_ONSET, and TTY_MODE. Get_tty_state() is responsible for determining NON_TTY_MODE and TTY_ONSET.

Changing TTY state from NON_TTY_MODE to TTY_ONSET requires that a "0" bit is followed by a "1" bit. This rule requires the presence of both the space tone and the mark tone and for the tones to be the correct duration. This test must be met in order to declare TTY_ONSET for the first time.

NON_TTY_MODE is declared by get_tty_state() whenever a non-TTY bit is detected or when a "0" bit occupies the bit preceding the start bit.

7.5.3 TTY/TDD Decoder Processing

The TTY decoder processing must recognize when TTY/TDD information is in the packet, recover from channel impairments to decode the TTY character being sent, and regenerate the Baudot tones corresponding to that character.

When the decoder is in NON_TTY_MODE, the packet is decoded in the usual manner for speech. When the decoder makes the transition to TTY_MODE, it mutes its output until it receives TTY character information or until it transitions back to NON_TTY_MODE.

7.5.3.1 TTY Decoder Inputs

- TTY information (header, TTY character, baud rate)

- Bad frame indicator

7.5.3.2 Decoding the TTY/TDD Information

The task of detecting the presence of TTY information, recovering from frame and bit errors, and decoding the TTY character is performed in tty_dec(). If the routine detects that TTY information is being sent, the tty_dec() flag is set to nonzero and the PCM buffer is filled with the appropriate Baudot tones. If TTY is not detected, the flag is set to zero and the PCM buffer is returned unmodified.

The routine labels each frame as NON_TTY, TTY_SILENCE, FER, or a TTY character, and maintains a history buffer of these classifications for 11 frames: 9 frames of lookahead, 1 current frame, and 1 frame of lookback (see Table 7.5.3.2-1). The most recent packet enters the buffer at location 0, but the decision for the current frame is based on the contents of element 9. The buffer is updated at the end of each frame, shifting its contents to the right by one. The buffer is initialized to NON_TTY.

Table 7.5.3.2-1 tty_dec() History Buffer

Frame	0	1	2	3	4	5	6	7	8	9	10
Description	Lookahead									Current Frame	Lookback

At the start of each frame, the most recent information is sanity checked to see if it is consistent with TTY information. If the frame erasure flag is set, the frame is labeled FER, otherwise, the TTY/TDD information is checked to see if the header and TTY character fields fall within the allowed range of values. If all of the tests pass, Frame 0 is labeled with the TTY character in the history buffer.

The TTY decoder processing can reliably regenerate the TTY characters despite channel impairments because the character information is transmitted a minimum of 6 times from the encoder. Errors are corrected by a voting process. The current frame and 9 frames of lookahead are used to determine the correct TTY header, character, and baud rate. Errors are replaced with the winner of the voting process.

Voting is conducted under the following conditions:

- Any time the current frame is labeled as FER.

- Every time the current frame contains information for the start of a new character. Because a new character must contain a minimum of 6 frames of the same information, a vote is taken to verify that the information is present before it will generate the tones for that character. Once a character wins the vote, any frame errors, bit errors, or other inconsistencies are corrected in the frame window where the character information is expected, i.e., the current frame and the adjacent frames of lookahead will contain the same header and character information.

- Any time the current frame contains TTY_SILENCE or a TTY character, and the frame of lookback contains NON_TTY. This makes it harder for the decoder to erroneously go into TTY_MODE, thus preventing false alarms when speech is present.

- Once tty_dec() makes its decision on the current frame, tty_dec() calls tty_gen() to generate the appropriate PCM samples.

7.5.3.3 Baudot Generator

Once the current frame is labeled by tty_dec(), tty_gen() is called to fill the PCM buffer with the appropriate Baudot tones. In the case of NON_TTY, the PCM buffer is returned unmodified. In the case of TTY_SILENCE, the PCM is muted.

Generating TTY characters is more involved because one character spans many frames, so tty_gen() must generate the Baudot tones one subframe at a time. When a TTY character needs to be regenerated, tty_gen() puts a subframe's worth of samples in the PCM buffer. It keeps track of which bit it is in the middle of generating and the number of samples left to generate for that bit, so that the next time it is called, it can pick up where it left off. Once tty_gen() begins to generate a character, it will generate the entire character before it will generate the next character. This is done so that the repeater will only generate valid TTY characters.

There exists logic in tty_gen() to detect when the next character arrives before the current one is finished. If the next character arrives before the current one can be regenerated, the bit duration is reduced by 4 samples and a minimum of 1 stop bit is generated. Otherwise, nominal bit lengths are generated, i.e., 22 ms bits for 45.45 baud and 20 ms bits for 50 baud, and a minimum of 1.5 stop bits is generated for 45.45 baud, and a minimum of 1.75 stop bits is generated for 50 baud.

There exists a provision in the [N11] for the TTY/TDD device to extend its stop bit in order to prevent a TTY/TDD device from detecting its own echo. This routine will extend the stop bit a maximum of 300 ms if a TTY character is followed by silence. If a new character arrives before 300 ms has elapsed, the extended stop bit is terminated and the new character is generated immediately.

The tones themselves are generated by tone_gen(). Before tty_gen() returns, it updates the decoder's lookback field in the TTY history buffer with the information corresponding to the last samples generated. For example, if tty_gen() finished generating a character in the middle of the subframe and started generating silence, the lookback field is updated with TTY_SILENCE.

7.5.3.4 Tone Generator

The routine tone_gen() is a sine wave generator. Given a frequency and the number of samples, it will generate the PCM samples by using a 2 tap marginally stable IIR filter. The filter implements the trigonometric identity:

$$\cos(k\omega) = 2 \cdot \cos(\omega) \cdot \cos((k-1)\omega) - \cos((k-2)\omega)$$

It is a zero excitation filter, using only its past 2 samples and the cosine of the frequency to be generated, to produce the next sample.

1 **7.6 DTMF**

2 **7.6.1 DTMF Detector**

3 The DTMF (Dual Tone Multiple Frequency) detector detects the presence and the information content of incoming
4 DTMF tone signals. Its row and its column frequency, listed in Table 7.6.1-1, uniquely identify each key on the
5 keypad. It outputs a 16-bit value associated with each of the 16 possible DTMF tone combinations.

6 NOTE: A row and a column tone is associated with each digit.

7 **Table 7.6.1-1 Touch-Tone Telephone Keypad**

		Column frequency group			
		1209 Hz	1336 Hz	1477 Hz	1633 Hz
Row frequency group	697 Hz	1	2	3	A
	770 Hz	4	5	6	B
	852 Hz	7	8	9	C
	941 Hz	*	0	#	D

8 **Description of the DTMF Detector:** The DTMF detector consists of three main functions:

9 • AGC – Normalizes the input frame.

10 • Goertzel – Performs spectral analysis at the 8 frequencies and at the second harmonic of the column frequencies
11 using an 80 point Goertzel algorithm. The Goertzel algorithm provides an efficient means of calculating the DFT
12 energy at a specific frequency (see Oppenheim and Schafer, Digital Signal Processing, Prentice-Hall). In
13 addition, a 160 point Goertzel algorithm is used at the first and fourth row frequency in order to satisfy the
14 requirements found in Bellcore document GR-506-CORE LSSGR: Signaling and Analog Interfaces, Issue 1,
15 June 1996.

16 • Check – Checks the spectral energies to determine if a valid DTMF tone is detected.

17 **Input:** Pointer to 16-bit linear time series for current 20 msec. frame and 10 msec. look-ahead.

18 **Output:** 16-bit valid DTMF tone digit if a tone is detected or 0xffff to indicate no DTMF present.

1

2

3

4

5

6   This page intentionally left blank.

1 ## 8 CODEBOOK MEMORIES AND CONSTANTS

2

3 **Table 8-1 LSP Quantization Table, Rate 1, Codebook 1**

j	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$	j	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$	j	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$
1	1.420163E-2	1.938816E-2	23	3.891726E-2	5.657889E-2	45	3.185912E-2	4.621231E-2
2	2.916675E-2	6.517492E-2	24	6.048006E-2	1.045370E-1	46	6.189098E-2	7.332318E-2
3	2.066932E-2	4.975649E-2	25	2.691566E-2	3.571689E-2	47	4.417184E-2	5.792409E-2
4	3.947198E-2	9.558509E-2	26	4.111172E-2	7.333230E-2	48	7.935962E-2	1.411774E-1
5	2.270125E-2	3.966258E-2	27	4.126607E-2	4.851652E-2	49	2.474123E-2	3.236294E-2
6	5.387895E-2	6.283478E-2	28	7.180496E-2	1.062024E-1	50	3.365639E-2	8.046506E-2
7	2.905255E-2	5.734358E-2	29	3.380379E-2	4.243004E-2	51	3.379437E-2	5.449772E-2
8	4.482806E-2	1.153646E-1	30	5.918182E-2	7.974680E-2	52	6.536490E-2	9.527759E-2
9	1.941107E-2	3.468897E-2	31	4.701079E-2	6.285638E-2	53	2.933642E-2	4.284110E-2
10	4.375030E-2	6.752285E-2	32	9.420119E-2	1.300532E-1	54	5.278705E-2	8.161594E-2
11	3.554973E-2	4.940868E-2	33	1.942443E-2	2.727323E-2	55	4.007249E-2	6.181447E-2
12	6.992199E-2	8.672798E-2	34	3.708317E-2	6.648982E-2	56	6.758486E-2	1.171961E-1
13	2.778802E-2	4.657485E-2	35	2.801364E-2	5.159849E-2	57	3.030650E-2	3.869141E-2
14	5.791110E-2	6.745425E-2	36	5.344610E-2	9.259042E-2	58	4.831063E-2	7.423830E-2
15	4.746644E-2	5.502715E-2	37	2.549592E-2	4.328448E-2	59	4.375483E-2	5.228423E-2
16	7.888989E-2	1.224430E-1	38	5.518607E-2	7.361823E-2	60	8.323100E-2	1.098820E-1
17	2.217159E-2	3.026288E-2	39	3.398511E-2	6.053291E-2	61	3.756006E-2	4.532172E-2
18	3.391345E-2	7.177040E-2	40	6.181821E-2	1.345813E-1	62	6.601132E-2	7.975802E-2
19	3.179891E-2	4.989961E-2	41	2.356692E-2	3.552420E-2	63	5.032251E-2	5.901763E-2
20	6.115560E-2	8.733612E-2	42	5.108042E-2	6.795625E-2	64	8.771333E-2	1.631874E-1
21	2.675065E-2	3.967359E-2	43	3.834650E-2	5.234694E-2			
22	4.441010E-2	8.267313E-2	44	7.442758E-2	9.661083E-2			

4

5 **Table 8-2 LSP Quantization Table, Rate 1, Codebook 2**

j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$	j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$	j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$
1	5.219596E-2	8.384457E-2	23	9.266608E-2	1.467255E-1	45	7.772978E-2	1.202872E-1
2	1.058741E-1	1.286942E-1	24	1.792855E-1	2.197060E-1	46	1.306480E-1	1.843318E-1
3	5.483239E-2	1.338429E-1	25	7.064585E-2	9.999245E-2	47	6.919396E-2	1.842180E-1
4	1.177685E-1	1.940373E-1	26	1.065005E-1	1.794434E-1	48	2.039041E-1	2.497152E-1
5	5.360865E-2	1.113987E-1	27	8.792497E-2	1.252877E-1	49	7.076717E-2	9.031861E-2
6	1.199897E-1	1.474747E-1	28	1.536402E-1	1.978527E-1	50	1.084716E-1	1.619665E-1
7	8.003736E-2	1.429997E-1	29	8.884301E-2	1.124657E-1	51	7.168864E-2	1.510932E-1
8	1.640866E-1	2.098218E-1	30	1.482867E-1	1.675170E-1	52	1.387795E-1	2.188018E-1
9	5.210592E-2	9.952294E-2	31	8.165681E-2	1.692740E-1	53	6.759071E-2	1.267403E-1
10	8.675680E-2	1.859665E-1	32	2.078105E-1	2.310336E-1	54	1.334124E-1	1.688389E-1
11	7.773411E-2	1.315069E-1	33	6.149280E-2	8.362632E-2	55	9.618226E-2	1.587287E-1
12	1.605455E-1	1.819303E-1	34	1.144733E-1	1.367800E-1	56	1.864856E-1	2.365609E-1

j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$	j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$	j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$
13	7.422437E-2	1.104371E-1	35	6.871299E-2	1.380991E-1	57	8.234471E-2	1.021260E-1
14	1.186351E-1	1.753068E-1	36	1.105114E-1	2.153529E-1	58	1.003366E-1	1.949185E-1
15	6.615578E-2	1.644419E-1	37	5.556523E-2	1.222428E-1	59	9.959820E-2	1.364251E-1
16	1.968109E-1	2.166820E-1	38	1.205576E-1	1.610725E-1	60	1.824485E-1	2.036552E-1
17	6.053178E-2	9.454086E-2	39	8.322497E-2	1.554755E-1	61	9.788907E-2	1.211455E-1
18	1.062714E-1	1.480139E-1	40	1.616385E-1	2.282689E-1	62	1.454531E-1	1.836045E-1
19	5.874866E-2	1.477246E-1	41	6.291523E-2	1.062296E-1	63	9.583955E-2	1.721949E-1
20	1.348165E-1	2.015180E-1	42	8.291869E-2	2.067745E-1	64	2.232959E-1	2.464186E-1
21	6.596983E-2	1.164474E-1	43	8.847569E-2	1.358000E-1			
22	1.322972E-1	1.532673E-1	44	1.697722E-1	1.937739E-1			

Table 8-3 LSP Quantization Table, Rate 1, Codebook 3

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$
1	1.364258E-1	1.686519E-1	2.046882E-1	257	1.476240E-1	1.812728E-1	2.047079E-1
2	1.857176E-1	2.287562E-1	2.519580E-1	258	1.937514E-1	2.209740E-1	2.617752E-1
3	1.227602E-1	1.859507E-1	2.794467E-1	259	1.320898E-1	1.948516E-1	2.835476E-1
4	1.964685E-1	2.644844E-1	2.893189E-1	260	2.077394E-1	2.705968E-1	2.922648E-1
5	1.256537E-1	1.505293E-1	2.761443E-1	261	1.277334E-1	1.668960E-1	2.838914E-1
6	1.963016E-1	2.417000E-1	2.882307E-1	262	2.053094E-1	2.478075E-1	2.836328E-1
7	1.400994E-1	2.223656E-1	2.746666E-1	263	1.542119E-1	2.250141E-1	2.700820E-1
8	2.599523E-1	2.753950E-1	3.109759E-1	264	2.675741E-1	2.844269E-1	3.093348E-1
9	1.584522E-1	1.885910E-1	2.073392E-1	265	1.688469E-1	1.870045E-1	2.024332E-1
10	1.956162E-1	2.213795E-1	2.870229E-1	266	2.024411E-1	2.167331E-1	2.930792E-1
11	1.694246E-1	2.016147E-1	2.756692E-1	267	1.636213E-1	2.156165E-1	2.827929E-1
12	2.123938E-1	2.642507E-1	3.179675E-1	268	2.255093E-1	2.662830E-1	3.178866E-1
13	1.829651E-1	1.995476E-1	2.295388E-1	269	1.891103E-1	2.056094E-1	2.221136E-1
14	2.152007E-1	2.624094E-1	2.824327E-1	270	2.212402E-1	2.602889E-1	2.925411E-1
15	1.464046E-1	2.369667E-1	2.900671E-1	271	1.555634E-1	2.468508E-1	2.896488E-1
16	2.453386E-1	3.033581E-1	3.422602E-1	272	2.484062E-1	3.052919E-1	3.553167E-1
17	1.374790E-1	1.582766E-1	2.392172E-1	273	1.271222E-1	1.580537E-2	2.541644E-1
18	2.019990E-1	2.201026E-1	2.695469E-1	274	2.049988E-1	2.194769E-1	2.783420E-1
19	1.183500E-1	2.302064E-1	2.835548E-1	275	1.333023E-1	2.296140E-1	2.869472E-1
20	2.255193E-1	2.722721E-1	3.060730E-1	276	2.367771E-1	2.679182E-1	3.082309E-1
21	1.356614E-1	1.916340E-1	2.659120E-1	277	1.408536E-1	2.034147E-1	2.732571E-1
22	1.957331E-1	2.319262E-1	3.143761E-1	278	2.076843E-1	2.345200E-1	3.245833E-1
23	1.679990E-1	2.277063E-1	2.769478E-1	279	1.771817E-1	2.295954E-1	2.835392E-1
24	2.501706E-1	3.016271E-1	3.210842E-1	280	2.613784E-1	3.011602E-1	3.217071E-1
25	1.334923E-1	2.012231E-1	2.338940E-1	281	1.485957E-1	2.077720E-1	2.469461E-1
26	2.064421E-1	2.387042E-1	2.775601E-1	282	2.143348E-1	2.480613E-1	2.722592E-1
27	1.790488E-1	1.957766E-1	2.806566E-1	283	1.763803E-1	1.968979E-1	2.922869E-1
28	2.061936E-1	2.640554E-1	3.330984E-1	284	1.981935E-1	2.754833E-1	3.490376E-1
29	1.751853E-1	1.911663E-1	2.575403E-1	285	1.761532E-1	1.932490E-1	2.695485E-1
30	2.283986E-1	2.452967E-1	3.089808E-1	286	2.369686E-1	2.500658E-1	3.068208E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$
31	1.808598E-1	2.435791E-1	2.966311E-1	287	1.760607E-1	2.540376E-1	3.035668E-1
32	2.761530E-1	3.082561E-1	3.468226E-1	288	2.829529E-1	3.017651E-1	3.539563E-1
33	1.371157E-1	1.800578E-1	2.209535E-1	289	1.453537E-1	1.836788E-1	2.347501E-1
34	1.813701E-1	2.267701E-1	2.703927E-1	290	1.938426E-1	2.306356E-1	2.678178E-1
35	1.252465E-1	1.796069E-1	3.103764E-1	291	1.389590E-1	1.867608E-1	3.131132E-1
36	1.907084E-1	2.877342E-1	3.134762E-1	292	1.999445E-1	2.776248E-1	3.250463E-1
37	1.304861E-1	1.604353E-1	3.002437E-1	293	1.429661E-1	1.713108E-1	3.030134E-1
38	1.973186E-1	2.563785E-1	2.784743E-1	294	2.077417E-1	2.586918E-1	2.887670E-1
39	1.585971E-1	2.373814E-1	2.629103E-1	295	1.717769E-1	2.402461E-1	2.732845E-1
40	2.618259E-1	2.777172E-1	3.313822E-1	296	2.710466E-1	2.851709E-1	3.274011E-1
41	1.641607E-1	1.858415E-1	2.356159E-1	297	1.698546E-1	1.875458E-1	2.244847E-1
42	2.094861E-1	2.214528E-1	2.921539E-1	298	2.152220E-1	2.273397E-1	2.950088E-1
43	1.668079E-1	2.136418E-1	2.706759E-1	299	1.755966E-1	2.179366E-1	2.748796E-1
44	2.298343E-1	2.883746E-1	3.062383E-1	300	2.346654E-1	2.895309E-1	3.164944E-1
45	1.821543E-1	2.008225E-1	2.401694E-1	301	1.899470E-1	2.049538E-1	2.469552E-1
46	2.249447E-1	2.698139E-1	2.914012E-1	302	2.372978E-1	2.683167E-1	2.906843E-1
47	1.639406E-1	2.503412E-1	2.783078E-1	303	1.699632E-1	2.533675E-1	2.925330E-1
48	2.567280E-1	2.951038E-1	3.532971E-1	304	2.706599E-1	2.971461E-1	3.561840E-1
49	1.402188E-1	1.766877E-1	2.467733E-1	305	1.525397E-1	1.701390E-1	2.527039E-1
50	2.152913E-1	2.292160E-1	2.642836E-1	306	2.191192E-1	2.359007E-1	2.697391E-1
51	1.210027E-1	2.183338E-1	3.223413E-1	307	1.422457E-1	2.181846E-1	3.282181E-1
52	2.542432E-1	2.739862E-1	2.962625E-1	308	2.614728E-1	2.780257E-1	3.023759E-1
53	1.603854E-1	1.837629E-1	2.815987E-1	309	1.535260E-1	1.907277E-1	2.928208E-1
54	1.878322E-1	2.374204E-1	3.297775E-1	310	2.092410E-1	2.498087E-1	3.247091E-1
55	1.777884E-1	2.267035E-1	3.023225E-1	311	1.751764E-1	2.386468E-1	3.063927E-1
56	2.751082E-1	2.937306E-1	3.123738E-1	312	2.732189E-1	3.039550E-1	3.205139E-1
57	1.701164E-1	1.852321E-1	2.461250E-1	313	1.639116E-1	1.896116E-1	2.562725E-1
58	2.217548E-1	2.399122E-1	2.868919E-1	314	2.269538E-1	2.401202E-1	2.927285E-1
59	1.950837E-1	2.083379E-1	2.883497E-1	315	1.955657E-1	2.119562E-1	2.973747E-1
60	2.375365E-1	2.750045E-1	3.397860E-1	316	2.410456E-1	2.884970E-1	3.363523E-1
61	1.883693E-1	2.043718E-1	2.573750E-1	317	1.949483E-1	2.094753E-1	2.563097E-1
62	2.472502E-1	2.605518E-1	3.021375E-1	318	2.478846E-1	2.633564E-1	3.112709E-1
63	1.669442E-1	2.469124E-1	3.188944E-1	319	1.691897E-1	2.358646E-1	3.362494E-1
64	2.781186E-1	3.130111E-1	3.653293E-1	320	2.860016E-1	3.254238E-1	3.596074E-1
65	1.452135E-1	1.630515E-1	2.249126E-1	321	1.562586E-1	1.767049E-1	2.143934E-1
66	2.056925E-1	2.208315E-1	2.528178E-1	322	2.089969E-1	2.239687E-1	2.608868E-1
67	1.211257E-1	1.963741E-1	3.001227E-1	323	1.357654E-1	2.035801E-1	3.055032E-1
68	2.155668E-1	2.656573E-1	2.992029E-1	324	2.189614E-1	2.794635E-1	2.994508E-1
69	1.091342E-1	1.784721E-1	2.883232E-1	325	1.340648E-1	1.783321E-1	2.901696E-1
70	2.035085E-1	2.403479E-1	2.963097E-1	326	2.132984E-1	2.400315E-1	3.003459E-1
71	1.531018E-1	2.254153E-1	2.848437E-1	327	1.643734E-1	2.264387E-1	2.871712E-1
72	2.502334E-1	2.777369E-1	3.248407E-1	328	2.507396E-1	2.808125E-1	3.353494E-1
73	1.663089E-1	1.941734E-1	2.116354E-1	329	1.636495E-1	1.971080E-1	2.211652E-1
74	2.012895E-1	2.260622E-1	2.932465E-1	330	2.081396E-1	2.308698E-1	2.961371E-1
75	1.495188E-1	2.142017E-1	2.838948E-1	331	1.591131E-1	2.181892E-1	2.955320E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$
76	2.218361E-1	2.852315E-1	3.200826E-1	332	2.398835E-1	2.818312E-1	3.260456E-1
77	1.895732E-1	2.065776E-1	2.303323E-1	333	1.893947E-1	2.081271E-1	2.384464E-1
78	2.312477E-1	2.468643E-1	2.898466E-1	334	2.329957E-1	2.596035E-1	2.934280E-1
79	1.391169E-1	2.591899E-1	2.980196E-1	335	1.605588E-1	2.551648E-1	3.028729E-1
80	2.445126E-1	2.826714E-1	3.612583E-1	336	2.535093E-1	2.960285E-1	3.677216E-1
81	1.225310E-1	1.685148E-1	2.708793E-1	337	1.301244E-1	1.748390E-1	2.604860E-1
82	2.043728E-1	2.303984E-1	2.717929E-1	338	2.102040E-1	2.335708E-1	2.830619E-1
83	1.426439E-1	2.224056E-1	2.920572E-1	339	1.523655E-1	2.253388E-1	3.037210E-1
84	2.426437E-1	2.774294E-1	2.971355E-1	340	2.405586E-1	2.771922E-1	3.058919E-1
85	1.520486E-1	1.969211E-1	2.610132E-1	341	1.637288E-1	1.947794E-1	2.692536E-1
86	2.178750E-1	2.458404E-1	3.081386E-1	342	2.257094E-1	2.409027E-1	3.180606E-1
87	1.901093E-1	2.310991E-1	2.801782E-1	343	1.920551E-1	2.298578E-1	2.898267E-1
88	2.543142E-1	2.940798E-1	3.396492E-1	344	2.627597E-1	3.042922E-1	3.356806E-1
89	1.566986E-1	2.085975E-1	2.280108E-1	345	1.660712E-1	2.068192E-1	2.397125E-1
90	2.250887E-1	2.500145E-1	2.762502E-1	346	2.239156E-1	2.501069E-1	2.852962E-1
91	1.782190E-1	1.982282E-1	3.041989E-1	347	1.884023E-1	2.037937E-1	3.030411E-1
92	2.085672E-1	2.923954E-1	3.467869E-1	348	2.306990E-1	2.870441E-1	3.498028E-1
93	1.710521E-1	2.034388E-1	2.626443E-1	349	1.820254E-1	2.140735E-1	2.634700E-1
94	2.302755E-1	2.588175E-1	3.119865E-1	350	2.372978E-1	2.650254E-1	3.178155E-1
95	1.853336E-1	2.457602E-1	3.105540E-1	351	1.892787E-1	2.588022E-1	3.048662E-1
96	2.894139E-1	3.110956E-1	3.464762E-1	352	2.972431E-1	3.171531E-1	3.565839E-1
97	1.503324E-1	1.675382E-1	2.401829E-1	353	1.586075E-1	1.786598E-1	2.419194E-1
98	1.799717E-1	2.371686E-1	2.608997E-1	354	1.948874E-1	2.416959E-1	2.621767E-1
99	1.498662E-1	1.978901E-1	3.079166E-1	355	1.581244E-1	2.117531E-1	3.113522E-1
100	2.107997E-1	2.881801E-1	3.297472E-1	356	2.169027E-1	2.987968E-1	3.209941E-1
101	1.317111E-1	1.659065E-1	3.228980E-1	357	1.492728E-1	1.749641E-1	3.153344E-1
102	2.148320E-1	2.528221E-1	2.975471E-1	358	2.216223E-1	2.561791E-1	3.039030E-1
103	1.837604E-1	2.375236E-1	2.746100E-1	359	1.759796E-1	2.435055E-1	2.858017E-1
104	2.555752E-1	2.754392E-1	3.460219E-1	360	2.645904E-1	2.855416E-1	3.451078E-1
105	1.826622E-1	1.994709E-1	2.160517E-1	361	1.801371E-1	2.052794E-1	2.222560E-1
106	2.092403E-1	2.224067E-1	3.023829E-1	362	2.107962E-1	2.263154E-1	3.144269E-1
107	1.840883E-1	2.113278E-1	2.825381E-1	363	1.791512E-1	2.094397E-1	2.932809E-1
108	2.411711E-1	2.970360E-1	3.159793E-1	364	2.497190E-1	2.912577E-1	3.271623E-1
109	1.968047E-1	2.118159E-1	2.416477E-1	365	1.987002E-1	2.158968E-1	2.499602E-1
110	2.427620E-1	2.585866E-1	2.932044E-1	366	2.407264E-1	2.648577E-1	2.996396E-1
111	1.589055E-1	2.650770E-1	2.898813E-1	367	1.712497E-1	2.681662E-1	3.035727E-1
112	2.580606E-1	3.189032E-1	3.478468E-1	368	2.695556E-1	3.161006E-1	3.565707E-1
113	1.487664E-1	1.668539E-1	2.668274E-1	369	1.505648E-1	1.841909E-1	2.686748E-1
114	2.159423E-1	2.299383E-1	2.760416E-1	370	2.169412E-1	2.408140E-1	2.789422E-1
115	1.384105E-1	2.392834E-1	3.279724E-1	371	1.353995E-1	2.605865E-1	3.326049E-1
116	2.437653E-1	2.884085E-1	3.060487E-1	372	2.561510E-1	2.878229E-1	3.061564E-1
117	1.701571E-1	1.899863E-1	2.812192E-1	373	1.663988E-1	1.887218E-1	2.930237E-1
118	2.191170E-1	2.580053E-1	3.265720E-1	374	2.292141E-1	2.615654E-1	3.274941E-1
119	1.921636E-1	2.236142E-1	2.986831E-1	375	1.982666E-1	2.329705E-1	2.991343E-1
120	2.735454E-1	3.120781E-1	3.307666E-1	376	2.870463E-1	3.071038E-1	3.272981E-1

j	$q_{rate}(3,1,j)$	$q_{rate}(3,2,j)$	$q_{rate}(3,3,j)$	*j*	$q_{rate}(3,1,j)$	$q_{rate}(3,2,j)$	$q_{rate}(3,3,j)$
121	1.624521E-1	2.049309E-1	2.533377E-1	377	1.758987E-1	2.118986E-1	2.513329E-1
122	2.238553E-1	2.376711E-1	3.032020E-1	378	2.320674E-1	2.446222E-1	2.994437E-1
123	1.939553E-1	2.123356E-1	3.075669E-1	379	1.907801E-1	2.120900E-1	3.250593E-1
124	2.299127E-1	2.975811E-1	3.374992E-1	380	2.315312E-1	3.141661E-1	3.427359E-1
125	1.893354E-1	2.041481E-1	2.786098E-1	381	1.950999E-1	2.095543E-1	2.794835E-1
126	2.423036E-1	2.731631E-1	3.153617E-1	382	2.404161E-1	2.696048E-1	3.280155E-1
127	1.550097E-1	2.880952E-1	3.359964E-1	383	1.718009E-1	2.822331E-1	3.147493E-1
128	2.737162E-1	3.312155E-1	3.625391E-1	384	2.692438E-1	3.384625E-1	3.799357E-1
129	1.523894E-1	1.726191E-1	1.905857E-1	385	1.599346E-1	1.779668E-1	2.008186E-1
130	1.969883E-1	2.263098E-1	2.461975E-1	386	2.019797E-1	2.306685E-1	2.567733E-1
131	1.205552E-1	2.063698E-1	2.811998E-1	387	1.340243E-1	2.109616E-1	2.846877E-1
132	1.937094E-1	2.719005E-1	3.013329E-1	388	2.037129E-1	2.830531E-1	3.033094E-1
133	1.367012E-1	1.540932E-1	2.822584E-1	389	1.445289E-1	1.647281E-1	2.850794E-1
134	1.972992E-1	2.536563E-1	2.903151E-1	390	2.062856E-1	2.486490E-1	2.963831E-1
135	1.434638E-1	2.438729E-1	2.755337E-1	391	1.581382E-1	2.343177E-1	2.796500E-1
136	2.584773E-1	2.732799E-1	3.211191E-1	392	2.649956E-1	2.799007E-1	3.186194E-1
137	1.544062E-1	1.937935E-1	2.158842E-1	393	1.665375E-1	1.842794E-1	2.145475E-1
138	2.059795E-1	2.242770E-1	2.857324E-1	394	2.030519E-1	2.351105E-1	2.887560E-1
139	1.745353E-1	2.084824E-1	2.796685E-1	395	1.684227E-1	2.039462E-1	2.874789E-j
140	2.188446E-1	2.724863E-1	3.270956E-1	396	2.317270E-1	2.740864E-1	3.247552E-1
141	1.776097E-1	2.129902E-1	2.391197E-1	397	1.853562E-1	2.141131E-1	2.290304E-1
142	2.291638E-1	2.591659E-1	2.835147E-1	398	2.424826E-1	2.606555E-1	2.830303E-1
143	1.573532E-1	2.399613E-1	3.042631E-1	399	1.675623E-1	2.420275E-1	2.994620E-1
144	2.456138E-1	3.168245E-1	3.429094E-1	400	2.388099E-1	3.190039E-1	3.584159E-1
145	1.429532E-1	1.619054E-1	2.537102E-1	401	1.379083E-1	1.547878E-1	2.656112E-1
146	2.101928E-1	2.228477E-1	2.711038E-1	402	2.110193E-1	2.246073E-1	2.799547E-1
147	1.268439E-1	2.167091E-1	2.977347E-1	403	1.375699E-1	2.251285E-1	3.093129E-1
148	2.310001E-1	2.801091E-1	2.997074E-1	404	2.292399E-1	2.761510E-1	3.152418E-1
149	1.529805E-1	1.939969E-1	2.728957E-1	405	1.604875E-1	1.954612E-1	2.831695E-1
150	2.128607E-1	2.415454E-1	3.165188E-1	406	2.185057E-1	2.381972E-1	3.303401E-1
151	1.711547E-1	2.224697E-1	2.937865E-1	407	1.819913E-1	2.330270E-1	2.932760E-1
152	2.519882E-1	3.042550E-1	3.312700E-1	408	2.545523E-1	3.143942E-1	3.363923E-1
153	1.331889E-1	2.079250E-1	2.553621E-1	409	1.440958E-1	2.266402E-1	2.505951E-1
154	2.120449E-1	2.421897E-1	2.889037E-1	410	2.151880E-1	2.514173E-1	2.850440E-1
155	1.846125E-1	2.011436E-1	2.863608E-1	411	1.876744E-1	2.044589E-1	2.941690E-1
156	2.182867E-1	2.767524E-1	3.445815E-1	412	2.304948E-1	2.684524E-1	3.523701E-1
157	1.835622E-1	1.994785E-1	2.621566E-1	413	1.850221E-1	1.990753E-1	2.719306E-1
158	2.331305E-1	2.495969E-1	3.158428E-1	414	2.425694E-1	2.553892E-1	3.113993E-1
159	1.898990E-1	2.468749E-1	2.971325E-1	415	1.951661E-1	2.491021E-1	2.989985E-1
160	2.750225E-1	3.224903E-1	3.469777E-1	416	2.836542E-1	3.146003E-1	3.556194E-1
161	1.423053E-1	1.926892E-1	2.161559E-1	417	1.514900E-1	1.977298E-1	2.324675E-1
162	1.956762E-1	2.222686E-1	2.765874E-1	418	2.000299E-1	2.301013E-1	2.819339E-1
163	1.332415E-1	1.977918E-1	3.228979E-1	419	1.387113E-1	1.918166E-1	3.457804E-1
164	1.848651E-1	2.971062E-1	3.261052E-1	420	1.965804E-1	3.047148E-1	3.405534E-1
165	1.502037E-1	1.767813E-1	2.915362E-1	421	1.381543E-1	1.885431E-1	2.994612E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$
166	2.031445E-1	2.596162E-1	2.991560E-1	422	2.056665E-1	2.689049E-1	3.055372E-1
167	1.654890E-1	2.383421E-1	2.874939E-1	423	1.724479E-1	2.335584E-1	2.936252E-1
168	2.710713E-1	2.895445E-1	3.195210E-1	424	2.701454E-1	2.986548E-1	3.285564E-1
169	1.685984E-1	1.988256E-1	2.303476E-1	425	1.754894E-1	1.913616E-1	2.355853E-1
170	2.138117E-1	2.344718E-1	2.909596E-1	426	2.205488E-1	2.347740E-1	2.953977E-1
171	1.746054E-1	2.172560E-1	2.856881E-1	427	1.856524E-1	2.223491E-1	2.798839E-1
172	2.285035E-1	2.961903E-1	3.165347E-1	428	2.294570E-1	3.045463E-1	3.246843E-1
173	1.871726E-1	2.205474E-1	2.396887E-1	429	1.869008E-1	2.154694E-1	2.518568E-1
174	2.288848E-1	2.635832E-1	3.013295E-1	430	2.349105E-1	2.712174E-1	2.998947E-1
175	1.778971E-1	2.581315E-1	2.814877E-1	431	1.851424E-1	2.560710E-1	2.932913E-1
176	2.595136E-1	3.072044E-1	3.487936E-1	432	2.638837E-1	3.071275E-1	3.625467E-1
177	1.452244E-1	1.787160E-1	2.591870E-1	433	1.609976E-1	1.789379E-1	2.558083E-1
178	2.190623E-1	2.382235E-1	2.604615E-1	434	2.256711E-1	2.437351E-1	2.686250E-1
179	1.436509E-1	2.097608E-1	3.158302E-1	435	1.550762E-1	2.303962E-1	3.210056E-1
180	2.501275E-1	2.791823E-1	3.051536E-1	436	2.517605E-1	2.796534E-1	3.142022E-1
181	1.489864E-1	2.012268E-1	2.825437E-1	437	1.569888E-1	2.074669E-1	2.899340E-1
182	2.083878E-1	2.356039E-1	3.453639E-1	438	2.174795E-1	2.596264E-1	3.406591E-1
183	1.858303E-1	2.216073E-1	3.107736E-1	439	1.768115E-1	2.310871E-1	3.175625E-1
184	2.809047E-1	2.954698E-1	3.254995E-1	440	2.829526E-1	2.998444E-1	3.368229E-j
185	1.729673E-1	1.970781E-1	2.458011E-1	441	1.820603E-1	1.987347E-1	2.519803E-1
186	2.194957E-1	2.447671E-1	2.935878E-1	442	2.258742E-1	2.524692E-1	2.933564E-1
187	1.839096E-1	2.150043E-1	3.003345E-1	443	2.008000E-1	2.177869E-1	3.022101E-1
188	2.453386E-1	2.685953E-1	3.483304E-1	444	2.474238E-1	2.868829E-1	3.478206E-1
189	1.929574E-1	2.066251E-1	2.673364E-1	445	2.011281E-1	2.147469E-1	2.622697E-1
190	2.548456E-1	2.686423E-1	3.035479E-1	446	2.539634E-1	2.694780E-1	3.121338E-1
191	1.768531E-1	2.593310E-1	3.162008E-1	447	1.910349E-1	2.557382E-1	3.325596E-1
192	2.909291E-1	3.156348E-1	3.687235E-1	448	2.910537E-1	3.314584E-1	3.685885E-1
193	1.571170E-1	1.735529E-1	2.287365E-1	449	1.572299E-1	1.853741E-1	2.253613E-1
194	2.125093E-1	2.305012E-1	2.522180E-1	450	2.080513E-1	2.383509E-1	2.642129E-1
195	1.425218E-1	2.019799E-1	2.930122E-1	451	1.468483E-1	2.130001E-1	3.001926E-1
196	2.149197E-1	2.780651E-1	3.141761E-1	452	2.186306E-1	2.902638E-1	3.090458E-1
197	1.359473E-1	1.810559E-1	2.754754E-1	453	1.436992E-1	1.878152E-1	2.837699E-1
198	1.984167E-1	2.416738E-1	3.051734E-1	454	2.073280E-1	2.450887E-1	3.089564E-1
199	1.595173E-1	2.315801E-1	2.954125E-1	455	1.642281E-1	2.278267E-1	3.089079E-1
200	2.582036E-1	2.873481E-1	3.203520E-1	456	2.619197E-1	2.913337E-1	3.315280E-1
201	1.748407E-1	1.928833E-1	2.112500E-1	457	1.706489E-1	2.021575E-1	2.178278E-1
202	2.021685E-1	2.270257E-1	3.048841E-1	458	2.077961E-1	2.347048E-1	3.067838E-1
203	1.695321E-1	2.118262E-1	2.973554E-1	459	1.721188E-1	2.140574E-1	3.101518E-1
204	2.300337E-1	2.915044E-1	3.265894E-1	460	2.291162E-1	2.809499E-1	3.337743E-1
205	1.950461E-1	2.117092E-1	2.277058E-1	461	1.966222E-1	2.166531E-1	2.332797E-1
206	2.379269E-1	2.524116E-1	2.977522E-1	462	2.377892E-1	2.589713E-1	3.046092E-1
207	1.537629E-1	2.465416E-1	3.147689E-1	463	1.551820E-1	2.630326E-1	3.189431E-1
208	2.360757E-1	3.035689E-1	3.706245E-1	464	2.493888E-1	3.169709E-1	3.777625E-1
209	1.386603E-1	1.679500E-1	2.735153E-1	465	1.513636E-1	1.750107E-1	2.782458E-1
210	2.138062E-1	2.272672E-1	2.862763E-1	466	2.198102E-1	2.323602E-1	2.850349E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$
211	1.250806E-1	2.440984E-1	3.025488E-1	467	1.426306E-1	2.406029E-1	3.041251E-1
212	2.357149E-1	2.812088E-1	3.089037E-1	468	2.427649E-1	2.837621E-1	3.154812E-1
213	1.516914E-1	2.108778E-1	2.638130E-1	469	1.574675E-1	2.075241E-1	2.756749E-1
214	2.207304E-1	2.527779E-1	3.164137E-1	470	2.287586E-1	2.490922E-1	3.281394E-1
215	1.849247E-1	2.394248E-1	2.851208E-1	471	1.908727E-1	2.381252E-1	2.948946E-1
216	2.595485E-1	3.098099E-1	3.264237E-1	472	2.663893E-1	3.143214E-1	3.386695E-1
217	1.629307E-1	2.199009E-1	2.361486E-1	473	1.706442E-1	2.259800E-1	2.473724E-1
218	2.341942E-1	2.499443E-1	2.775491E-1	474	2.364428E-1	2.530035E-1	2.882204E-1
219	1.708702E-1	1.982916E-1	3.214126E-1	475	1.854238E-1	2.048889E-1	3.146088E-1
220	2.315669E-1	2.750151E-1	3.697104E-1	476	2.173790E-1	2.945536E-1	3.678310E-1
221	1.800024E-1	2.067010E-1	2.712049E-1	477	1.885640E-1	2.151743E-1	2.729997E-1
222	2.380753E-1	2.540062E-1	3.238276E-1	478	2.451021E-1	2.597704E-1	3.218856E-1
223	1.991483E-1	2.542739E-1	3.074797E-1	479	1.984442E-1	2.611607E-1	3.170979E-1
224	2.874286E-1	3.250451E-1	3.486346E-1	480	2.990139E-1	3.289653E-1	3.566812E-1
225	1.452850E-1	1.913592E-1	2.496914E-1	481	1.582488E-1	1.922057E-1	2.460591E-1
226	1.946593E-1	2.408212E-1	2.773027E-1	482	2.023854E-1	2.479658E-1	2.717497E-1
227	1.531510E-1	1.943757E-1	3.275504E-1	483	1.617108E-1	2.137081E-1	3.273847E-1
228	2.040858E-1	2.985957E-1	3.214801E-1	484	2.144197E-1	3.055525E-1	3.337216E-1
229	1.560097E-1	1.810127E-1	3.009317E-1	485	1.618200E-1	1.898973E-1	3.105016E-1
230	2.109624E-1	2.557703E-1	3.080861E-1	486	2.194363E-1	2.650296E-1	3.092888E-1
231	1.854441E-1	2.490213E-1	2.740298E-1	487	1.883039E-1	2.496332E-1	2.854995E-1
232	2.744935E-1	2.894420E-1	3.387949E-1	488	2.693254E-1	2.998070E-1	3.417225E-1
233	1.769419E-1	1.944769E-1	2.220773E-1	489	1.724060E-1	2.109773E-1	2.277732E-1
234	2.163775E-1	2.307358E-1	3.036893E-1	490	2.202815E-1	2.340158E-1	3.128461E-1
235	1.896835E-1	2.146608E-1	2.884454E-1	491	1.832573E-1	2.220620E-1	2.910524E-1
236	2.408273E-1	2.981418E-1	3.273784E-1	492	2.425312E-1	3.095276E-1	3.303897E-1
237	2.017878E-1	2.194418E-1	2.393275E-1	493	2.075467E-1	2.246626E-1	2.444201E-1
238	2.488125E-1	2.658659E-1	2.933824E-1	494	2.458582E-1	2.702860E-1	3.051321E-1
239	1.820278E-1	2.682791E-1	2.939914E-1	495	1.848406E-1	2.720968E-1	3.125311E-1
240	2.564986E-1	3.199845E-1	3.626632E-1	496	2.742526E-1	3.212524E-1	3.746582E-1
241	1.587993E-1	1.754337E-1	2.673899E-1	497	1.664258E-1	1.844916E-1	2.682781E-1
242	2.242593E-1	2.366683E-1	2.776391E-1	498	2.284237E-1	2.430254E-1	2.811849E-1
243	1.492034E-1	2.265853E-1	3.452556E-1	499	1.600913E-1	2.529533E-1	3.358223E-1
244	2.506558E-1	2.922649E-1	3.135743E-1	500	2.621100E-1	2.955819E-1	3.133541E-1
245	1.580963E-1	2.021932E-1	2.987117E-1	501	1.677028E-1	2.015369E-1	3.018016E-1
246	2.288209E-1	2.485573E-1	3.447265E-1	502	2.378230E-1	2.598948E-1	3.382311E-1
247	1.879721E-1	2.341094E-1	3.042356E-1	503	1.972062E-1	2.454909E-1	3.178954E-1
248	2.856571E-1	3.148781E-1	3.369315E-1	504	2.984553E-1	3.192098E-1	3.409717E-1
249	1.626800E-1	2.178201E-1	2.574368E-1	505	1.711953E-1	2.243278E-1	2.627361E-1
250	2.240498E-1	2.467398E-1	3.007959E-1	506	2.306269E-1	2.533102E-1	3.012068E-1
251	2.013546E-1	2.182867E-1	3.130363E-1	507	2.048142E-1	2.218816E-1	3.259666E-1
252	2.380285E-1	2.981035E-1	3.535038E-1	508	2.229875E-1	3.063391E-1	3.507172E-1
253	1.988300E-1	2.128771E-1	2.729808E-1	509	2.008554E-1	2.153599E-1	2.841435E-1
254	2.506165E-1	2.676600E-1	3.206119E-1	510	2.509517E-1	2.661893E-1	3.333606E-1
255	1.709018E-1	2.693304E-1	3.344282E-1	511	1.756103E-1	2.937913E-1	3.403269E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	*j*	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$
256	3.049889E-1	3.361967E-1	3.652354E-1	512	2.917451E-1	3.406025E-1	3.813972E-1

1

2

Table 8-4 LSP Quantization Table, Rate 1, Codebook 4

j	$q_{rate}(4, 1, j)$	$q_{rate}(4, 2, j)$	$q_{rate}(4, 3, j)$	*j*	$q_{rate}(4, 1, j)$	$q_{rate}(4, 2, j)$	$q_{rate}(4, 3, j)$
1	2.774615E-1	3.169721E-1	3.954983E-1	65	2.491101E-1	3.256969E-1	4.117283E-1
2	3.365604E-1	3.601570E-1	3.814730E-1	66	3.459292E-1	3.685775E-1	3.884733E-1
3	3.105093E-1	3.317324E-1	3.668644E-1	67	3.132197E-1	3.392295E-1	3.875979E-1
4	3.374710E-1	3.967953E-1	4.123563E-1	68	3.514540E-1	3.987303E-1	4.126562E-1
5	2.796604E-1	3.665201E-1	3.853135E-1	69	2.934871E-1	3.757631E-1	3.944881E-1
6	3.160390E-1	3.856093E-1	4.013048E-1	70	3.244708E-1	3.942029E-1	4.088827E-1
7	3.099604E-1	3.434107E-1	4.247455E-1	71	3.127108E-1	3.577203E-1	4.140612E-1
8	3.542436E-1	4.086993E-1	4.221680E-1	72	3.665072E-1	4.081713E-1	4.238914E-1
9	2.955872E-1	3.337411E-1	3.874217E-1	73	2.999657E-1	3.319934E-1	4.078602E-1
10	3.334464E-1	3.869748E-1	4.013531E-1	74	3.349252E-1	3.861430E-1	4.115381E-1
11	3.234128E-1	3.652697E-1	3.851933E-1	75	3.347880E-1	3.661962E-1	3.933471E-1
12	3.427320E-1	4.031925E-1	4.199204E-1	76	3.478479E-1	4.059265E-1	4.305073E-1
13	2.776818E-1	3.824950E-1	4.042742E-1	77	2.859529E-1	3.952833E-1	4.161193E-1
14	3.182480E-1	3.959853E-1	4.313532E-1	78	3.238674E-1	4.064767E-1	4.424828E-1
15	3.037114E-1	3.803197E-1	4.371736E-1	79	3.167167E-1	3.844516E-1	4.394110E-1
16	3.782888E-1	4.070773E-1	4.226791E-1	80	3.867729E-1	4.118246E-1	4.278315E-1
17	2.381165E-1	3.424543E-1	4.246247E-1	81	2.380724E-1	3.623424E-1	4.309317E-1
18	3.456157E-1	3.686811E-1	4.008173E-1	82	3.464500E-1	3.790829E-1	4.065678E-1
19	3.176881E-1	3.419027E-1	4.056018E-1	83	3.165766E-1	3.564686E-1	3.962183E-1
20	3.663690E-1	3.890399E-1	4.061545E-1	84	3.665392E-1	3.895909E-1	4.210556E-1
21	2.993980E-1	3.520217E-1	3.999557E-1	85	3.082914E-1	3.713243E-1	4.078674E-1
22	3.249919E-1	3.900288E-1	4.194787E-1	86	3.364352E-1	3.915144E-1	4.229771E-1
23	3.230258E-1	3.681143E-1	4.020878E-1	87	3.230355E-1	3.804473E-1	4.095502E-1
24	3.623263E-1	4.169280E-1	4.327737E-1	88	3.652281E-1	4.279104E-1	4.436913E-1
25	2.726964E-1	3.592050E-1	4.268807E-1	89	2.720380E-1	3.765968E-1	4.336859E-1
26	3.465399E-1	3.696166E-1	4.156212E-1	90	3.576658E-1	3.777616E-1	4.091790E-1
27	3.341091E-1	3.557363E-1	3.967496E-1	91	3.364986E-1	3.642159E-1	4.092555E-1
28	3.374690E-1	4.103927E-1	4.259868E-1	92	3.480824E-1	4.176318E-1	4.332845E-1
29	2.994690E-1	3.806483E-1	4.182841E-1	93	3.027545E-1	3.959748E-1	4.337173E-1
30	3.213782E-1	4.111980E-1	4.287925E-1	94	3.316763E-1	4.175872E-1	4.362398E-1
31	3.278412E-1	3.693451E-1	4.343956E-1	95	3.332876E-1	3.807991E-1	4.396207E-1
32	3.806691E-1	4.260864E-1	4.427546E-1	96	3.881120E-1	4.369336E-1	4.508293E-1
33	2.689437E-1	3.429430E-1	3.986815E-1	97	2.560266E-1	3.480152E-1	4.229226E-1
34	3.381029E-1	3.763388E-1	3.920432E-1	98	3.457740E-1	3.817258E-1	3.967941E-1
35	3.235935E-1	3.487421E-1	3.725520E-1	99	3.256238E-1	3.503919E-1	3.873307E-1
36	3.475508E-1	3.928854E-1	4.211699E-1	100	3.568681E-1	3.985748E-1	4.231772E-1
37	3.041828E-1	3.598167E-1	3.816333E-1	101	3.012262E-1	3.869070E-1	4.033356E-1
38	3.142214E-1	4.021086E-1	4.200853E-1	102	3.281784E-1	4.020902E-1	4.193893E-1
39	3.013066E-1	3.626627E-1	4.292628E-1	103	3.143854E-1	3.690439E-1	4.343753E-1

j	$q_{rate}(4, 1, j)$	$q_{rate}(4, 2, j)$	$q_{rate}(4, 3, j)$	j	$q_{rate}(4, 1, j)$	$q_{rate}(4, 2, j)$	$q_{rate}(4, 3, j)$
40	3.717703E-1	3.986964E-1	4.314390E-1	104	3.723211E-1	4.116724E-1	4.405187E-1
41	2.745913E-1	3.355955E-1	4.200797E-1	105	2.904797E-1	3.481219E-1	4.262165E-1
42	3.445408E-1	3.904518E-1	4.064121E-1	106	3.444388E-1	3.826664E-1	4.173211E-1
43	3.252398E-1	3.783445E-1	3.946733E-1	107	3.348668E-1	3.762357E-1	4.044752E-1
44	3.566835E-1	3.905742E-1	4.338511E-1	108	3.590254E-1	4.047219E-1	4.348384E-1
45	2.635013E-1	3.952601E-1	4.231164E-1	109	2.791280E-1	4.111066E-1	4.353606E-1
46	3.375207E-1	3.925635E-1	4.434158E-1	110	3.481255E-1	3.987321E-1	4.469274E-1
47	3.145223E-1	3.809686E-1	4.226764E-1	111	3.270189E-1	3.901073E-1	4.417075E-1
48	3.762351E-1	4.172987E-1	4.314513E-1	112	3.908584E-1	4.198139E-1	4.351535E-1
49	2.618550E-1	3.686461E-1	4.042606E-1	113	2.553193E-1	3.704060E-1	4.321886E-1
50	3.555802E-1	3.779945E-1	3.958682E-1	114	3.546520E-1	3.883327E-1	4.029561E-1
51	3.277428E-1	3.538728E-1	4.110406E-1	115	3.216082E-1	3.544898E-1	4.282998E-1
52	3.629606E-1	3.994670E-1	4.146902E-1	116	3.751635E-1	3.988340E-1	4.141774E-1
53	3.094109E-1	3.737961E-1	3.926725E-1	117	3.119536E-1	3.914307E-1	4.125525E-1
54	3.310163E-1	4.008016E-1	4.317593E-1	118	3.425288E-1	3.963653E-1	4.324974E-1
55	3.235731E-1	3.686196E-1	4.174551E-1	119	3.337444E-1	3.764224E-1	4.205370E-1
56	3.491159E-1	4.268401E-1	4.439140E-1	120	3.535291E-1	4.292311E-1	4.596993E-1
57	2.897386E-1	3.637593E-1	4.105118E-1	121	2.880179E-1	3.780000E-1	4.340117E-1
58	3.552865E-1	3.893313E-1	4.134324E-1	122	3.556835E-1	3.807800E-1	4.231455E-1
59	3.365659E-1	3.602225E-1	4.241790E-1	123	3.443583E-1	3.721849E-1	4.312654E-1
60	3.399327E-1	4.092288E-1	4.401849E-1	124	3.539661E-1	4.141667E-1	4.429413E-1
61	3.008897E-1	4.000811E-1	4.179557E-1	125	3.047702E-1	4.125175E-1	4.341831E-1
62	3.170521E-1	4.222881E-1	4.422296E-1	126	3.359134E-1	4.245908E-1	4.463785E-1
63	3.273368E-1	3.843117E-1	4.302886E-1	127	3.437382E-1	3.847662E-1	4.352714E-1
64	3.989909E-1	4.294984E-1	4.434752E-1	128	4.109413E-1	4.406630E-1	4.521134E-1

Table 8-5 LSP Quantization Table, Rate 1/2, Codebook 1

j	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$	$q_{rate}(1, 3, j)$	j	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$	$q_{rate}(1, 3, j)$
1	1.352263E-2	1.820813E-2	3.939407E-2	65	2.104905E-2	2.918910E-2	4.600358E-2
2	2.293929E-2	3.578312E-2	1.053529E-1	66	3.648633E-2	4.623870E-2	1.070442E-1
3	2.091065E-2	3.041591E-2	8.939411E-2	67	2.686521E-2	3.929376E-2	8.411799E-2
4	1.889090E-2	3.827222E-2	1.378204E-1	68	2.729040E-2	5.538051E-2	1.415862E-1
5	2.051438E-2	2.854812E-2	7.397622E-2	69	2.484767E-2	3.632777E-2	7.624309E-2
6	4.695103E-2	6.840319E-2	1.091238E-1	70	5.254308E-2	7.757787E-2	1.145680E-1
7	3.155572E-2	5.691400E-2	8.570576E-2	71	4.077414E-2	5.399238E-2	9.076405E-2
8	3.811819E-2	7.777847E-2	1.925329E-1	72	5.730433E-2	7.658031E-2	1.795790E-1
9	2.162972E-2	2.929089E-2	6.250420E-2	73	2.460324E-2	3.414084E-2	6.789908E-2
10	3.114140E-2	5.990793E-2	1.028607E-1	74	4.082201E-2	6.297838E-2	9.951913E-2
11	3.027993E-2	5.350124E-2	7.809258E-2	75	3.830250E-2	5.528575E-2	7.900193E-2
12	6.508462E-2	9.066247E-2	1.428510E-1	76	7.241113E-2	1.019039E-1	1.469796E-1
13	3.273404E-2	5.040278E-2	6.264923E-2	77	3.739022E-2	4.704639E-2	6.546845E-2
14	5.274399E-2	6.225743E-2	1.221983E-1	78	5.273975E-2	6.727704E-2	1.396804E-1
15	3.488404E-2	6.422224E-2	9.160246E-2	79	4.053654E-2	7.050813E-2	9.256686E-2
16	4.889844E-2	1.050580E-1	1.688135E-1	80	4.434253E-2	1.103672E-1	1.996363E-1

j	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$	$q_{rate}(1, 3, j)$	*j*	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$	$q_{rate}(1, 3, j)$
17	2.357911E-2	3.210347E-2	5.608996E-2	81	2.549207E-2	3.476040E-2	6.059020E-2
18	2.772528E-2	4.872818E-2	1.012242E-1	82	4.354655E-2	5.323695E-2	1.083260E-1
19	2.743480E-2	4.049659E-2	9.349261E-2	83	2.795998E-2	4.913248E-2	8.842845E-2
20	4.383601E-2	6.032613E-2	1.524009E-1	84	4.980519E-2	8.817289E-2	1.525973E-1
21	2.689949E-2	4.529064E-2	6.498004E-2	85	3.193463E-2	4.621693E-2	6.852064E-2
22	5.160590E-2	6.083122E-2	1.087996E-1	86	5.802463E-2	6.842687E-2	1.150853E-1
23	4.200649E-2	6.118451E-2	8.544740E-2	87	4.339047E-2	6.905756E-2	8.449844E-2
24	7.135027E-2	1.019721E-1	1.746410E-1	88	7.396916E-2	1.192405E-1	1.773402E-1
25	2.889067E-2	4.139644E-2	5.259280E-2	89	3.187675E-2	4.596974E-2	5.723726E-2
26	3.163645E-2	6.635321E-2	1.249503E-1	90	4.508738E-2	5.665094E-2	1.320058E-1
27	4.302895E-2	5.140233E-2	7.968777E-2	91	4.590970E-2	5.455804E-2	8.614233E-2
28	5.709708E-2	1.084445E-1	1.440756E-1	92	7.446858E-2	1.138154E-1	1.615706E-1
29	3.388403E-2	5.047469E-2	7.297654E-2	93	3.975096E-2	4.953595E-2	7.225423E-2
30	6.542657E-2	7.909877E-2	1.155706E-1	94	6.762578E-2	8.310290E-2	1.279901E-1
31	3.854235E-2	7.331254E-2	1.023075E-1	95	5.762581E-2	6.953264E-2	1.050130E-1
32	6.578245E-2	1.029098E-1	2.118744E-1	96	6.853135E-2	1.217588E-1	2.206266E-1
33	1.547279E-2	2.045597E-2	5.461213E-2	97	2.184805E-2	2.991309E-2	5.162080E-2
34	2.279502E-2	3.909542E-2	1.194438E-1	98	3.643432E-2	4.917951E-2	1.232772E-1
35	3.068892E-2	4.545402E-2	8.204189E-2	99	3.896113E-2	4.766350E-2	8.617166E-2
36	2.259572E-2	4.791017E-2	1.718444E-1	100	4.146352E-2	6.880070E-2	1.693562E-1
37	2.710880E-2	4.017396E-2	7.019229E-2	101	3.355146E-2	4.178152E-2	7.371594E-2
38	4.957894E-2	7.929633E-2	1.048625E-1	102	5.802247E-2	8.703142E-2	1.129175E-1
39	3.060959E-2	5.640594E-2	9.495841E-2	103	4.802431E-2	5.694865E-2	1.007557E-1
40	6.342246E-2	9.116555E-2	1.847244E-1	104	5.988731E-2	8.579423E-2	2.013889E-1
41	2.433424E-2	3.919983E-2	6.314062E-2	105	2.993100E-2	3.948284E-2	6.463761E-2
42	3.380120E-2	6.608465E-2	1.110315E-1	106	3.886266E-2	8.074436E-2	1.155198E-1
43	3.517841E-2	5.793973E-2	7.207029E-2	107	3.494440E-2	6.289110E-2	8.049820E-2
44	6.490541E-2	8.658319E-2	1.546487E-1	108	6.888179E-2	9.924311E-2	1.603933E-1
45	2.919347E-2	5.162046E-2	6.944373E-2	109	3.642377E-2	5.340165E-2	6.701520E-2
46	5.945228E-2	7.198293E-2	1.274345E-1	110	5.834927E-2	7.852858E-2	1.417467E-1
47	5.318885E-2	6.381821E-2	9.882185E-2	111	4.864696E-2	7.267369E-2	9.483159E-2
48	8.682910E-2	1.411354E-1	1.917285E-1	112	5.855336E-2	1.362898E-1	1.986397E-1
49	2.499911E-2	3.625560E-2	5.037240E-2	113	2.608885E-2	3.734068E-2	5.578532E-2
50	2.822464E-2	5.445723E-2	1.126635E-1	114	4.585044E-2	5.605125E-2	1.179279E-1
51	3.626181E-2	4.590732E-2	9.433436E-2	115	4.288013E-2	5.147391E-2	9.753090E-2
52	5.704553E-2	7.463004E-2	1.591572E-1	116	6.376116E-2	8.735529E-2	1.683349E-1
53	2.729875E-2	4.566259E-2	7.525297E-2	117	3.767099E-2	4.582160E-2	7.865281E-2
54	5.128602E-2	8.511270E-2	1.235880E-1	118	6.751946E-2	8.986979E-2	1.194181E-1
55	4.914520E-2	5.934831E-2	9.226860E-2	119	5.463743E-2	6.668059E-2	8.938138E-2
56	7.069619E-2	1.054520E-1	1.926021E-1	120	7.730866E-2	1.217544E-1	1.995792E-1
57	2.807338E-2	4.185092E-2	5.871598E-2	121	3.156213E-2	4.517022E-2	6.257685E-2
58	4.644490E-2	7.066988E-2	1.260384E-1	122	3.787827E-2	8.034865E-2	1.389617E-1
59	4.184537E-2	6.304453E-2	7.661699E-2	123	5.083033E-2	6.187406E-2	8.311538E-2
60	8.424164E-2	1.132829E-1	1.436871E-1	124	8.963114E-2	1.287538E-1	1.648916E-1
61	4.176156E-2	5.594729E-2	7.098728E-2	125	4.735035E-2	5.757244E-2	7.652646E-2

j	$q_{rate}(1,1,j)$	$q_{rate}(1,2,j)$	$q_{rate}(1,3,j)$	j	$q_{rate}(1,1,j)$	$q_{rate}(1,2,j)$	$q_{rate}(1,3,j)$
62	5.551614E-2	9.501267E-2	1.277272E-1	126	7.168986E-2	9.898957E-2	1.300784E-1
63	5.909355E-2	7.367300E-2	9.659359E-2	127	6.290826E-2	7.907788E-2	1.051111E-1
64	7.841367E-2	1.414324E-1	2.174286E-1	128	8.806498E-2	1.652062E-1	2.132142E-1

Table 8-6 LSP Quantization Table, Rate 1/2, Codebook 2

j	$q_{rate}(2,1,j)$	$q_{rate}(2,2,j)$	$q_{rate}(2,3,j)$	j	$q_{rate}(2,1,j)$	$q_{rate}(2,2,j)$	$q_{rate}(2,3,j)$
1	9.759153E-2	1.237015E-1	1.694380E-1	65	1.140764E-1	1.330714E-1	1.731815E-1
2	9.495363E-2	2.010818E-1	2.268553E-1	66	1.135758E-1	1.903073E-1	2.416812E-1
3	9.004966E-2	1.491649E-1	2.265328E-1	67	8.591653E-2	1.639202E-1	2.379345E-1
4	1.703027E-1	1.972229E-1	2.499748E-1	68	1.929169E-1	2.150824E-1	2.391281E-1
5	1.087736E-1	1.519724E-1	1.751234E-1	69	1.372918E-1	1.594233E-1	1.797222E-1
6	1.302789E-1	2.132292E-1	2.296464E-1	70	1.404354E-1	2.220923E-1	2.409608E-1
7	1.249180E-1	1.873478E-1	2.047120E-1	71	1.403872E-1	1.896012E-1	2.056357E-1
8	2.006702E-1	2.289636E-1	2.694208E-1	72	2.116955E-1	2.365784E-1	2.812489E-1
9	8.983756E-2	1.253328E-1	2.105394E-1	73	9.030106E-2	1.271574E-1	2.335679E-1
10	9.623767E-2	2.071859E-1	2.541745E-1	74	1.101181E-1	2.093284E-1	2.728363E-1
11	1.056946E-1	1.788564E-1	2.001210E-1	75	1.167104E-1	1.778540E-1	2.228088E-1
12	1.560490E-1	2.195737E-1	2.910794E-1	76	1.816915E-1	2.322652E-1	2.749912E-1
13	1.373923E-1	1.599933E-1	1.946985E-1	77	1.465535E-1	1.694747E-1	1.902460E-1
14	1.072625E-1	2.377910E-1	2.707408E-1	78	1.092138E-1	2.632920E-1	2.884908E-1
15	1.429765E-1	2.015505E-1	2.184689E-1	79	1.498151E-1	2.113427E-1	2.288995E-1
16	2.142705E-1	2.718814E-1	3.012002E-1	80	1.976455E-1	2.832300E-1	3.148823E-1
17	1.107292E-1	1.336882E-1	1.548772E-1	81	1.244956E-1	1.460980E-1	1.661252E-1
18	1.066677E-1	1.766788E-1	2.627989E-1	82	1.348786E-1	1.830301E-1	2.892883E-1
19	9.163529E-2	1.745928E-1	2.193293E-1	83	9.330321E-2	1.839622E-1	2.385430E-1
20	1.840386E-1	2.279641E-1	2.477622E-1	84	1.928443E-1	2.395883E-1	2.584215E-1
21	1.105724E-1	1.582072E-1	1.960131E-1	85	1.237968E-1	1.655566E-1	2.084084E-1
22	1.335434E-1	2.322697E-1	2.518282E-1	86	1.511443E-1	2.358011E-1	2.592806E-1
23	1.559223E-1	1.779413E-1	2.180966E-1	87	1.506577E-1	1.900525E-1	2.283626E-1
24	1.922601E-1	2.495125E-1	2.899115E-1	88	1.981810E-1	2.567942E-1	3.089756E-1
25	1.137089E-1	1.378724E-1	2.029299E-1	89	1.284900E-1	1.490840E-1	1.983765E-1
26	1.025575E-1	1.848201E-1	2.921646E-1	90	9.205958E-2	2.122313E-1	2.929488E-1
27	1.365956E-1	1.586874E-1	2.413996E-1	91	1.416981E-1	1.723567E-1	2.584541E-1
28	1.728138E-1	2.493034E-1	3.004586E-1	92	1.967335E-1	2.297097E-1	2.957802E-1
29	1.368712E-1	1.572498E-1	2.109132E-1	93	1.470622E-1	1.689181E-1	2.073636E-1
30	1.289748E-1	2.451679E-1	2.676536E-1	94	1.363099E-1	2.603731E-1	2.826074E-1
31	1.668123E-1	1.889980E-1	2.313459E-1	95	1.810411E-1	2.018261E-1	2.388676E-1
32	2.322485E-1	2.631961E-1	3.167549E-1	96	2.453263E-1	2.801831E-1	3.119543E-1
33	9.245610E-2	1.199775E-1	1.912623E-1	97	1.041318E-1	1.330407E-1	1.898347E-1
34	1.130853E-1	2.084615E-1	2.293681E-1	98	1.232982E-1	2.096211E-1	2.478132E-1
35	1.007164E-1	1.406701E-1	2.580630E-1	99	1.240408E-1	1.598274E-1	2.588561E-1
36	1.670104E-1	2.181055E-1	2.625925E-1	100	1.870489E-1	2.124881E-1	2.596291E-1
37	1.254872E-1	1.626870E-1	1.844092E-1	101	1.242553E-1	1.737690E-1	1.928500E-1
38	1.524066E-1	2.071317E-1	2.475824E-1	102	1.589178E-1	2.253898E-1	2.432848E-1
39	1.374412E-1	1.802624E-1	2.176988E-1	103	1.534212E-1	1.918073E-1	2.092495E-1

j	$q_{rate}(2,1,j)$	$q_{rate}(2,2,j)$	$q_{rate}(2,3,j)$	j	$q_{rate}(2,1,j)$	$q_{rate}(2,2,j)$	$q_{rate}(2,3,j)$
40	2.078535E-1	2.492095E-1	2.698301E-1	104	2.271545E-1	2.511812E-1	2.726004E-1
41	9.352573E-2	1.491974E-1	2.046520E-1	105	1.099221E-1	1.571003E-1	2.200250E-1
42	1.119972E-1	2.252331E-1	2.470031E-1	106	1.327824E-1	2.194855E-1	2.670289E-1
43	1.093150E-1	1.938119E-1	2.138022E-1	107	1.268575E-1	1.988363E-1	2.179285E-1
44	1.751186E-1	2.525203E-1	2.750828E-1	108	1.914150E-1	2.524242E-1	2.726528E-1
45	1.369187E-1	1.774406E-1	1.979311E-1	109	1.552776E-1	1.795735E-1	2.007736E-1
46	1.368112E-1	2.374262E-1	2.847378E-1	110	1.175477E-1	2.478699E-1	3.082793E-1
47	1.607598E-1	2.008332E-1	2.180845E-1	111	1.657070E-1	2.103395E-1	2.291993E-1
48	2.337102E-1	2.663726E-1	2.918021E-1	112	2.256949E-1	2.844382E-1	3.121061E-1
49	1.191711E-1	1.397032E-1	1.877233E-1	113	1.295032E-1	1.484201E-1	1.801804E-1
50	1.310500E-1	1.936967E-1	2.604270E-1	114	1.547525E-1	1.977485E-1	2.672750E-1
51	1.082671E-1	1.651948E-1	2.395230E-1	115	1.285902E-1	1.761784E-1	2.399059E-1
52	2.031950E-1	2.259422E-1	2.494032E-1	116	2.149268E-1	2.376344E-1	2.587940E-1
53	1.238429E-1	1.457946E-1	2.156356E-1	117	1.283223E-1	1.593385E-1	2.266266E-1
54	1.712263E-1	2.380545E-1	2.579756E-1	118	1.557476E-1	2.477405E-1	2.737268E-1
55	1.669238E-1	1.886047E-1	2.111242E-1	119	1.757417E-1	1.979524E-1	2.191159E-1
56	2.106208E-1	2.624427E-1	2.831280E-1	120	2.186264E-1	2.458092E-1	3.004797E-1
57	1.057488E-1	1.362865E-1	2.200502E-1	121	1.177090E-1	1.455129E-1	2.380445E-1
58	9.729458E-2	2.334715E-1	2.961140E-1	122	1.180069E-1	2.237755E-1	2.941751E-1
59	1.342984E-1	1.939554E-1	2.391488E-1	123	1.513492E-1	1.881578E-1	2.487433E-1
60	1.642293E-1	2.700678E-1	2.941425E-1	124	1.893122E-1	2.695805E-1	2.937860E-1
61	1.427603E-1	1.650334E-1	2.241004E-1	125	1.498956E-1	1.745373E-1	2.374300E-1
62	1.464145E-1	2.479423E-1	3.007081E-1	126	1.397755E-1	2.717094E-1	3.078395E-1
63	1.747788E-1	2.193493E-1	2.381630E-1	127	1.839457E-1	2.077172E-1	2.267222E-1
64	2.363111E-1	2.906697E-1	3.280110E-1	128	2.545522E-1	2.966409E-1	3.248015E-1

Table 8-7 LSP Quantization Table, Rate 1/2, Codebook 3

j	$q_{rate}(3,1,j)$	$q_{rate}(3,2,j)$	$q_{rate}(3,3,j)$	$q_{rate}(3,4,j)$
1	2.369047E-1	2.561044E-1	3.169558E-1	4.075205E-1
2	2.975969E-1	3.234825E-1	3.476675E-1	3.745512E-1
3	2.737212E-1	2.982975E-1	3.299239E-1	3.835991E-1
4	3.078496E-1	3.328363E-1	3.893403E-1	4.055760E-1
5	2.338036E-1	2.602965E-1	3.673520E-1	4.043883E-1
6	2.975137E-1	3.153566E-1	3.851352E-1	4.021971E-1
7	2.856188E-1	3.108728E-1	3.650224E-1	3.848168E-1
8	3.352716E-1	3.552222E-1	3.819211E-1	3.986858E-1
9	2.002656E-1	2.505023E-1	3.703982E-1	4.320127E-1
10	3.079821E-1	3.337677E-1	3.581991E-1	3.783868E-1
11	2.600861E-1	3.255203E-1	3.568733E-1	3.847378E-1
12	3.013564E-1	3.413694E-1	4.002968E-1	4.173372E-1
13	2.670810E-1	2.976744E-1	3.697020E-1	3.891392E-1
14	2.726699E-1	3.497041E-1	3.919253E-1	4.063833E-1
15	2.528259E-1	3.496366E-1	3.845510E-1	4.059310E-1
16	3.429271E-1	3.742740E-1	4.054682E-1	4.203519E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	$q_{rate}(3, 4, j)$
17	2.524087E-1	2.803758E-1	3.214366E-1	3.884369E-1
18	2.969702E-1	3.171736E-1	3.653426E-1	4.027368E-1
19	2.819052E-1	3.014792E-1	3.343356E-1	4.076335E-1
20	3.268729E-1	3.471777E-1	3.750177E-1	4.053724E-1
21	2.363711E-1	3.164411E-1	3.487070E-1	3.820304E-1
22	2.878176E-1	3.136270E-1	4.051297E-1	4.233797E-1
23	2.775025E-1	3.018438E-1	3.722509E-1	4.192128E-1
24	3.289889E-1	3.619011E-1	4.020155E-1	4.192298E-1
25	2.249605E-1	2.746364E-1	3.770161E-1	3.947265E-1
26	3.010455E-1	3.404862E-1	3.748881E-1	4.025322E-1
27	2.598980E-1	3.303350E-1	3.574938E-1	4.086580E-1
28	3.009619E-1	3.564491E-1	4.047795E-1	4.225090E-1
29	2.209796E-1	3.164777E-1	4.017441E-1	4.207358E-1
30	2.797550E-1	3.307761E-1	4.111529E-1	4.326870E-1
31	2.642469E-1	3.166106E-1	3.838767E-1	4.366838E-1
32	3.443812E-1	3.853657E-1	4.249495E-1	4.415602E-1
33	2.194883E-1	2.364599E-1	3.424660E-1	4.249900E-1
34	2.914651E-1	3.222820E-1	3.728528E-1	3.916359E-1
35	2.747924E-1	3.165363E-1	3.453926E-1	3.745552E-1
36	3.105835E-1	3.352649E-1	3.875272E-1	4.230762E-1
37	2.232115E-1	2.984976E-1	3.684262E-1	3.902137E-1
38	2.890788E-1	3.265128E-1	3.763087E-1	4.095537E-1
39	2.638301E-1	3.089773E-1	3.814530E-1	4.046608E-1
40	3.470736E-1	3.647978E-1	3.867635E-1	4.045117E-1
41	2.184527E-1	2.756141E-1	3.627111E-1	4.182790E-1
42	3.150428E-1	3.408132E-1	3.786272E-1	3.963168E-1
43	2.797277E-1	3.312597E-1	3.600613E-1	3.811755E-1
44	3.186024E-1	3.380443E-1	4.090108E-1	4.303004E-1
45	2.641969E-1	2.906725E-1	3.685950E-1	4.318568E-1
46	2.726456E-1	3.635148E-1	3.965188E-1	4.200912E-1
47	2.265410E-1	3.500551E-1	3.938515E-1	4.125970E-1
48	3.530539E-1	3.699296E-1	4.096561E-1	4.263873E-1
49	2.607884E-1	2.851725E-1	3.459433E-1	3.975007E-1
50	3.011131E-1	3.282019E-1	3.560680E-1	4.108038E-1
51	2.881016E-1	3.095596E-1	3.437568E-1	4.248729E-1
52	3.104894E-1	3.514219E-1	3.937174E-1	4.155505E-1
53	2.223083E-1	3.267982E-1	3.779817E-1	3.986350E-1
54	3.029155E-1	3.227819E-1	3.985589E-1	4.254896E-1
55	2.771368E-1	3.199926E-1	3.774909E-1	4.291775E-1
56	3.387318E-1	3.581644E-1	4.083864E-1	4.254954E-1
57	2.187262E-1	2.843850E-1	3.940537E-1	4.163470E-1
58	3.010060E-1	3.440937E-1	3.690137E-1	4.150913E-1
59	2.807837E-1	3.330537E-1	3.767262E-1	3.975269E-1
60	3.143941E-1	3.626788E-1	4.236690E-1	4.418992E-1
61	2.664536E-1	3.085138E-1	3.974072E-1	4.174502E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	$q_{rate}(3, 4, j)$
62	2.942227E-1	3.419044E-1	4.127269E-1	4.348889E-1
63	2.873007E-1	3.324346E-1	3.788567E-1	4.382340E-1
64	3.571466E-1	3.981471E-1	4.298757E-1	4.442439E-1
65	2.296713E-1	2.510186E-1	3.410466E-1	4.043763E-1
66	2.944726E-1	3.349446E-1	3.604097E-1	3.836829E-1
67	2.882510E-1	3.117227E-1	3.316801E-1	3.651047E-1
68	3.248816E-1	3.456567E-1	3.883064E-1	4.059549E-1
69	2.508292E-1	2.776235E-1	3.707995E-1	3.904792E-1
70	2.935234E-1	3.283192E-1	3.921123E-1	4.094641E-1
71	2.836088E-1	3.038856E-1	3.785044E-1	3.973106E-1
72	3.340398E-1	3.528374E-1	3.972729E-1	4.143220E-1
73	2.218919E-1	2.518775E-1	3.717235E-1	4.317910E-1
74	3.132014E-1	3.411754E-1	3.655036E-1	3.885672E-1
75	2.713305E-1	3.391637E-1	3.626164E-1	3.957360E-1
76	3.075501E-1	3.477777E-1	4.010496E-1	4.327675E-1
77	2.593874E-1	2.872438E-1	3.868173E-1	4.060427E-1
78	2.854852E-1	3.440950E-1	4.020505E-1	4.194138E-1
79	2.657814E-1	3.400844E-1	3.694077E-1	4.270317E-1
80	3.537409E-1	3.844633E-1	4.117478E-1	4.261818E-1
81	2.438665E-1	2.683502E-1	3.422020E-1	3.984572E-1
82	2.931452E-1	3.347542E-1	3.617028E-1	3.984166E-1
83	2.913430E-1	3.131552E-1	3.365259E-1	3.877486E-1
84	3.056568E-1	3.629046E-1	3.881534E-1	4.055432E-1
85	2.174923E-1	3.117235E-1	3.759848E-1	4.289978E-1
86	2.911493E-1	3.293809E-1	4.039004E-1	4.223332E-1
87	2.903621E-1	3.095310E-1	3.789942E-1	4.136884E-1
88	3.295649E-1	3.774047E-1	4.065849E-1	4.247397E-1
89	2.464616E-1	2.715933E-1	3.663383E-1	4.307538E-1
90	3.141077E-1	3.370119E-1	3.804097E-1	4.110994E-1
91	2.765684E-1	3.273207E-1	3.588443E-1	4.289495E-1
92	3.171791E-1	3.589724E-1	4.047658E-1	4.403763E-1
93	2.427778E-1	3.349548E-1	3.969435E-1	4.133184E-1
94	2.888955E-1	3.256912E-1	4.228596E-1	4.437587E-1
95	2.775833E-1	3.254790E-1	3.891447E-1	4.410759E-1
96	3.591257E-1	3.906941E-1	4.210095E-1	4.357085E-1
97	2.201724E-1	2.477193E-1	3.543819E-1	4.253981E-1
98	3.060468E-1	3.279247E-1	3.669928E-1	3.931926E-1
99	2.708056E-1	3.168266E-1	3.456487E-1	4.117176E-1
100	3.231889E-1	3.454631E-1	3.897788E-1	4.215708E-1
101	2.461361E-1	3.123920E-1	3.721886E-1	3.958427E-1
102	3.038567E-1	3.243548E-1	3.857473E-1	4.141550E-1
103	2.810754E-1	3.186085E-1	3.856469E-1	4.027036E-1
104	3.535171E-1	3.727025E-1	3.962646E-1	4.130749E-1
105	2.092211E-1	2.952622E-1	3.803143E-1	4.312782E-1
106	3.253136E-1	3.467355E-1	3.707240E-1	3.910456E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	$q_{rate}(3, 4, j)$
107	2.863965E-1	3.435600E-1	3.697136E-1	3.898678E-1
108	3.277947E-1	3.473678E-1	4.054651E-1	4.245662E-1
109	2.530550E-1	3.026563E-1	3.821651E-1	4.298983E-1
110	2.944185E-1	3.707454E-1	3.954433E-1	4.195148E-1
111	2.628731E-1	3.450692E-1	4.041409E-1	4.219021E-1
112	3.650635E-1	3.824351E-1	4.134248E-1	4.312417E-1
113	2.487885E-1	2.823728E-1	3.657723E-1	4.109811E-1
114	3.072888E-1	3.278289E-1	3.776650E-1	4.362209E-1
115	2.985423E-1	3.206273E-1	3.505697E-1	4.276202E-1
116	3.162580E-1	3.629038E-1	3.882251E-1	4.256089E-1
117	2.390779E-1	3.313105E-1	3.703179E-1	4.159959E-1
118	3.037358E-1	3.328061E-1	4.102328E-1	4.277511E-1
119	2.960025E-1	3.190148E-1	3.810625E-1	4.269550E-1
120	3.325089E-1	3.625170E-1	4.233151E-1	4.409952E-1
121	2.351287E-1	2.747311E-1	4.120706E-1	4.354788E-1
122	2.980738E-1	3.553388E-1	3.790878E-1	4.153188E-1
123	2.834298E-1	3.452649E-1	3.703763E-1	4.099008E-1
124	3.235931E-1	3.654128E-1	4.128131E-1	4.310235E-1
125	2.766264E-1	3.005084E-1	4.022369E-1	4.266388E-1
126	2.945129E-1	3.614432E-1	4.196352E-1	4.369992E-1
127	2.908073E-1	3.416894E-1	3.927793E-1	4.434903E-1
128	3.593915E-1	4.039851E-1	4.408438E-1	4.530286E-1
129	2.232955E-1	2.391925E-1	3.237680E-1	4.216895E-1
130	2.947781E-1	3.187987E-1	3.532178E-1	3.919064E-1
131	2.590321E-1	3.102405E-1	3.435690E-1	3.950642E-1
132	3.164747E-1	3.385444E-1	3.933290E-1	4.122356E-1
133	2.401082E-1	2.846312E-1	3.602810E-1	3.799738E-1
134	2.969091E-1	3.157983E-1	3.949643E-1	4.151276E-1
135	2.854341E-1	3.049215E-1	3.619747E-1	4.057673E-1
136	3.374071E-1	3.566722E-1	3.851551E-1	4.111867E-1
137	2.240149E-1	2.601162E-1	3.947725E-1	4.195859E-1
138	3.006479E-1	3.416407E-1	3.702235E-1	3.895201E-1
139	2.659460E-1	3.250392E-1	3.743399E-1	3.923461E-1
140	3.160293E-1	3.404913E-1	4.023553E-1	4.204842E-1
141	2.698415E-1	2.945624E-1	3.623418E-1	4.064155E-1
142	2.788973E-1	3.598310E-1	3.820258E-1	4.105775E-1
143	2.607608E-1	3.310885E-1	3.888263E-1	4.054866E-1
144	3.433723E-1	3.826470E-1	4.147166E-1	4.315929E-1
145	2.479981E-1	2.733932E-1	3.311604E-1	4.189432E-1
146	3.035796E-1	3.252025E-1	3.709844E-1	4.144205E-1
147	2.768969E-1	3.004995E-1	3.541782E-1	4.288070E-1
148	3.236556E-1	3.598170E-1	3.895254E-1	4.092887E-1
149	2.389278E-1	3.099192E-1	3.539156E-1	4.166343E-1
150	2.811717E-1	3.075203E-1	4.162649E-1	4.385238E-1
151	2.888587E-1	3.098108E-1	3.678452E-1	4.360356E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	$q_{rate}(3, 4, j)$
152	3.384235E-1	3.706344E-1	4.154500E-1	4.315345E-1
153	2.412604E-1	2.736179E-1	3.895546E-1	4.125395E-1
154	2.980467E-1	3.401221E-1	3.861837E-1	4.138264E-1
155	2.824364E-1	3.315975E-1	3.579414E-1	4.121152E-1
156	3.038202E-1	3.705886E-1	4.057750E-1	4.315171E-1
157	2.390777E-1	3.116385E-1	4.139358E-1	4.353041E-1
158	2.671168E-1	3.419379E-1	4.174094E-1	4.391848E-1
159	2.679468E-1	3.333439E-1	3.864814E-1	4.374625E-1
160	3.405110E-1	3.908780E-1	4.354851E-1	4.491019E-1
161	2.100699E-1	2.325245E-1	3.617814E-1	4.313579E-1
162	2.945099E-1	3.337098E-1	3.822786E-1	3.986389E-1
163	2.805252E-1	3.259052E-1	3.506470E-1	3.928739E-1
164	3.199996E-1	3.436747E-1	3.910705E-1	4.375011E-1
165	2.205810E-1	3.031519E-1	3.817655E-1	4.044882E-1
166	2.861227E-1	3.297465E-1	3.881028E-1	4.242477E-1
167	2.698071E-1	3.253323E-1	3.791545E-1	4.151382E-1
168	3.348589E-1	3.692584E-1	3.947431E-1	4.119222E-1
169	2.071098E-1	2.727795E-1	3.785664E-1	4.345800E-1
170	3.064662E-1	3.466957E-1	3.871383E-1	4.035583E-1
171	2.701486E-1	3.466545E-1	3.776967E-1	3.964345E-1
172	3.187459E-1	3.402257E-1	4.149916E-1	4.415788E-1
173	2.585928E-1	3.143701E-1	3.650838E-1	4.216152E-1
174	2.827130E-1	3.541371E-1	4.067460E-1	4.292679E-1
175	2.520218E-1	3.591051E-1	3.951029E-1	4.181484E-1
176	3.549062E-1	3.749529E-1	4.189660E-1	4.361444E-1
177	2.648411E-1	2.929418E-1	3.277515E-1	4.087905E-1
178	3.077743E-1	3.355862E-1	3.622096E-1	4.253942E-1
179	2.884663E-1	3.160757E-1	3.609896E-1	4.195514E-1
180	3.171284E-1	3.557722E-1	4.058088E-1	4.239730E-1
181	2.470897E-1	3.381846E-1	3.718596E-1	3.959715E-1
182	3.079817E-1	3.326918E-1	4.005342E-1	4.382737E-1
183	2.794848E-1	3.161835E-1	3.972377E-1	4.347466E-1
184	3.444905E-1	3.661532E-1	4.109594E-1	4.417271E-1
185	2.357418E-1	2.945873E-1	3.980725E-1	4.168334E-1
186	3.140385E-1	3.522720E-1	3.791389E-1	4.109691E-1
187	2.830025E-1	3.381363E-1	3.886419E-1	4.061933E-1
188	3.236253E-1	3.502434E-1	4.280896E-1	4.466304E-1
189	2.612521E-1	3.249710E-1	4.002145E-1	4.253218E-1
190	3.052845E-1	3.421642E-1	4.244751E-1	4.438310E-1
191	2.873748E-1	3.325006E-1	3.943083E-1	4.425385E-1
192	3.740754E-1	4.020264E-1	4.309335E-1	4.441600E-1
193	2.345040E-1	2.562186E-1	3.412388E-1	4.230453E-1
194	3.054926E-1	3.291570E-1	3.527098E-1	3.924391E-1
195	2.813236E-1	3.032923E-1	3.489254E-1	3.931639E-1
196	3.218935E-1	3.504199E-1	3.973175E-1	4.145603E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	$q_{rate}(3, 4, j)$
197	2.396846E-1	2.924515E-1	3.789374E-1	3.965355E-1
198	3.073072E-1	3.291279E-1	3.984556E-1	4.161433E-1
199	2.852746E-1	3.087745E-1	3.929165E-1	4.144377E-1
200	3.444464E-1	3.622019E-1	3.976198E-1	4.177436E-1
201	2.320831E-1	2.678080E-1	3.780757E-1	4.345609E-1
202	3.047387E-1	3.518653E-1	3.759732E-1	3.952937E-1
203	2.619909E-1	3.462073E-1	3.712969E-1	4.124389E-1
204	3.110809E-1	3.510409E-1	4.160828E-1	4.343401E-1
205	2.749804E-1	2.966315E-1	3.875205E-1	4.092438E-1
206	2.909391E-1	3.544556E-1	3.934270E-1	4.082203E-1
207	2.718719E-1	3.455108E-1	3.871253E-1	4.225906E-1
208	3.632459E-1	3.819322E-1	4.041149E-1	4.183707E-1
209	2.457707E-1	2.729093E-1	3.483179E-1	4.251618E-1
210	3.141390E-1	3.378723E-1	3.651952E-1	4.044234E-1
211	2.940758E-1	3.169355E-1	3.430472E-1	4.061304E-1
212	3.146275E-1	3.724134E-1	4.006607E-1	4.179308E-1
213	2.340142E-1	3.140072E-1	3.830035E-1	4.348292E-1
214	2.936357E-1	3.205300E-1	4.108374E-1	4.363931E-1
215	2.895058E-1	3.118289E-1	3.863115E-1	4.387713E-1
216	3.263174E-1	3.808582E-1	4.197214E-1	4.387955E-1
217	2.508095E-1	2.830181E-1	3.822474E-1	4.342444E-1
218	3.189941E-1	3.448551E-1	3.726901E-1	4.230670E-1
219	2.883801E-1	3.366222E-1	3.697423E-1	4.250576E-1
220	3.061077E-1	3.818569E-1	4.182062E-1	4.328684E-1
221	2.338983E-1	3.448618E-1	4.121766E-1	4.292162E-1
222	2.859809E-1	3.429038E-1	4.251129E-1	4.442997E-1
223	2.798588E-1	3.387893E-1	3.920854E-1	4.405410E-1
224	3.645093E-1	3.822027E-1	4.298306E-1	4.458184E-1
225	2.343923E-1	2.573774E-1	3.595671E-1	4.300886E-1
226	3.050319E-1	3.275894E-1	3.783056E-1	4.010261E-1
227	2.775226E-1	3.181303E-1	3.677943E-1	4.015430E-1
228	3.330358E-1	3.558210E-1	3.875489E-1	4.246287E-1
229	2.450210E-1	3.125607E-1	3.911476E-1	4.087628E-1
230	2.970591E-1	3.402469E-1	3.929193E-1	4.288997E-1
231	2.778393E-1	3.250198E-1	3.974364E-1	4.159209E-1
232	3.494653E-1	3.703625E-1	3.954825E-1	4.319234E-1
233	2.314856E-1	2.910234E-1	3.779095E-1	4.322597E-1
234	3.192835E-1	3.536711E-1	3.809829E-1	3.978434E-1
235	2.896892E-1	3.502657E-1	3.807297E-1	3.979694E-1
236	3.289873E-1	3.520054E-1	4.125572E-1	4.375979E-1
237	2.762733E-1	3.022672E-1	3.817234E-1	4.349891E-1
238	2.796273E-1	3.737273E-1	4.123746E-1	4.306263E-1
239	2.534428E-1	3.659400E-1	4.149370E-1	4.327436E-1
240	3.761072E-1	3.951420E-1	4.167877E-1	4.330236E-1
241	2.628158E-1	2.882705E-1	3.473972E-1	4.241826E-1

j	$q_{rate}(3, 1, j)$	$q_{rate}(3, 2, j)$	$q_{rate}(3, 3, j)$	$q_{rate}(3, 4, j)$
242	3.019313E-1	3.436526E-1	3.770313E-1	4.342045E-1
243	2.978343E-1	3.234954E-1	3.644924E-1	4.335508E-1
244	3.317745E-1	3.643249E-1	3.982436E-1	4.350783E-1
245	2.490497E-1	3.278708E-1	3.835870E-1	4.355581E-1
246	3.046534E-1	3.276712E-1	4.184847E-1	4.413788E-1
247	2.969609E-1	3.238989E-1	3.904637E-1	4.399160E-1
248	3.439238E-1	3.671005E-1	4.295232E-1	4.452150E-1
249	2.593997E-1	2.916027E-1	4.043725E-1	4.314132E-1
250	2.975375E-1	3.575738E-1	3.889918E-1	4.300070E-1
251	2.840689E-1	3.495746E-1	3.810428E-1	4.297128E-1
252	3.257163E-1	3.748759E-1	4.319593E-1	4.472908E-1
253	2.653030E-1	3.147460E-1	4.167035E-1	4.372947E-1
254	3.003986E-1	3.541473E-1	4.285381E-1	4.603364E-1
255	2.980772E-1	3.493049E-1	4.004293E-1	4.482135E-1
256	3.755762E-1	4.166573E-1	4.421368E-1	4.527286E-1

Table 8-8 LSP Quantization Table, Rate 1/8, Codebook 1

j	$q_{rate}(1, 1, j)$	$q_{rate}(1, 2, j)$	$q_{rate}(1, 3, j)$	$q_{rate}(1, 4, j)$	$q_{rate}(1, 5, j)$
1	4.209106E-2	6.947497E-2	1.116895E-1	1.457197E-1	2.089358E-1
2	5.494466E-2	9.824226E-2	1.100788E-1	1.589078E-1	2.054824E-1
3	4.518857E-2	7.519943E-2	1.142339E-1	1.546973E-1	1.974671E-1
4	4.947500E-2	7.966750E-2	1.257135E-1	1.694478E-1	2.077532E-1
5	4.178938E-2	6.345956E-2	1.206803E-1	1.585077E-1	2.040682E-1
6	4.715924E-2	7.912955E-2	1.218311E-1	1.565005E-1	2.230923E-1
7	5.453992E-2	8.034305E-2	1.294776E-1	1.518615E-1	2.017172E-1
8	5.585208E-2	9.411485E-2	1.401603E-1	1.780708E-1	2.295549E-1
9	4.544353E-2	7.354141E-2	1.193766E-1	1.544203E-1	2.101075E-1
10	6.317801E-2	9.523149E-2	1.236498E-1	1.767254E-1	2.174373E-1
11	5.276537E-2	8.435144E-2	1.158909E-1	1.579092E-1	2.073235E-1
12	5.186575E-2	8.132854E-2	1.375623E-1	1.832288E-1	2.164007E-1
13	4.441953E-2	6.887446E-2	1.311525E-1	1.626358E-1	2.165910E-1
14	4.937844E-2	8.188255E-2	1.306717E-1	1.682190E-1	2.313608E-1
15	5.590978E-2	9.078330E-2	1.334885E-1	1.629847E-1	2.096152E-1
16	6.137821E-2	9.860277E-2	1.479333E-1	1.928319E-1	2.315651E-1

Table 8-9 LSP Quantization Table, Rate 1/8, Codebook 2

j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$	$q_{rate}(2, 3, j)$	$q_{rate}(2, 4, j)$	$q_{rate}(2, 5, j)$
1	2.682296E-1	3.058530E-1	3.111035E-1	3.682334E-1	4.077447E-1
2	2.441801E-1	2.897017E-1	3.257376E-1	3.902148E-1	4.134584E-1
3	2.334183E-1	3.007829E-1	3.289390E-1	3.855733E-1	4.106846E-1
4	2.590587E-1	2.975686E-1	3.419662E-1	3.853117E-1	4.129523E-1

j	$q_{rate}(2, 1, j)$	$q_{rate}(2, 2, j)$	$q_{rate}(2, 3, j)$	$q_{rate}(2, 4, j)$	$q_{rate}(2, 5, j)$
5	2.429045E-1	2.922362E-1	3.271855E-1	3.778814E-1	4.033293E-1
6	2.467419E-1	2.974937E-1	3.363123E-1	3.942606E-1	4.225895E-1
7	2.137760E-1	3.314042E-1	3.406769E-1	3.822208E-1	4.093902E-1
8	2.667348E-1	3.079165E-1	3.441972E-1	3.961151E-1	4.238752E-1
9	2.612143E-1	3.049254E-1	3.299724E-1	3.848680E-1	4.202374E-1
10	2.495487E-1	2.937286E-1	3.338274E-1	3.785067E-1	4.171406E-1
11	2.415889E-1	3.017342E-1	3.412825E-1	3.842858E-1	4.161965E-1
12	2.581891E-1	3.173641E-1	3.490434E-1	3.876993E-1	4.155136E-1
13	2.445059E-1	3.067345E-1	3.357932E-1	3.784443E-1	4.055705E-1
14	2.516403E-1	3.122508E-1	3.384779E-1	3.955419E-1	4.239680E-1
15	2.278799E-1	3.177920E-1	3.383191E-1	4.004411E-1	4.118556E-1
16	2.789686E-1	3.226197E-1	3.565811E-1	4.020676E-1	4.237045E-1

Table 8-10 Interpolation Filter Coefficients, Cutoff=0.5 (1/2)

n	$I_\varepsilon(7n)$	$I_\varepsilon(1 + 7n)$	$I_\varepsilon(2 + 7n)$	$I_\varepsilon(3 + 7n)$
0	-2.279553E-2	9.638780E-2	4.296515E-1	4.296515E-1
1	-2.165019E-2	6.437141E-2	3.936069E-1	4.594094E-1
2	-1.928710E-2	3.758768E-2	3.527558E-1	4.816282E-1
3	-1.634340E-2	1.617878E-2	3.087218E-1	4.953565E-1
4	-1.332172E-2	0.000000E+0	2.631803E-1	5.000000E-1
5	-1.057099E-2	-1.134067E-2	2.177682E-1	4.953565E-1
6	-8.282481E-3	-1.842952E-2	1.739981E-1	4.816282E-1
7	-6.500361E-3	-2.198557E-2	1.331849E-1	4.594094E-1

Table 8-11 Interpolation Filter Coefficients, Cutoff=0.5 (2/2)

n	$I_\varepsilon(4 + 7n)$	$I_\varepsilon(5 + 7n)$	$I_\varepsilon(6 + 7n)$
0	9.638780E-2	-2.279553E-2	-5.144665E-3
1	1.331849E-1	-2.198557E-2	-6.500361E-3
2	1.739981E-1	-1.842952E-2	-8.282481E-3
3	2.177682E-1	-1.134067E-2	-1.057099E-2
4	2.631803E-1	0.000000E+0	-1.332172E-2
5	3.087218E-1	1.617878E-2	-1.634340E-2
6	3.527558E-1	3.758768E-2	-1.928710E-2
7	3.936069E-1	6.437141E-2	-2.165019E-2

Table 8-12 Interpolation Filter Coefficients, Cutoff=0.9 (1/3)

n	$I_E(17n)$	$I_E(1 + 17n)$	$I_E(2 + 17n)$	$I_E(3 + 17n)$	$I_E(4 + 17n)$
0	3.333057E-3	-4.447694E-3	3.032594E-3	5.505681E-3	-2.749339E-2
1	1.814939E-3	-1.051010E-3	-3.490370E-3	1.602235E-2	-4.161043E-2
2	3.143855E-4	1.903512E-3	-8.614938E-3	2.320955E-2	-4.897606E-2

n	$I_E(17n)$	$I_E(1 + 17n)$	$I_E(2 + 17n)$	$I_E(3 + 17n)$	$I_E(4 + 17n)$
3	-1.011791E-3	4.128520E-3	-1.191523E-2	2.662804E-2	-4.954024E-2
4	-2.054155E-3	5.477784E-3	-1.325900E-2	2.636338E-2	-4.408088E-2
5	-2.749624E-3	5.939437E-3	-1.278129E-2	2.294780E-2	-3.401131E-2
6	-3.074976E-3	5.612335E-3	-1.082686E-2	1.723493E-2	-2.113197E-2
7	-3.038022E-3	4.671966E-3	-7.873206E-3	1.024916E-2	-7.363276E-3

Table 8-13 Interpolation Filter Coefficients, Cutoff=0.9 (2/3)

n	$I_E(5 + 17n)$	$I_E(6 + 17n)$	$I_E(7 + 17n)$	$I_E(8 + 17n)$	$I_E(9 + 17n)$	$I_E(10 + 17n)$
0	7.357492E-2	-1.760500E-1	6.238571E-1	6.238571E-1	-1.760500E-1	7.357492E-2
1	8.809371E-2	-1.788657E-1	4.934066E-1	7.373307E-1	-1.480059E-1	4.678921E-2
2	9.025505E-2	-1.604327E-1	3.555172E-1	8.252803E-1	-9.275004E-2	9.433596E-3
3	8.139432E-2	-1.261773E-1	2.198452E-1	8.809472E-1	-1.067283E-2	-3.518170E-2
4	6.395375E-2	-8.231784E-2	9.530775E-2	9.000000E-1	9.530775E-2	-8.231784E-2
5	4.107321E-2	-3.518170E-2	-1.067283E-2	8.809472E-1	2.198452E-1	-1.261773E-1
6	1.614589E-2	9.433596E-3	-9.275004E-2	8.252803E-1	3.555172E-1	-1.604327E-1
7	-7.606618E-3	4.678921E-2	-1.480059E-1	7.373307E-1	4.934066E-1	-1.788657E-1

Table 8-14 Interpolation Filter Coefficients, Cutoff=0.9 (3/3)

n	$I_E(11 + 17n)$	$I_E(12 + 17n)$	$I_E(13 + 17n)$	$I_E(14 + 17n)$	$I_E(15 + 17n)$	$I_E(16 + 17n)$
0	-2.749339E-2	5.505681E-3	3.032594E-3	-4.447694E-3	3.333057E-3	-2.669328E-3
1	-7.606618E-3	-7.363276E-3	1.024916E-2	-7.873206E-3	4.671966E-3	-3.038022E-3
2	1.614589E-2	-2.113197E-2	1.723493E-2	-1.082686E-2	5.612335E-3	-3.074976E-3
3	4.107321E-2	-3.401131E-2	2.294780E-2	-1.278129E-2	5.939437E-3	-2.749624E-3
4	6.395375E-2	-4.408088E-2	2.636338E-2	-1.325900E-2	5.477784E-3	-2.054155E-3
5	8.139432E-2	-4.954024E-2	2.662804E-2	-1.191523E-2	4.128520E-3	-1.011791E-3
6	9.025505E-2	-4.897606E-2	2.320955E-2	-8.614938E-3	1.903512E-3	3.143855E-4
7	8.809371E-2	-4.161043E-2	1.602235E-2	-3.490370E-3	-1.051010E-3	1.814939E-3

Table 8-15 Fixed Codebook Gain Quantization, Rate 1

k	$g_{ccb}(k)$	k	$g_{ccb}(k)$
0	1.2840254E+0	16	7.0105412E+1
1	1.6487213E+0	17	9.0017131E+1
2	2.1170000E+0	18	1.1558429E+2
3	2.7182818E+0	19	1.4841316E+2
4	3.4903430E+0	20	1.9056627E+2
5	4.4816891E+0	21	2.4469193E+2
6	5.7546027E+0	22	3.1419066E+2
7	7.3890561E+0	23	4.0342879E+2
8	9.4877358E+0	24	5.1801283E+2

k	$g_{ccb}(k)$	k	$g_{ccb}(k)$
9	1.2182494E+1	25	6.6514163E+2
10	1.5642632E+1	26	8.5405876E+2
11	2.0085537E+1	27	1.0966332E+3
12	2.5790340E+1	28	1.4081049E+3
13	3.3115452E+1	29	1.8080424E+3
14	4.2521082E+1	30	2.3215724E+3
15	5.4598150E+1	31	2.9809580E+3

1

2

Table 8-16 Fixed Codebook Gain Quantization, Rate 1/2

k	$G_{ccb}(k)$	k	$G_{ccb}(k)$
0	1.6487213E+0	8	9.0017131E+1
1	2.7182818E+0	9	1.4841316E+2
2	4.4816891E+0	10	2.4469193E+2
3	7.3890561E+0	11	4.0342879E+2
4	1.2182494E+1	12	6.6514163E+2
5	2.0085537E+1	13	1.0966332E+3
6	3.3115452E+1	14	1.8080424E+3
7	5.4598150E+1	15	2.9809580E+3

3

4

Table 8-17 Residual Shift Interpolation Filter Coefficients

j	$I_f(-1, j)$	$I_f(0, j)$	$I_f(1, j)$
0	3.750000E-01	7.500000E-01	-1.250000E-01
1	2.578125E-01	8.593750E-01	-1.171875E-01
2	1.562500E-01	9.375000E-01	-9.375000E-02
3	7.031250E-02	9.843750E-01	-5.468750E-02
4	0.000000E+00	1.000000E+00	0.000000E+00
5	-5.468750E-02	9.843750E-01	7.031250E-02
6	-9.375000E-02	9.375000E-01	1.562500E-01
7	-1.171875E-01	8.593750E-01	2.578125E-01

5

Table 8-18 Rate 1/8 Frame Energy Quantization

k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$	k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$
0	-2.464E-02	-4.005E-03	-1.107E-01	128	-4.208E-02	-1.491E-01	-7.639E-02
1	8.734E-01	1.004E+00	9.930E-01	129	1.046E+00	9.598E-01	9.176E-01
2	4.222E-01	3.894E-01	5.020E-01	130	4.478E-01	4.605E-01	5.111E-01
3	1.450E+00	1.328E+00	1.278E+00	131	1.521E+00	1.292E+00	1.342E+00
4	1.957E-01	2.169E-01	2.735E-01	132	2.220E-01	2.549E-01	2.510E-01
5	1.142E+00	1.240E+00	1.157E+00	133	1.186E+00	1.254E+00	1.171E+00
6	7.881E-01	6.778E-01	4.185E-01	134	8.999E-01	4.960E-01	4.943E-01
7	1.504E+00	1.468E+00	1.534E+00	135	1.423E+00	1.484E+00	1.620E+00
8	3.173E-01	2.693E-01	-9.526E-02	136	2.796E-01	2.778E-01	-2.820E-01

k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$	k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$
9	1.141E+00	1.154E+00	1.044E+00	137	1.170E+00	1.181E+00	1.076E+00
10	5.147E-01	5.784E-01	8.802E-01	138	4.068E-01	8.541E-01	9.352E-01
11	1.502E+00	1.407E+00	1.409E+00	139	1.584E+00	1.416E+00	1.387E+00
12	3.163E-01	3.592E-01	2.830E-01	140	3.325E-01	3.655E-01	3.340E-01
13	1.217E+00	1.213E+00	1.216E+00	141	1.224E+00	1.257E+00	1.245E+00
14	1.023E+00	1.139E+00	-9.526E-02	142	1.061E+00	1.138E+00	-9.526E-02
15	1.619E+00	1.655E+00	1.642E+00	143	1.681E+00	1.704E+00	1.673E+00
16	1.437E-01	1.505E-01	6.838E-02	144	1.932E-01	1.489E-01	1.258E-01
17	9.794E-01	1.021E+00	1.117E+00	145	1.023E+00	1.088E+00	1.145E+00
18	4.701E-01	6.426E-01	5.519E-01	146	5.190E-01	6.873E-01	5.172E-01
19	1.366E+00	1.397E+00	1.406E+00	147	1.380E+00	1.405E+00	1.474E+00
20	2.918E-01	3.022E-01	2.420E-01	148	3.393E-01	3.100E-01	2.231E-01
21	1.309E+00	1.241E+00	1.220E+00	149	1.354E+00	1.249E+00	1.270E+00
22	7.989E-01	7.654E-01	7.391E-01	150	7.363E-01	8.508E-01	8.247E-01
23	1.612E+00	1.502E+00	1.447E+00	151	1.612E+00	1.537E+00	1.509E+00
24	2.594E-01	1.948E-01	2.555E-01	152	2.952E-01	2.053E-01	2.590E-01
25	1.091E+00	1.150E+00	1.272E+00	153	1.138E+00	1.219E+00	1.262E+00
26	3.423E-01	4.150E-01	1.294E+00	154	1.345E+00	1.289E+00	1.338E+00
27	1.729E+00	1.377E+00	1.065E+00	155	1.437E+00	1.360E+00	1.442E+00
28	4.103E-01	3.287E-01	3.228E-01	156	4.826E-01	3.298E-01	3.842E-01
29	1.144E+00	1.281E+00	1.416E+00	157	1.219E+00	1.311E+00	1.413E+00
30	1.047E+00	1.117E+00	6.188E-01	158	1.212E+00	1.186E+00	6.357E-01
31	1.914E+00	1.777E+00	1.516E+00	159	1.873E+00	1.939E+00	1.674E+00
32	-2.117E-02	2.159E-01	2.351E-01	160	1.260E+00	1.306E+00	1.368E+00
33	1.093E+00	1.088E+00	1.026E+00	161	1.146E+00	1.077E+00	1.025E+00
34	5.567E-01	5.092E-01	4.654E-01	162	6.029E-01	5.039E-01	5.781E-01
35	1.510E+00	1.449E+00	1.201E+00	163	1.514E+00	1.420E+00	1.324E+00
36	2.362E-01	3.426E-01	2.549E-01	164	2.652E-01	3.192E-01	3.042E-01
37	1.340E+00	1.225E+00	1.117E+00	165	1.368E+00	1.198E+00	1.200E+00
38	1.203E+00	3.819E-01	2.269E-01	166	1.234E+00	4.910E-01	3.464E-02
39	1.373E+00	1.404E+00	1.830E+00	167	1.347E+00	1.560E+00	1.861E+00
40	2.570E-01	2.668E-01	1.636E-01	168	2.766E-01	2.887E-01	2.029E-01
41	1.219E+00	1.098E+00	1.122E+00	169	1.257E+00	1.105E+00	1.145E+00
42	6.985E-01	8.456E-01	1.069E+00	170	1.351E+00	1.353E+00	1.406E+00
43	1.550E+00	1.501E+00	1.388E+00	171	1.506E+00	1.580E+00	1.362E+00
44	2.870E-01	3.060E-01	3.599E-01	172	2.794E-01	3.868E-01	4.277E-01
45	1.178E+00	1.345E+00	1.302E+00	173	1.234E+00	1.334E+00	1.336E+00
46	1.270E+00	1.215E+00	1.812E-01	174	1.280E+00	1.252E+00	1.805E-01
47	1.725E+00	1.777E+00	1.693E+00	175	1.387E+00	1.396E+00	1.434E+00
48	2.074E-01	2.104E-01	1.539E-01	176	2.902E-01	1.170E-01	1.698E-01
49	1.105E+00	1.034E+00	1.104E+00	177	1.134E+00	1.077E+00	1.117E+00
50	6.683E-01	6.646E-01	6.639E-01	178	6.986E-01	7.177E-01	7.366E-01
51	1.403E+00	1.462E+00	1.435E+00	179	1.370E+00	1.491E+00	1.495E+00
52	3.389E-01	3.754E-01	2.150E-01	180	4.031E-01	5.144E-01	1.751E-01
53	1.288E+00	1.325E+00	1.257E+00	181	1.333E+00	1.377E+00	1.257E+00

k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$	k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$
54	8.933E-01	8.253E-01	8.133E-01	182	9.212E-01	8.934E-01	8.897E-01
55	1.555E+00	1.579E+00	1.565E+00	183	1.589E+00	1.614E+00	1.523E+00
56	3.264E-01	2.434E-01	2.852E-01	184	3.152E-01	2.164E-01	3.230E-01
57	1.242E+00	1.180E+00	1.202E+00	185	1.300E+00	1.145E+00	1.212E+00
58	1.314E-01	1.698E-01	1.646E+00	186	1.269E+00	1.245E+00	1.497E+00
59	1.797E+00	1.597E+00	1.241E+00	187	1.763E+00	1.716E+00	1.311E+00
60	4.721E-01	5.346E-01	3.066E-01	188	4.702E-01	5.422E-01	4.306E-01
61	1.274E+00	1.401E+00	1.351E+00	189	1.342E+00	1.433E+00	1.423E+00
62	1.455E+00	1.386E+00	6.430E-01	190	1.472E+00	1.404E+00	8.371E-01
63	1.828E+00	1.867E+00	1.825E+00	191	1.936E+00	1.883E+00	1.838E+00
64	-3.265E-01	-2.956E-01	-2.462E-01	192	1.266E+00	1.295E+00	1.302E+00
65	1.035E+00	1.020E+00	1.003E+00	193	1.074E+00	1.002E+00	1.023E+00
66	3.702E-01	4.307E-01	7.072E-01	194	5.206E-01	4.045E-01	6.549E-01
67	1.424E+00	1.345E+00	1.352E+00	195	1.457E+00	1.378E+00	1.363E+00
68	2.267E-01	2.680E-01	3.037E-01	196	2.715E-01	2.629E-01	2.841E-01
69	1.235E+00	1.249E+00	1.146E+00	197	1.264E+00	1.271E+00	1.175E+00
70	9.944E-01	6.485E-01	5.248E-01	198	1.337E+00	1.305E+00	1.306E+00
71	1.539E+00	1.492E+00	1.612E+00	199	1.555E+00	1.571E+00	1.657E+00
72	3.815E-01	3.360E-01	-9.526E-02	200	3.341E-01	4.147E-01	-3.648E-01
73	1.163E+00	1.144E+00	1.117E+00	201	1.188E+00	1.185E+00	1.161E+00
74	6.734E-01	7.656E-01	1.014E+00	202	6.198E-01	7.208E-01	1.157E+00
75	1.568E+00	1.438E+00	1.455E+00	203	1.582E+00	1.465E+00	1.513E+00
76	3.409E-01	3.317E-01	3.856E-01	204	3.839E-01	3.651E-01	3.814E-01
77	1.180E+00	1.284E+00	1.284E+00	205	1.214E+00	1.256E+00	1.292E+00
78	1.244E+00	1.214E+00	-9.526E-02	206	1.361E+00	1.363E+00	1.312E+00
79	1.753E+00	1.598E+00	1.744E+00	207	1.793E+00	1.693E+00	1.669E+00
80	1.548E-01	1.388E-01	2.020E-01	208	1.889E-01	1.275E-01	2.534E-01
81	1.027E+00	1.133E+00	1.093E+00	209	1.066E+00	1.174E+00	1.133E+00
82	3.906E-01	7.505E-01	5.705E-01	210	4.999E-01	8.207E-01	5.813E-01
83	1.420E+00	1.357E+00	1.543E+00	211	1.478E+00	1.416E+00	1.497E+00
84	3.252E-01	3.136E-01	2.804E-01	212	3.814E-01	3.138E-01	2.889E-01
85	1.351E+00	1.309E+00	1.224E+00	213	1.396E+00	1.265E+00	1.233E+00
86	8.781E-01	8.095E-01	7.109E-01	214	9.458E-01	9.161E-01	5.875E-01
87	1.614E+00	1.580E+00	1.433E+00	215	1.672E+00	1.632E+00	1.553E+00
88	3.222E-01	2.298E-01	2.157E-01	216	3.505E-01	2.525E-01	2.364E-01
89	1.216E+00	1.077E+00	1.247E+00	217	1.211E+00	1.138E+00	1.235E+00
90	1.363E+00	1.280E+00	1.317E+00	218	1.391E+00	1.231E+00	1.355E+00
91	1.751E+00	1.457E+00	1.182E+00	219	1.783E+00	1.510E+00	1.199E+00
92	4.428E-01	4.082E-01	3.181E-01	220	4.227E-01	4.548E-01	3.671E-01
93	1.157E+00	1.227E+00	1.604E+00	221	1.281E+00	1.254E+00	1.661E+00
94	1.286E+00	1.268E+00	8.167E-01	222	1.338E+00	1.379E+00	9.531E-01
95	1.994E+00	2.018E+00	1.307E+00	223	2.148E+00	1.965E+00	1.584E+00
96	2.671E-02	2.594E-01	3.397E-01	224	9.324E-02	3.575E-01	3.522E-01
97	1.164E+00	1.080E+00	9.321E-01	225	1.212E+00	1.086E+00	1.044E+00
98	5.998E-01	6.076E-01	5.081E-01	226	6.128E-01	6.136E-01	6.060E-01

k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$	k	$q_{log}(0, k)$	$q_{log}(1, k)$	$q_{log}(2, k)$
99	1.442E+00	1.442E+00	1.375E+00	227	1.484E+00	1.507E+00	1.396E+00
100	2.390E-01	3.554E-01	3.426E-01	228	2.820E-01	3.848E-01	3.156E-01
101	1.287E+00	1.307E+00	1.144E+00	229	1.368E+00	1.287E+00	1.128E+00
102	1.200E+00	7.495E-01	3.967E-01	230	1.369E+00	1.352E+00	1.358E+00
103	1.561E+00	1.517E+00	1.898E+00	231	1.381E+00	1.765E+00	2.113E+00
104	3.598E-01	3.463E-01	1.200E-01	232	1.314E+00	1.345E+00	1.334E+00
105	1.298E+00	1.125E+00	1.062E+00	233	1.290E+00	1.172E+00	1.119E+00
106	7.577E-01	1.013E+00	1.194E+00	234	1.304E+00	1.377E+00	1.427E+00
107	1.537E+00	1.513E+00	1.464E+00	235	1.490E+00	1.540E+00	1.536E+00
108	4.041E-01	4.038E-01	3.897E-01	236	3.994E-01	4.402E-01	4.173E-01
109	1.293E+00	1.219E+00	1.378E+00	237	1.323E+00	1.307E+00	1.392E+00
110	1.250E+00	1.391E+00	2.451E-01	238	1.400E+00	1.388E+00	1.369E+00
111	1.558E+00	1.764E+00	1.728E+00	239	1.669E+00	1.818E+00	1.834E+00
112	2.700E-01	1.894E-01	1.924E-01	240	2.742E-01	2.235E-01	1.986E-01
113	1.111E+00	1.112E+00	1.173E+00	241	1.137E+00	1.139E+00	1.201E+00
114	7.579E-01	8.342E-01	4.781E-01	242	1.324E+00	1.385E+00	1.349E+00
115	1.464E+00	1.477E+00	1.469E+00	243	1.455E+00	1.574E+00	1.454E+00
116	4.001E-01	3.104E-01	2.217E-01	244	5.019E-01	3.255E-01	2.555E-01
117	1.346E+00	1.421E+00	1.312E+00	245	1.388E+00	1.438E+00	1.300E+00
118	1.071E+00	8.967E-01	7.511E-01	246	1.394E+00	1.349E+00	1.411E+00
119	1.616E+00	1.551E+00	1.574E+00	247	1.639E+00	1.580E+00	1.681E+00
120	3.329E-01	2.785E-01	3.140E-01	248	3.920E-01	2.498E-01	3.523E-01
121	1.281E+00	1.209E+00	1.239E+00	249	1.301E+00	1.221E+00	1.285E+00
122	2.805E-01	2.687E-01	1.646E+00	250	1.318E+00	1.342E+00	1.494E+00
123	1.814E+00	1.514E+00	1.510E+00	251	1.910E+00	1.680E+00	1.470E+00
124	6.231E-01	4.200E-01	3.701E-01	252	6.082E-01	5.270E-01	4.173E-01
125	1.255E+00	1.429E+00	1.454E+00	253	1.255E+00	1.477E+00	1.503E+00
126	1.642E+00	1.581E+00	7.112E-01	254	1.807E+00	1.742E+00	6.553E-01
127	1.844E+00	1.963E+00	1.895E+00	255	2.000E+00	2.072E+00	2.051E+00

1

2