ETSI/TC SMG
Released by : ETSI/PT 12
Release date: February 1992

RELEASE NOTE

Recommendation GSM 06.10

GSM Full Rate Speech Transcoding

Previously distributed version : 3.2.0 (Release 1/90)
New Released version February 92 : 3.2.0 (Release 92, Phase 1)

1. Reason for changes

No changes since the previously distributed version.

GSM recommendation: 06.10

Title: GSM full rate speech transcoding

Date: February 1992

List of contents:

NOTE: This Recommendation is a reproduction of recommendation T/L/03/11 "13 kbit/s Regular Pulse Excitation - Long Term Prediction - Linear Predictive Coder for use in the Pan-European Digital Mobile Radio System".

Floppy disks containing the digital test sequences described in section 5 can be distributed by ETSI Secretariat on request.

The contact information of the ETSI secretariat is:

ETSI

B.P. 152

F 06561 Valbonne Cedex

France

Tel +33 92 94 42 00

Fax +33 93 65 47 16

Language of original: English

Number of pages: 93

Detailed list of contents

****

1. GENERAL

1.1. SCOPE

The transcoding procedure specified in this recommendation is applicable for the full-rate traffic channel (TCH) in the Pan-European Digital Mobile Radio (DMR) system. The use of this transcoding scheme for other applications has not been considered.

In recommendation GSM 06.01, a reference configuration for the speech transmission chain of the Pan-European DMR system is shown. According to this reference configuration, the speech encoder takes its input as a 13 bit uniform PCM signal either from the audio part of the mobile station or on the network side, from the PSTN via an 8 bit/A-law to 13 bit uniform PCM conversion. The encoded speech at the output of the speech encoder is delivered to a channel encoder unit which is specified in Rec.GSM 05.03. In the receive direction, the inverse operations take place.

This recommendation describes the detailed mapping between input blocks of 160 speech samples in 13 bit uniform PCM format to encoded blocks of 260 bits and from encoded blocks of 260 bits to output blocks of 160 reconstructed speech samples. The sampling rate is 8000 sample/s leading to an average bit rate for the encoded bit stream of 13 kbit/s. The coding scheme is the so-called Regular Pulse Excitation - Long Term prediction - Linear Predictive Coder, here-after referred to as RPE-LTP.

The recommendation also specifies the conversion between A-law PCM and 13 bit uniform PCM. Performance requirements for the audio input and output parts are included only to the extent that they affect the transcoder performance. The recommendation also describes the codec down to the bit level, thus enabling the verification of compliance to the recommendation to a high degree of confidence by use of a set of digital test sequences. These test sequences are also described and are available on floppy disks.

1.2. OUTLINE DESCRIPTION

The recommendation is structured as follows:

Section 1.3 contains a functional description of the audio parts including the A/D and D/A functions. Section 1.4 describes the conversion between 13 bit uniform and 8 bit A-law samples. Sections 1.5 and 1.6 present a simplified description of the principles of the RPE-LTP encoding and decoding process respectively. In section 1.7, the sequence and subjective importance of encoded parameters are given.

Section 2 deals with the transmission characteristics of the audio parts that are relevant for the performance of the RPE-LTP codec. Some transmission characteristics of the RPE-LTP codec are also specified in section 2. Section 3 presents the functional description of the RPE-LTP coding and decoding procedures, whereas section 4 describes the computational details of the algorithm. Procedures for the verification of the correct functioning of the RPE-LTP are described in section 5.

Performance and network aspects of the RPE-LTP codec are contained in annex 1.

1.3. FUNCTIONAL DESCRIPTION OF AUDIO PARTS

The analogue-to-digital and digital-to-analogue conversion will in principle comprise the following elements:

1) Analogue to uniform digital

- microphone,
- input level adjustment device,
- input anti-aliasing filter,
- sample-hold device sampling at 8 kHz,
- analogue-to-uniform digital conversion to 13 bits representation.

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

2) Uniform digital to analogue

- conversion from 13 bit /8kHz uniform PCM to analogue,
- a hold device,
- reconstruction filter including x/sin x correction,
- output level adjustment device,
- earphone or loudspeaker.

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

- by direct conversion to 13 bit uniform PCM format.
- or by conversion to 8 bit/A-law companded format, based on a standard A-law codec/filter according to CCITT rec. G.711/714, followed by the 8-bit to 13-bit conversion according to the procedure specified in section 1.4.

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

In the latter case it should be noted that the specifications in CCITT recommendation G.714 are concerned with PCM equipment located in the central parts of the network. When used in the terminal equipment, this specification does not on its own ensure sufficient out-of-band attenuation.

The specification of out-of-band signals is defined in section 2 between the acoustic signal and the digital interface to take into account that the filtering in the terminal can be achieved both by electronic and acoustical design.

1.4. PCM FORMAT CONVERSION

The conversion between 8 bit A-law companded format and the 13-bit uniform format shall be as defined in CCITT Recommendation G.721, section 4.2.1, sub-block EXPAND and section 4.2.7, sub-block COMPRESS. The parameter LAW = 1 should be used.

1.5. PRINCIPLES OF THE RPE-LTP ENCODER

A simplified block diagram of the RPE-LTP encoder is shown in Fig 1.1. In this diagram the coding and quantization functions are not shown explicitly.

The input speech frame, consisting of 160 signal samples (uniform 13 bit PCM samples), is first pre-processed to produce an offset-free signal, which is then subjected to a first order pre-emphasis filter. The 160 samples obtained are then analyzed to determine the coefficients for the short term analysis filter (LPC analysis). These parameters are then used for the filtering of the same 160 samples. The result is 160 samples of the short term residual signal. The filter parameters, termed reflection coefficients, are transformed to log.area ratios, LARs, before transmission.

For the following operations, the speech frame is divided into 4 sub-frames with 40 samples of the short term residual signal in each. Each sub-frame is processed blockwise by the subsequent functional elements.

Before the processing of each sub-block of 40 short term residual samples, the parame-ters of the long term analysis filter, the LTP lag and the LTP gain, are estimated and updated in the LTP analysis block, on the basis of the current sub-block of the present and a stored sequence of the 120 previous reconstructed short term residual samples.

A block of 40 long term residual signal samples is obtained by subtracting 40 estimates of the short term residual signal from the short term residual signal itself. The resulting block of 40 long term residual samples is fed to the Regular Pulse Excitation analysis which performs the basic compression function of the algorithm.

As a result of the RPE-analysis, the block of 40 input long term residual samples are represented by one of 4 candidate sub-sequences of 13 pulses each. The subsequence selected is identified by the RPE grid position (M). The 13 RPE pulses are encoded using Adaptive Pulse Code Modulation (APCM) with estimation of the sub-block amplitude which is transmitted to the decoder as side information.

The RPE parameters are also fed to a local RPE decoding and reconstruction module which produces a block of 40 samples of the quantized version of the long term residual signal.

By adding these 40 quantized samples of the long term residual to the previous block of short term residual signal estimates, a reconstructed version of the current short term residual signal is obtained.

The block of reconstructed short term residual signal samples is then fed to the long term analysis filter which produces the new block of 40 short term residual signal estimates to be used for the next sub-block thereby completing the feedback loop.

1.6. PRINCIPLES OF THE RPE-LTP DECODER

The simplified block diagram of the RPE-LTP decoder is shown in fig 1.2. The decoder includes the same structure as the feed-back loop of the encoder. In error-free transmission, the output of this stage will be the reconstructed short term residual samples. These samples are then applied to the short term synthesis filter followed by the de-emphasis filter resulting in the reconstructed speech signal samples.

1.7. SEQUENCE AND SUBJECTIVE IMPORTANCE OF ENCODED PARAMETERS

As indicated in fig 1.1 the three different groups of data are produced by the encoder are:

- the short term filter parameters,
- the Long Term Prediction (LTP) parameters
- the RPE parameters

The encoder will produce this information in a unique sequence and format, and the decoder must receive the same information in the same way. In table 1.1, the sequence of output bits b1 to b260 and the bit allocation for each parameter is shown.

The different parameters of the encoded speech and their individual bits have unequal importance with respect to subjective quality. Before being submitted to the channel encoder, the bits have to be rearranged in the sequence importance as given in table 1.2. The ranking has been determined by subjective testing and the procedure used is described in annex 2.

Parameter	Parameter number	Parameter name	Var. name	Number of bits	Bit no. (LSB-MSB)
	1		LAR 1	6	b1 - b6
	2		LAR 2	6	b7 - b12
FILTER	3	Log. Area	LAR 3	5	b13 - b17
	4	ratios	LAR 4	5	b18 - b22
PARAMETERS	5	1 - 8	LAR 5	4	b23 - b26
	6		LAR 6	4	b27 - b30
	7		LAR 7	3	b31 - b33
	8		LAR 8	3	b34 - b36
Sub-frame no.1					
LTP	9	LTP lag	N1	7	b37 - b43
PARAMETERS	10	LTP gain	b1	2	b44 - b45
	11	RPE grid position	M1	2	b46 - b47
RPE	12	Block amplitude	Xmax1	6	b48 - b53
PARAMETERS	13	RPE-pulse no.1	x1(0)	3	b54 - b56
	14	RPE-pulse no.2	x1(1)	3	b57 - b59
	..	...			...
	25	RPE-pulse no.13	x1(12)	3	b90 - b92
Sub-frame no.2					
LTP	26	LTP lag	N2	7	b93 - b99
PARAMETERS	27	LTP gain	b2	2	b100- b101
	28	RPE grid position	M2	2	b102- b103
RPE	29	Block amplitude	Xmax2	6	b104- b109
PARAMETERS	30	RPE-pulse no.1	x2(0)	3	b110- b112
	31	RPE-pulse no.2	x2(1)	3	b113- b115
	..	...			...
	42	RPE-pulse no.13	x2(12)	3	b146- b148

Table 1.1a. Encoder output parameters in order of occurrence and bit allocation within the speech frame of 260 bits/20 ms

--

Sub-frame no.3

LTP	43	LTP lag	N3	7	b149- b155
PARAMETERS	44	LTP gain	b3	2	b156- b157
	45	RPE grid position	M3	2	b158- b159
RPE	46	Block amplitude	Xmax3	6	b160- b165
PARAMETERS	47	RPE-pulse no.1	x3(0)	3	b166- b168
	48	RPE-pulse no.2	x3(1)	3	b169- b171
	..	...			...
	59	RPE-pulse no.13	x3(12)	3	b202- b204

Sub-frame no.4

LTP	60	LTP lag	N4	7	b205- b211
PARAMETERS	61	LTP gain	b4	2	b212- b213
	62	RPE grid position	M4	2	b214- b215
RPE	63	Block amplitude	Xmax4	6	b216- b221
PARAMETERS	64	RPE-pulse no.1	x4(0)	3	b222- b224
	65	RPE-pulse no.2	x4(1)	3	b225- b227
	..	...			...
	76	RPE-pulse no.13	x4(12)	3	b258- b260

Table 1.1b. Encoder output parameters in order of occurrence and bit allocation within the speech frame of 260 bits/20 ms

--

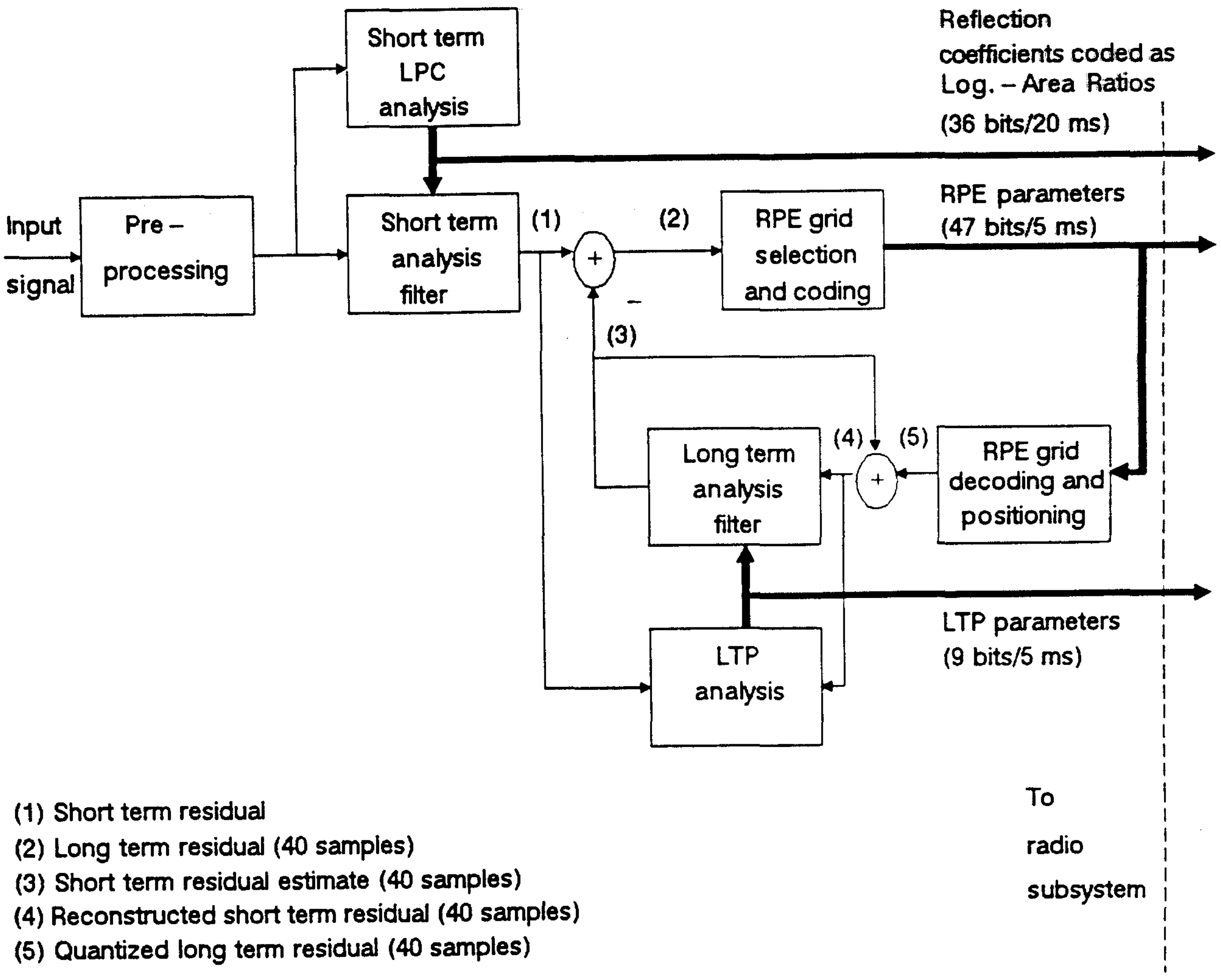
Order of bit importance	Parameter name	Parameter number	Bit number
Class Ia			
1	Log.area ratio 1	1	b6
2,3,4,5	Block amplitude	12,29,46,63	b53,b109,b165,b221
6	Log.area ratio 1	1	b5
7	Log.area ratio 2	2	b12
8	Log.area ratio 3	3	b17
.	Log.area ratio 1	1	b4
.	Log.area ratio 2	2	b11
	Log.area ratio 3	3	b16
	Log.area ratio 4	4	b22
	LTP lag	9,26,43,60	b43,b99,b155,b211
	Block amplitude	12,29,46,63	b52,b108,b164,b220
	Log.area ratio 2,5,6	2,5,6	b10,b26,b30
	LTP lag	9,26,43,60	b42,b98,b154,b210
	LTP lag	9,26,43,60	b41,b97,b153,b209
	LTP lag	9,26,43,60	b40,b96,b152,b208
	LTP lag	9,26,43,60	b39,b95,b151,b207
	Block amplitude	12,29,46,63	b51,b107,b163,b219
	Log.area ratio 1	1	b3
.	Log.area ratio 4	4	b21
.	Log.area ratio 7	7	b33
..49,50	LTP lag	9,26,43,60	b38,b94,b150,b206
Class Ib			
51,52	Log.area ratio 5,6	5,6	b25,b29
.	LTP gain	10,27,44,61	b45,b101,b157,b213
.	LTP lag	9,26,43,60	b37,b93,b149,b205
	Grid position	11,28,45,62	b47,b103,b159,b215
	Log.area ratio 1	1	b2
	Log.area ratio 2,3,8,4	2,3,8,4	b9,b15,b36,b20
	Log.area ratio 5,7	5,7	b24,b32
	LTP gain	10,27,44,61	b44,b100,b156,b212
	Block amplitude	12,29,46,63	b50,b106,b162,b218
	RPE pulses	13..25	b56,b59,..,b92
	RPE pulses	30..42	b112,b115,..,b148
	RPE pulses	47..59	b168,b171,..,b204
	RPE pulses	64..76	b224,b227,..,b260
	Grid position	11,28,45,62	b46,b102,b158,b214
	Block amplitude	12,29,46,63	b49,b105,b161,b217
	RPE pulses	13..25	b55,b58,..,b91
.	RPE pulses	30..42	b111,b114,..,b147
.	RPE pulses	47..59	b167,b170,..,b203
...182	RPE pulses	64..67	b223,b226,b229,b232

Table 1.2a. Subjective importance of encoded bits (the parameter and bit numbers refer to table 1.1)

Class II			
183...	RPE pulses	68..76	b235,b238,..,b259
	Log.area ratio 1	1	b1
	Log.area ratio 2,3,6	2,3,6	b8,b14,b28
	Log.area ratio 7	7	b31
	Log.area ratio 8	8	b35
	Log.area ratio 8,3	8,3	b34,b13
	Log.area ratio 4	4	b19
	Log.area ratio 4,5	4,5	b18,b23
	Block amplitude	12,29,46,63	b48,b104,b160,b216
	RPE pulses	13..25	b54,b57,..,b90
	RPE pulses	30..42	b110,b113,..,b146
	RPE pulses	47..59	b166,b169,..,b202
	RPE pulses	64..76	b222,b225,..,b258
259,260	Log.area ratio 2,6	2,6	b7,b27

Table 1.2b. Subjective importance of encoded bits (the parameter and bit numbers refer to table 1.1)

NOTE: The subdivisions in table 1.2 indicate the border between protection classes Ia, Ib and II as defined in recommendation GSM 05.03.


Short term LPC analysis
Reflection coefficients coded as Log. – Area Ratios (36 bits/20 ms)
Input signal
Pre – processing
Short term analysis filter
(1)
+
(2)
–
RPE grid selection and coding
RPE parameters (47 bits/5 ms)
(3)
Long term analysis filter
(4)
+
(5)
RPE grid decoding and positioning
LTP analysis
LTP parameters (9 bits/5 ms)
To radio subsystem

(1) Short term residual
(2) Long term residual (40 samples)
(3) Short term residual estimate (40 samples)
(4) Reconstructed short term residual (40 samples)
(5) Quantized long term residual (40 samples)

<u>Fig 1.1. Simplified block diagram of the RPE – LTP encoder</u>

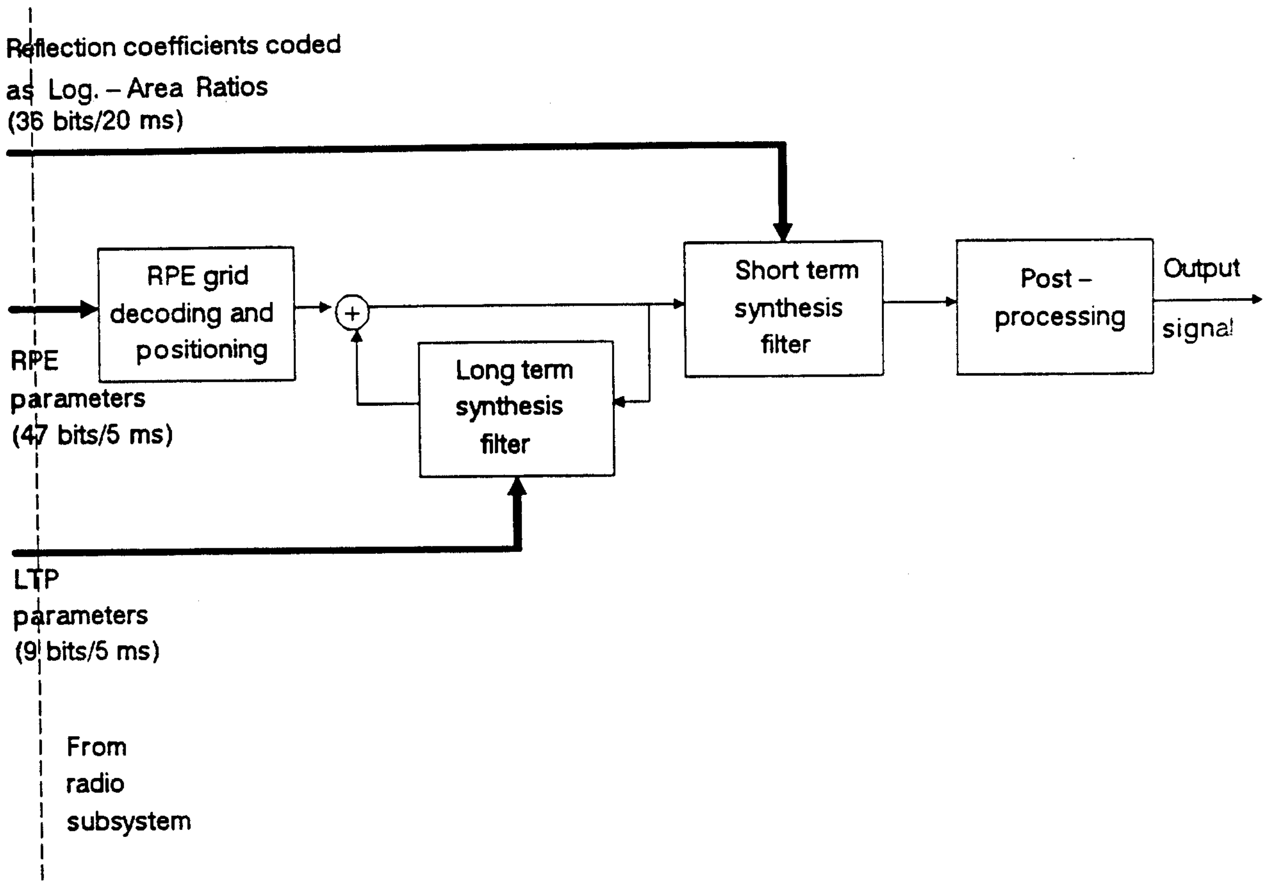


Fig 1.2. Simplified block diagram of the RPE – LTP decoder

2. TRANSMISSION CHARACTERISTICS

This section specifies the necessary performance characteristics of the audio parts for proper functioning of the speech trancoder. Some transmission performance characteristics of the RPE-LTP transcoder are also given to assist the designer of the speech transcoder function. The information given here is redundant and the detailed specifications are contained in recommendation GSM 11.10.

The performance characteristics are referred to the 13 bit uniform PCM interface.

NOTE: To simplify the verification of the specifications, the performance limits may be referred to an A-law measurement interface according to CCITT Rec-ommendation G.711. In this way, standard measuring equipments for PCM systems can be utilized for measurements. The relationship between the 13 bit format and the A-law companded shall follow the procedures defined in section 1.4.

2.1. PERFORMANCE CHARACTERISTICS OF THE ANALOGUE/DIGITAL INTERFACES

Concerning 1) discrimination against out-of-band signals (sending) and 2) spurious out-of-band signals (receiving), the same requirements as defined in ETSI standard TE 04-15 (digital telephone, candidate NET33) apply.

2.2. TRANSCODER DELAY

Consider a back to back configuration where the parameters generated by the encoder are delivered to the speech decoder as soon as they are available.

The transcoder delay is defined as the time interval between the instant a speech frame of 160 samples has been received at the encoder input and the instant the corresponding 160 reconstructed speech samples have been out-put by the speech decoder at an 8 kHz sample rate.

The theoretical minimum delay which can be achieved is 20 ms. The requirement is that the transcoder delay should be less than 30 ms.

3. FUNCTIONAL DESCRIPTION OF THE RPE-LTP CODEC

The block diagram of the RPE-LTP-coder is shown in fig 3.1. The individual blocks are described in the following sections.

3.1. FUNCTIONAL DESCRIPTION OF THE RPE-LTP ENCODER

The Preprocessing section of the RPE-LTP encoder comprises the following two sub-blocks:

* Offset compensation (3.1.1)
* Preemphasis (3.1.2)

The LPC analysis section of the RPE-LTP encoder comprises the following five sub-blocks:

* Segmentation (3.1.3)
* Auto-Correlation (3.1.4)
* Schur Recursion (3.1.5)
* Transformation of reflection coefficients to Log.-Area Ratios (3.1.6)
* Quantization and coding of Log.-Area Ratios (3.1.7)

The Short term analysis filtering section of the RPE-LTP comprises the following four sub-blocks:

* Decoding of the quantized Log.-Area Ratios (LARs) (3.1.8)
* Interpolation of Log.-Area Ratios (3.1.9)
* Transformation of Log.-Area Ratios into reflection coefficients (3.1.10)
* Short term analysis filtering (3.1.11)

The Long Term Predictor (LTP) section comprises 4 sub-blocks working on subsegments (3.1.12) of the short term residual samples.

* Calculation of LTP parameters (3.1.13)
* Coding of the LTP lags (3.1.14) and the LTP gains (3.1.15)
* Decoding of the LTP lags (3.1.14) and the LTP gains (3.1.15)
* Long term analysis filtering (3.1.16), and
  Long term synthesis filtering (3.1.17)

The RPE encoding section comprises five different sub-blocks:

* Weighting filter (3.1.18)
* Adaptive sample rate decimation by RPE grid selection (3.1.19)
* APCM quantization of the selected RPE sequence (3.1.20)
* APCM inverse quantization (3.1.21)
* RPE grid positioning (3.1.22)

PREPROCESSING SECTION

3.1.1. Offset compensation

Prior to the speech encoder an offset compensation,by a notch filter is applied in order to remove the offset of the input signal s_o to produce the offset-free signal s_{of}.

$$s_{of}(k) = s_o(k) - s_o(k-1) + alpha*s_{of}(k-1) \qquad (3.1.1)$$

$$alpha = 32735*2^{-15}$$

3.1.2. Preemphasis

The signal s_{of} is applied to a first order FIR preemphasis filter leading to the input signal s of the analysis section.

$$s(k) = s_{of}(k) - beta*s_{of}(k-1) \qquad (3.1.2)$$

$$beta= 28180*2^{-15}$$

LPC ANALYSIS SECTION

3.1.3. Segmentation

The speech signal s(k) is divided into non-overlapping frames having a length of T_0 = 20 ms (160 samples). A new LPC-analysis of order p=8 is performed for each frame.

3.1.4. Autocorrelation

The first p+1 = 9 values of the Auto-Correlation function are calculated by

$$ACF(k) = \sum_{i=k}^{159} s(i)s(i-k) \qquad ,k = 0,1...,8 \qquad (3.2)$$

3.1.5. Schur Recursion

The reflection coefficients are calculated as shown in Fig 3.2 using the Schur Recursion algorithm. The term "reflection coefficient" comes from the theory of linear prediction of speech (LPC), where a vocal tract representation consisting of series of uniform cylindrical sections is assumed. Such a representation can be described by the reflection coefficents or the area ratios of connected sections.

3.1.6. Transformation of reflection coefficients to Log.-Area Ratios

The reflection coefficients r(i), (i=1..8), calculated by the Schur algorithm, are in the range

$$-1 <= r(i) <= +1$$

Due to the favourable quantization characteristics, the reflection coefficients are converted into Log.-Area Ratios which are strictly defined as follows:

$$Logarea(i) = \log_{10} \left(\frac{1 + r(i)}{1 - r(i)}\right) \qquad (3.3)$$

Since it is the companding characteristic of this transformation that is of importance, the following segmented approximation is used.

$$LAR(i) = \begin{cases} r(i) & ; \quad |r(i)| < 0.675 \\ sign[r(i)]*[2|r(i)|-0.675] & ; \quad 0.675 <= |r(i)| < 0.950 \\ sign[r(i)]*[8|r(i)|-6.375] & ; \quad 0.950 <= |r(i)| <= 1.000 \end{cases} \qquad (3.4)$$

with the result that instead of having to divide and obtain the logarithm of particular values, it is merely necessary to multiply, add and compare these values.

The following equation (3.5) gives the inverse transformation.

$$r'(i) = \begin{cases} LAR'(i) & ; \quad |LAR'(i)|<0.675 \\ sign[LAR'(i)]*[0.500*|LAR'(i)| +0.337500] & ; \quad 0.675<=|LAR'(i)|<1.225 \\ sign[LAR'(i)]*[0.125*|LAR'(i)| +0.796875] & ; \quad 1.225<=|LAR'(i)|<=1.625 \end{cases} \qquad (3.5)$$

3.1.7. Quantization and coding of Log.-Area Ratios

The Log.-Area Ratios LAR(i) have different dynamic ranges and different asymmetric distribution densities. For this reason, the transformed coefficients LAR(i) are limited and quantized differently according to the following equation (3.6), with $LAR_C(i)$ denoting the quantized and integer coded version of LAR(i).

$$LAR_C(i) = Nint\{A(i)*LAR(i) + B(i)\} \qquad (3.6)$$

with

$$Nint\{z\} := int\{z+sign\{z\}*0.5\} \qquad (3.6a)$$

Function Nint defines the rounding to the nearest integer value, with the coefficients A(i), B(i), and different extreme values of $LAR_C(i)$ for each coefficient LAR(i) given in table 3.1.

LAR No i	A(i)	B(i)	Minimum $LAR_C(i)$	Maximum $LAR_C(i)$
1	20.000	0.000	-32	+31
2	20.000	0.000	-32	+31
3	20.000	4.000	-16	+15
4	20.000	-5.000	-16	+15
5	13.637	0.184	- 8	+ 7
6	15.000	-3.500	- 8	+ 7
7	8.334	-0.666	- 4	+ 3
8	8.824	-2.235	- 4	+ 3

Table 3.1. Quantization of the Log.-Area Ratios LAR(i)

SHORT-TERM ANALYSIS FILTERING SECTION

The current frame of the speech signal s is retained in memory until calculation of the LPC parameters LAR(i) is completed. The frame is then read out and fed to the short term analysis filter of order p=8. However, prior to the analysis filtering operation, the filter coefficients are decoded and preprocessed by interpolation.

3.1.8. Decoding of the quantized Log.-Area Ratios

In this block the quantized and coded Log.-Area Ratios ($LAR_C(i)$) are decoded according to equation (3.7).

$$LAR''(i) = (LAR_C(i) - B(i))/ A(i) \qquad (3.7)$$

3.1.9. Interpolation of Log.-Area Ratios

To avoid spurious transients which may occur if the filter coefficients are changed abruptly, two subsequent sets of Log.-Area Ratios are interpolated linearly. Within each frame of 160 analysed speech samples the short term analysis filter and the short term synthesis filter operate with four different sets of coefficients derived according to table 3.2.

k	$LAR'_J(i) =$
0...12	$0.75*LAR''_{J-1}(i) + 0.25*LAR''_J(i)$
13...26	$0.50*LAR''_{J-1}(i) + 0.50*LAR''_J(i)$
27...39	$0.25*LAR''_{J-1}(i) + 0.75*LAR''_J(i)$
40..159	$LAR''_J(i)$

Table 3.2. Interpolation of LAR parameters (J=actual segment)

3.1.10. Transformation of Log.-Area Ratios into reflection coefficients

The reflection coefficients are finally determined using the inverse transformation according to equation (3.5).

3.1.11. Short Term Analysis Filtering

The Short term analysis filter is implemented according to the lattice structure depicted in fig 3.3.

$$d_0(k) = s(k) \qquad (3.8a)$$
$$u_0(k) = s(k) \qquad (3.8b)$$
$$d_i(k) = d_{i-1}(k) + r'_i * u_{i-1}(k-1) \quad \text{with } i=1,...8 \qquad (3.8c)$$
$$u_i(k) = u_{i-1}(k-1) + r'_i * d_{i-1}(k) \quad \text{with } i=1,...8 \qquad (3.8d)$$
$$d(k\ ) = d_8(k) \qquad (3.8e)$$

LONG-TERM PREDICTOR (LTP) SECTION

3.1.12. Sub-segmentation

Each input frame of the short term residual signal contains 160 samples, corresponding to 20 ms. The long term correlation is evaluated four times per frame, for each 5 ms subsegment. For convenience in the following, we note j=0,...,3 the sub-segment number, so that the samples pertaining to the j-th sub-segment of the residual signal are now denoted by $d(k_j+k)$ with $j = 0,...,3$; $k_j = k_0 + j*40$ and $k = 0,...,39$ where k_0 corresponds to the first value of the current frame.

3.1.13. Calculation of the LTP parameters

For each of the four sub-segments a long term correlation lag N_j, ($j=0,\dots,3$), and an associated gain factor b_j, ($j=0,\dots,3$) are determined. For each sub-segment, the determination of these parameters is implemented in three steps.

1) The first step is the evaluation of the cross-correlation $R_j(\text{lambda})$ of the current sub-segment of short term residual signal $d(k_j+i)$, ($i=0,\dots,39$) and the previous samples of the reconstructed short term residual signal $d'(k_j+i)$, ($i=-120,\dots,-1$):

$$R_j(\text{lambda}) = \sum_{i=0}^{39} d(k_j+i)*d'(k_j+i-\text{lambda}); \quad \begin{array}{l} j = 0,\dots 3 \\ k_j = k_0 + j*40 \\ \text{lambda} = 40,\dots,120 \end{array} \qquad (3.9)$$

The cross-correlation is evaluated for lags lambda greater than or equal to 40 and less than or equal to 120, ie corresponding to samples outside the current sub-segment and not delayed by more than two sub-segments.

2) The second step is to find the position N_j of the peak of the cross-correlation function within this interval:

$$R_j(N_j) = \max \{ R_j(\text{lambda});\ \text{lambda} = 40..120 \};$$
$$j = 0,\dots,3 \qquad (3.10)$$

3) The third step is the evaluation of the gain factor b_j according to:

$$b_j = R_j(N_j) \ / \ S_j(N_j); \qquad j = 0,\dots,3 \qquad (3.11)$$

with

$$S_j(N_j) = \sum_{i=0}^{39} d'^2 (k_j+i-N_j); \quad j = 0,\dots,3 \qquad (3.12)$$

It is clear that the last 120 samples of the reconstructed short term residual signal $d'(k_j+i), (i=-120,\ldots,-1)$ must be retained until the next sub-segment so as to allow the evaluation of the relations (3.9),...,(3.12).

3.1.14. Coding/Decoding of the LTP lags

The long term correlation lags $N_j, (j=0,\ldots,3)$ can have values in the range (40,...,120), and so must be coded using 7 bits with:

$$N_{cj} = N_j; \qquad j = 0,\ldots,3 \qquad (3.13)$$

At the receiving end, assuming an error free transmission, the decoding of these values will restore the actual lags:

$$N_j' = N_{cj}; \qquad j = 0,\ldots,3 \qquad (3.14)$$

3.1.15. Coding/Decoding of the LTP gains

The long term prediction gains $b_j, (j=0,\ldots,3)$ are encoded with 2 bits each, according to the following algorithm:

$$\begin{array}{llll} \text{if } & b_j \le DLB(i) & \text{then } b_{cj} = 0; & i=0 \\ \text{if } DLB(i-1) < & b_j \le DLB(i) & \text{then } b_{cj} = i; & i=1,2 \qquad (3.15) \\ \text{if } DLB(i-1) < & b_j & \text{then } b_{cj} = 3; & i=3 \end{array}$$

where $DLB(i), (i=0,\ldots,2)$ denotes the decision levels of the quantizer, and b_{cj} represents the coded gain value. Decision levels and quantizing levels are given in table 3.3.

i	Decision level DLB(i)	Quantizing level QLB(i)
0	0.2	0.10
1	0.5	0.35
2	0.8	0.65
3		1.00

Table 3.3. Quantization table for the LTP gain

The decoding rule is implemented according to:

$$b_j' = QLB(b_{cj}) \quad ; \quad j = 0,\ldots,3 \qquad (3.16)$$

where QLB(i),(i=0,...,3) denotes the quantizing levels, and b_j' represents the decoded gain value (see table 3.3).

3.1.16. Long term analysis filtering

The short term residual signal $d(k_0+k)$,(k=0,...,159) is processed by sub-segments of 40 samples. From each of the four sub-segments (j=0,...,3) of short term residual samples, denoted here $d(k_j+k)$, (k=0,...,39), an estimate $d''(k_j+k)$, (k=0,...,39) of the signal is subtracted to give the long term residual signal $e(k_j+k)$, (k=0,...,39) (see fig 3.1):

$$e(k_j+k) = d(k_j+k) - d''(k_j+k) \quad ; \quad \begin{array}{l} j = 0,\ldots,3 \\ k = 0,\ldots,39 \\ k_j = k_0 + j*40 \end{array} \qquad (3.17)$$

Prior to this subtraction, the estimated samples $d''(k_j+k)$ are computed from the previously reconstructed short term residual samples d', adjusted to the current sub-segment LTP lag N_j' and weighted with the sub-segment LTP gain b_j':

$$d''(k_j+k) = b_j'*d'(k_j+k-N_j') \quad ; \quad \begin{array}{l} j = 0,\ldots,3 \\ k = 0,\ldots,39 \\ k_j = k_0 + j*40 \end{array} \qquad (3.18)$$

3.1.17. Long term synthesis filtering

The reconstructed long term residual signal $e'(k_0+k)$,(k=0,...,159) is processed by sub-segments of 40 samples. To each sub-segment, denoted here $e'(k_j+k)$, (k=0,...,39), the estimate $d''(k_j+k)$, (k=0,...,39) of the signal is added to give the reconstructed short term residual signal $d'(k_j+k)$,(k=0,...,39):

$$d'(k_j+k) = e'(k_j+k) + d''(k_j+k) \quad ; \quad \begin{array}{l} j = 0,\ldots,3 \\ k = 0,\ldots,39 \\ k_j = k_0 + j*40 \end{array} \qquad (3.19)$$

RPE ENCODING SECTION

3.1.18. Weighting Filter

A FIR 'block filter' algorithm is applied to each sub-segment by convolving 40 samples e(k) with the impulse response H(i) ; i=0,...,10 (see table 3.4).

i	5	4 (6)	3 (7)	2 (8)	1 (9)	0 (10)
$H(i)*2^{13}$	8192	5741	2054	0	-374	-134

|H(Omega=0)| = 2.779;

Table 3.4. Impulse response of block filter (weighting filter)

The conventional convolution of a sequence having 40 samples with an 11-tap impulse response would produce 40+11-1=50 samples. In contrast to this, the 'block filter' algorithm produces the 40 central samples of the conventional convolution operation. For notational convenience the block filtered version of each sub-segment is denoted by x(k), k=0,...,39.

$$x(k) = \sum_{i=0}^{10} H(i) * e(k+5-i) \qquad \text{with } k = 0,\dots,39 \qquad (3.20)$$

NOTE: e(k+5-i) = 0 for k+5-i<0 and k+5-i>39.

3.1.19. Adaptive sample rate decimation by RPE grid selection

For the next step, the filtered signal x is down-sampled by a ratio of 3 resulting in 3 interleaved sequences of lengths 14, 13 and 13, which are split up again into 4 sub-sequences x_m of length 13:

$$x_m(i) = x(k_j+m+3*i) \qquad ; \quad i = 0,\dots,12 \quad m = 0,\dots,3 \qquad (3.21)$$

with m denoting the position of the decimation grid. According to the explicit solution of the RPE mean squared error criterion, the optimum candidate sub-sequence x_M is selected which is the one with the maximum energy

$$E_M = \max_{m} \sum_{i=0}^{12} x_m^2(i) \quad ; \quad m = 0,\dots,3 \qquad (3.22)$$

The optimum grid position M is coded as M_c with 2 bits.

3.1.20. APCM quantization of the selected RPE sequence

The selected sub-sequence $x_M(i)$ (RPE sequence) is quantized, applying APCM (Adaptive Pulse Code Modulation). For each RPE sequence consisting of a set of 13 samples $x_M(i)$,the maximum x_{max} of the absolute values $|x_M(i)|$ is selected and quantized logarithmically with 6 bits as x_{maxc} as given in table 3.5.

x_{max}	x'_{max}	x_{maxc}	x_{max}	x'_{max}	x_{maxc}
0 .. 31	31	0	2048 .. 2303	2303	32
32 .. 63	63	1	2304 .. 2559	2559	33
64 .. 95	95	2	2560 .. 2815	2815	34
96 .. 127	127	3	2816 .. 3071	3071	35
128 .. 159	159	4	3072 .. 3327	3327	36
160 .. 191	191	5	3328 .. 3583	3583	37
192 .. 223	223	6	3584 .. 3839	3839	38
224 .. 255	255	7	3840 .. 4095	4095	39
256 .. 287	287	8	4096 .. 4607	4607	40
288 .. 319	319	9	4608 .. 5119	5119	41
320 .. 351	351	10	5120 .. 5631	5631	42
352 .. 383	383	11	5632 .. 6143	6143	43
384 .. 415	415	12	6144 .. 6655	6655	44
416 .. 447	447	13	6656 .. 7167	7167	45
448 .. 479	479	14	7168 .. 7679	7679	46
480 .. 511	511	15	7680 .. 8191	8191	47
512 .. 575	575	16	8192 .. 9215	9215	48
576 .. 639	639	17	9216 .. 10239	10239	49
640 .. 703	703	18	10240 .. 11263	11263	50
704 .. 767	767	19	11264 .. 12287	12287	51
768 .. 831	831	20	12288 .. 13311	13311	52
832 .. 895	895	21	13312 .. 14335	14335	53
896 .. 959	959	22	14336 .. 15359	15359	54
960 .. 1023	1023	23	15360 .. 16383	16383	55
1024 .. 1151	1151	24	16384 .. 18431	18431	56
1152 .. 1279	1279	25	18432 .. 20479	20479	57
1280 .. 1407	1407	26	20480 .. 22527	22527	58
1408 .. 1535	1535	27	22528 .. 24575	24575	59
1536 .. 1663	1663	28	24576 .. 26623	26623	60
1664 .. 1791	1791	29	26624 .. 28671	28671	61
1792 .. 1919	1919	30	28672 .. 30719	30719	62
1920 .. 2047	2047	31	30720 .. 32767	32767	63

Table 3.5. Quantization of the block maximum x_{max}

For the normalization, the 13 samples are divided by the decoded version x'_{max} of the block maximum. Finally, the normalized samples

$$x'(i) = x_M(i)/x'_{max} \; ; \qquad i=0,\dots,12 \qquad (3.23)$$

are quantized uniformly with three bits to $x_{Mc}(i)$ as given in table 3.6.

$x'*2^{15}$ (Interval-limits)	$\underline{x}_M'*2^{15}$	x_{Mc} (Channel)
-32768 ... -24577	-28672	0 = 000
-24576 ... -16385	-20480	1 = 001
-16384 ... -8193	-12288	2 = 010
-8192 ... -1	-4096	3 = 011
0 ... 8191	4096	4 = 100
8192 ... 16383	12288	5 = 101
16384 ... 24575	20480	6 = 110
24576 ... 32767	28672	7 = 111

Table 3.6. Quantization of the normalized RPE-samples

3.1.21. APCM inverse quantization

The $x_{Mc}(i)$ are decoded to $\underline{x}_M'(i)$ and denormalized using the decoded value x'_{maxc} leading to the decoded sub-sequence $x'_M(i)$.

3.1.22. RPE grid positioning

The quantized sub-sequence is upsampled by a ratio of 3 by inserting zero values according to the grid position given with Mc.

3.2. DECODER

The decoder comprises the following 4 sections. Most of the sub-blocks are also needed in the encoder and have been described already. Only the short term tynthesis filter and the deemphasis filter are added in the decoder as new sub-blocks.

* RPE decoding section (3.2.1)
* Long Term Prediction section (3.2.2)
* Short term synthesis filtering section (3.2.3)
* Postprocessing (3.2.4)

The complete block diagram for the decoder is shown in fig 3.4. The variables and parameters of the decoder are marked by the index r to distinguish the received values from the encoder values.

3.2.1. RPE decoding section

The input signal of the long term synthesis filter (reconstruction of the long term residual signal) is formed by decoding and denormalizing the RPE-samples (APCM inverse quantization - 3.1.21) and by placing them in the correct time position (RPE grid positioning - 3.1.22). At this stage, the sampling frequency is increased by a factor of 3 by inserting the appropriate number of intermediate zero-valued samples.

3.2.2. Long Term Prediction section

The the reconstructed long term residual signal e_r' is applied to the long term synthesis filter (see 3.1.16 and 3.1.17) which produces the reconstructed short term residual signal d_r' for the short term synthesiser.

3.2.3. Short term synthesis filtering section

The coefficients of the short term synthesis filter (see fig 3.5) are reconstructed applying the identical procedure to that in the encoder (3.1.8 - 3.1.10). The short term synthesis filter is implemented according to the lattice structure depicted in fig 3.5.

$s_{r(0)}(k) = d_r'(k)$ (3.24a)

$s_{r(i)}(k) = s_{r(i-1)}(k) - r_r'(9-i) * v_{8-i}(k-1);$ $i=1,\ldots,8$ (3.24b)

$v_{9-i}(k) = v_{8-i}(k-1) + r_r'(9-i) * s_{r(i)}(k);$ $i=1,\ldots,8$ (3.24c)

$s_r'(k) = s_{r(8)}(k)$ (3.24d)

$v_0(k) = s_{r(8)}(k)$ (3.24e)

3.2.4. Postprocessing

The output of the synthesis filter $s_r(k)$ is fed into the IIR-deemphasis filter leading to the output signal s_{ro}.

$s_{ro}(k) = s_r(k) + beta*s_{ro}(k-1)$; $beta= 28180*2^{-15}$ (3.25)

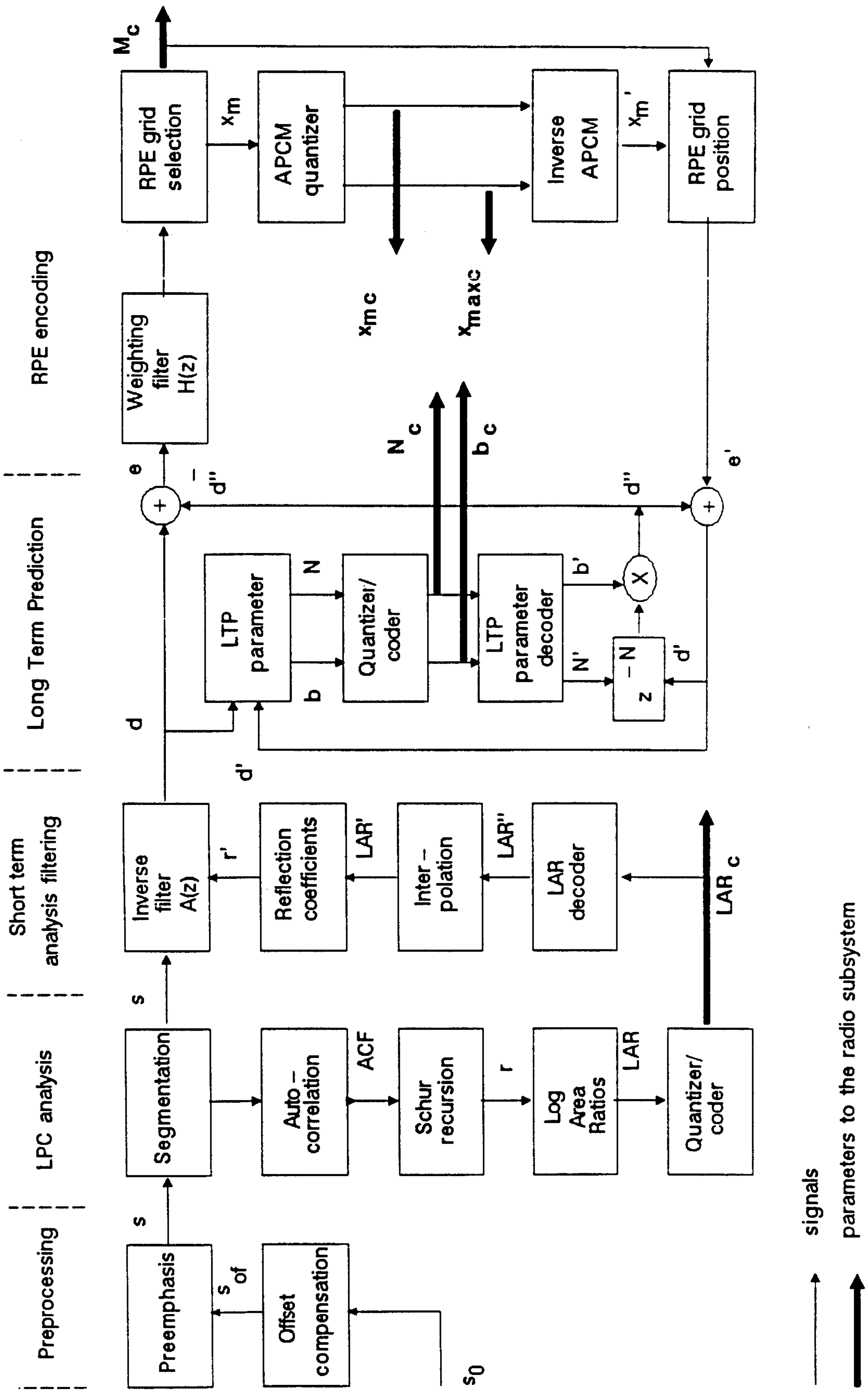
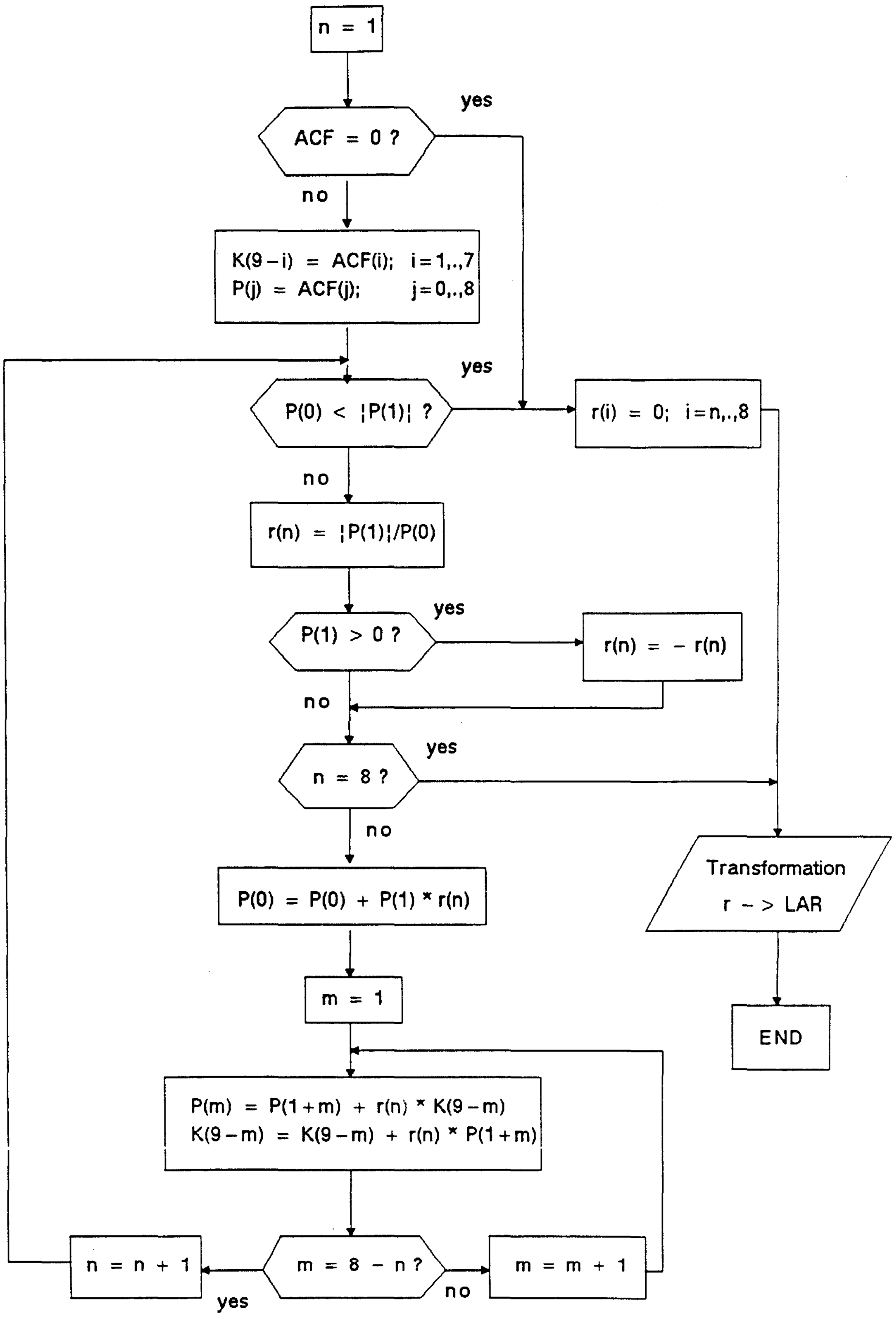


Fig 3.1. Block diagram of the RPE – LTP encoder



<u>Fig 3.2. LPC analysis using Schur recursion</u>

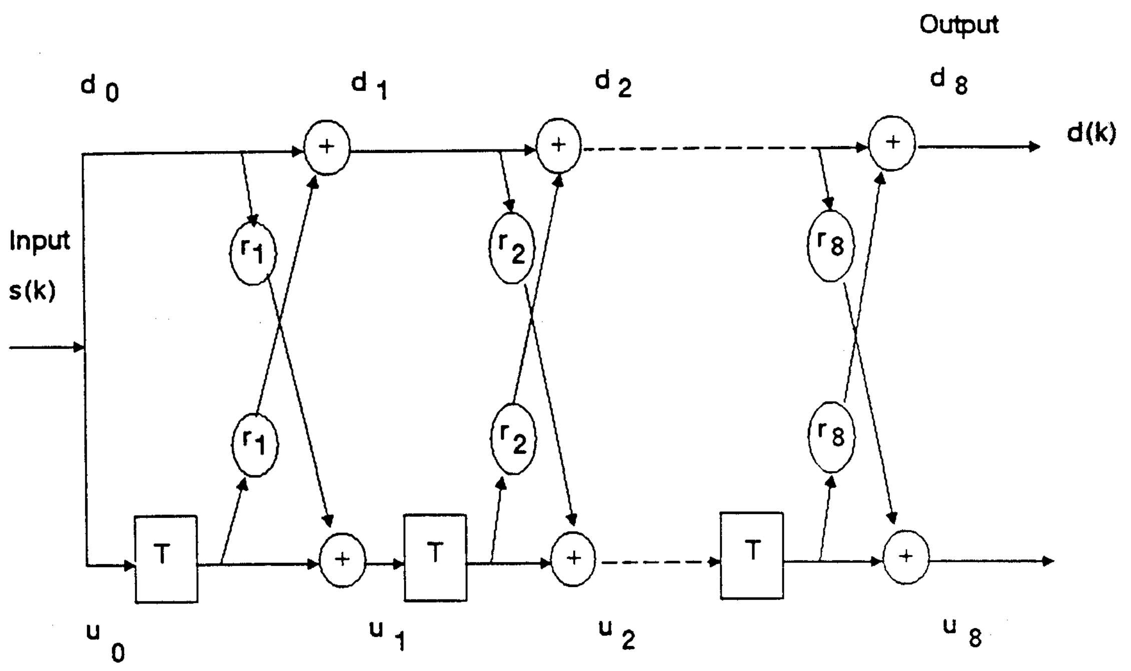


Fig 3.3. Short term analysis filter

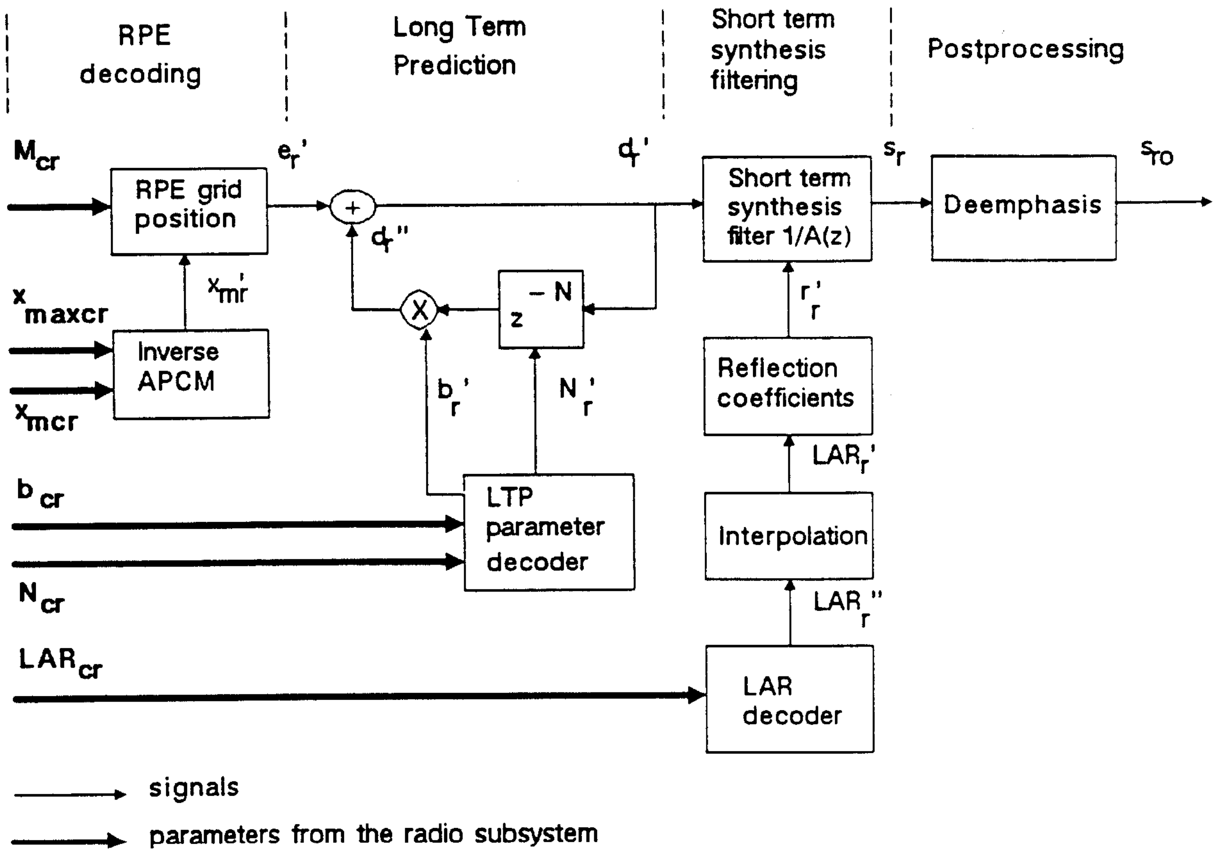


Fig 3.4. Block diagram of the RPE – LTP decoder

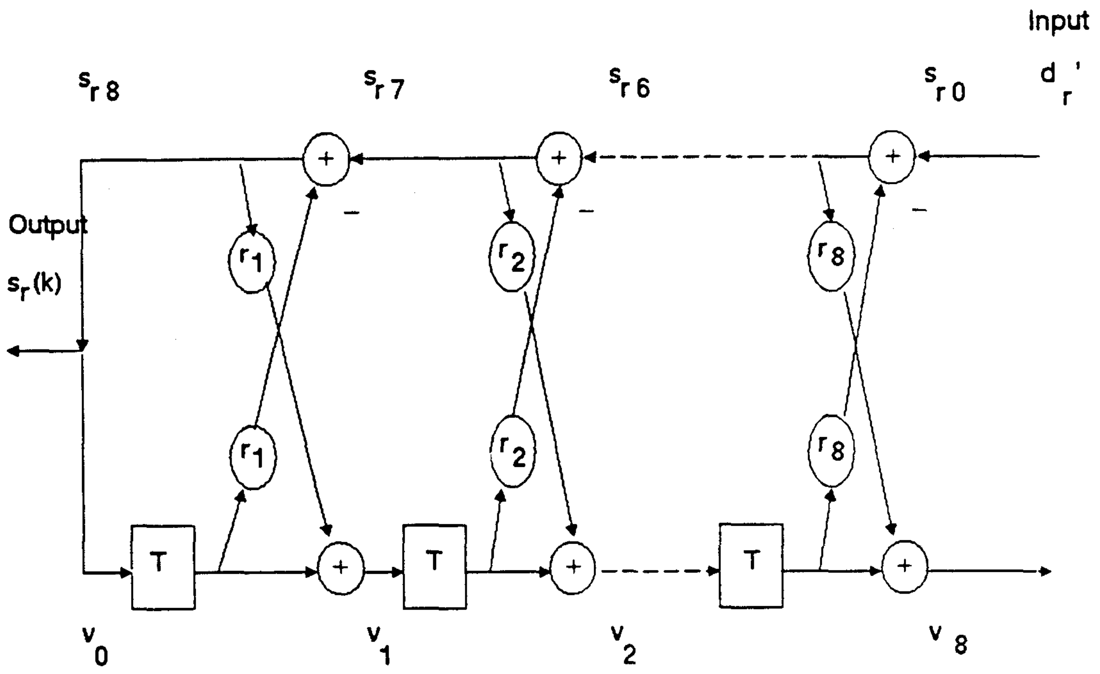


Fig 3.5. Short term synthesis filter

4. COMPUTATIONAL DETAILS OF THE RPE-LTP CODEC

4.1. DATA REPRESENTATION AND ARITHMETIC OPERATIONS

Only two types of variables are used along the implementation of the RPE-LTP algorithm in fixed point arithmetic. These two types are:

Integer on 16 bits;

Long integer on 32 bits;

This assumption simplifies the detailed description and allows the maximum reach of precision.

In different places of the recommendation, different scaling factors are used according to different operations. To help the reader in the comparison of corresponding floating point and fixed point values given in section 3 and 4 comments of the format:

```
/* var  = integer(  real_var * scalefactor ) */
```

are used at several points of section 4. var is the rounded fixed point representation of the floating point representation of var (real_var) using the given scaling factor.

In the description, input signal samples, coded parameters and output signal samples are represented by 16 bit words. At the receiving part it must therefore be ensured that only valid bits (13 bits for samples signal and two to seven bits for coded parameters) are used. In verification tests, the testing system may introduce random bit at non valid places inside these samples (3 LSBs) or parameters (MSBs) to test this function. In the digital test sequences all non valid bits are set to 0.

The following part of this section describes the required set of arithmetic operations to implement the RPE-LTP algorithm in fixed point.

For arithmetics operations or variables with a long integer type (32 bit) a prefix L_ is used in order to distinguish them from the 16 bit variables or arithmetic operations.

All the names of the variables are identical to those of the functional description of the RPE-LTP Codec (Section 3) but variables like x', x'' are respectively called:

```
x' -----> xp
x''-----> xpp
```

in order to avoid any confusing notation.

NOTE: The x',x" variables are examples but are not used within the following description.

The following notations are used in the arithmetic operations:

Square brackets ([..]) are used for arrays and when needed, the starting index and the ending index are put inside the bracket. For example x[0..159] means that x is an array of 160 words of 16 bits with beginning index 0 and ending index 159 and x[k] is an element of the array x[0..159].

All functions' names are underlined. For example add(x, y) means that we perform the addition of x and y.

<< n :	denotes a n-bit arithmetic shift left operation (zero fill) on variables of type short or long; if n is less than 0, this operation becomes an arithmetic right shift of -n.
>> n :	denotes a n-bit arithmetic right shift operation (sign extension) on variables of type short or long; if n is less than 0, this operation becomes an arithmetic left shift of -n (zero fill).
a > b :	denotes the "greater than" condition;
a >= b :	denotes the "greater than or equal" condition;
a < b :	denotes the "less than" condition;
a <= b :	denotes the "less than or equal" condition;
a == b :	denotes the "equal to" condition.

The basic structure of the FOR-NEXT loop is used in this description for loop computation; the declaration is:

```
|== FOR k= start to end:
|     inner computation;
|== NEXT k:
```

Also the IF.. ELSE IF structure is used throughout this detailed description. The basic structure is:

```
       IF (condition1) THEN statement1;

          ELSE IF ( condition2) THEN statement2;

               ELSE IF ( condition3) THEN statement3;
```

The word EXIT is used to exit immediately from a procedure.

The following arithmetic operations are defined:

<u>add</u>(var1, var2)	: performs the addition (var1+var2) with : overflow control and saturation; the result : is set at +32767 when overflow occurs or at : -32768 when underflow occurs.
<u>sub</u>(var1, var2)	: performs the subtraction (var1-var2) with : overflow control and saturation; the result : is set at +32767 when overflow occurs or at : -32768 when underflow occurs.
<u>mult</u>(var1, var2)	: performs the multiplication of var1 by var2 : and gives a 16 bits result which is scaled ie : <u>mult</u>(var1,var2) = (var1 times var2) >> 15 : and <u>mult</u>(-32768, -32768) = 32767
<u>mult_r</u>(var1, var2)	: same as <u>mult</u> but with rounding ie : <u>mult_r</u>(var1, var2) = : ((var1 times var2) + 16384) >> 15 : and <u>mult_r</u>(-32768, -32768) = 32767
<u>abs</u>(var1)	: absolute value of var1; <u>abs</u>(-32768) = 32767
<u>div</u>(var1, var2)	: <u>div</u> produces a result which is the : fractional integer division of var1 by var2; : var1 and var2 must be positive and var2 must : be greater or equal to var1; The result is : positive (leading bit equal to 0) and : truncated to 16 bits. if var1 == var2 then : <u>div</u>(var1, var2) = 32767
<u>L_mult</u>(var1, var2)	: <u>L_mult</u> is a 32 bit result for the : multiplication of var1 times var2 with a one : bit shift left. <u>L_mult</u>(var1, var2) = | (var1 times var2) << 1. The condition | <u>L_mult</u>(-32768, -32768) does not occur in | the algorithm.

L_add(L_var1, L_var2)	32 bits addition of two 32 bits variables (L_var1 + L_var2) with overflow control and saturation; the result is set at 2147483647 when overflow occurs and at -2147483648 when underflow occurs.
L_sub(L_var1,L_var2)	32 bits subtraction of two 32 bits variables (L_var1 - L_var2) with overflow control and saturation; the result is set at 2147483647 when overflow occurs and at -2147483648 when underflow occurs.
norm(L_var1)	norm produces the number of left shifts needed to normalize the 32 bits variable L_var1 for positive values on the interval with minimum of 1073741824 and maximum of 2147483647 and for negative values on the interval with minimum of -2147483648 and maximum of -1073741824; in order to normalize the result, the following operation must be done: L_norm_var1 = L_var1 << norm(L_var1)
L_var2 = var1;	deposit the 16 bits of var1 in the LSB 16 bits of L_var2 with sign extension.
var2 = L_var1;	extract the 16 LSB bits of L_var1 to put in var2.

When a constant is used in an operation on 32 bits, it must be first sign-extended on 32 bits.

4.2. FIXED POINT IMPLEMENTATION OF THE RPE-LTP CODER

The RPE-LTP coder works on a frame by frame basis. The length of the frame is equal to 160 samples. Some computations are done once per frame (analysis) and some others for each of the four sub-segments (40 samples).

In the following detailed description, procedure 4.2.0 to 4.2.10 are done once per frame to produce at the output of the coder the LARc[1..8] parameters which are the coded LAR coefficients and also to realize the inverse filtering operation for the entire frame (160 samples of signal d[0..159]). These parts produce at the output of the coder:

```
| LARc[1..8]      : Coded LAR coefficients

|--> These parameters are calculated and sent once per
     frame.
```

Procedure 4.2.11 to 4.2.18 are to be executed four times per frame. That means once for each sub-segment RPE-LTP analysis of 40 samples. These parts produce at the output of the coder:

```
| Nc          : LTP lag;

| bc          : Coded LTP gain;

| Mc          : RPE grid selection;

| xmaxc           : Coded maximum amplitude of the RPE
                    sequence;

| xMc[0..12]      : Codes of the normalized RPE samples;

|--> These parameters are calculated and sent four times
     per frame.
```

PREPROCESSING SECTION

4.2.0. Scaling of the input variable

After A-law to linear conversion (or directly from the A to D converter) the following scaling is assumed for input to the RPE-LTP algorithm:

S.v.v.v.v.v.v.v.v.v.v.v.v.x.x.x (2's complement format).

Where S is the sign bit, v a valid bit, and x a "don't care" bit.

The original signal is called sop[..];

4.2.1. Downscaling of the input signal

```
|== FOR k=0 to 159:
|    so[k]  = sop[k] >> 3;
|    so[k]  = so[k] << 2;
|== NEXT k:
```

4.2.2. Offset compensation

This part implements a high-pass filter and requires extended arithmetic precision for the recursive part of this filter.

The input of this procedure is the array so[0..159] and the output the array sof[0..159].

```
|== FOR k = 0 to 159:
| Compute the non-recursive part.
|    s1 = sub( so[k], z1 );
|    z1 = so[k];
| Compute the recursive part.
|    L_s2 = s1;
|    L_s2 = L_s2 << 15;
| Execution of a 31 by 16 bits multiplication.
|    msp = L_z2 >> 15;
|    lsp = L_sub( L_z2, ( msp << 15 ) );
|    temp = mult_r( lsp, 32735 );
|    L_s2 = L_add( L_s2, temp );
|    L_z2 = L_add(  L_mult( msp, 32735 ) >> 1, L_s2 );
| Compute sof[k] with rounding.
|    sof[k] =  L_add( L_z2, 16384 )  >> 15;
|== NEXT k:
```

Keep z1 and L-z2 in memory for the next frame.

Initial value: z1=0; L_z2=0;

4.2.3. Preemphasis

```
|== FOR k=0 to 159:
|    s[k] = add( sof[k], mult_r( mp, -28180 ) );
|    mp = sof[k];
|== NEXT k:
```

Keep mp in memory for the next frame.

Initial value: mp=0;

LPC ANALYSIS SECTION

4.2.4. Autocorrelation

The goal is to compute the array L_ACF[k]. The signal s[i] must be scaled in order to avoid an overflow situation.

Dynamic scaling of the array s[0..159].

Search for the maximum.

```
smax = 0;
|== FOR k = 0 to 159:
|    temp = abs( s [k] );
|    IF ( temp > smax ) THEN smax = temp;
|== NEXT k;
```

Computation of the scaling factor.

```
IF ( smax == 0 ) THEN scalauto = 0;
  ELSE  scalauto = sub( 4, norm( smax << 16 )  );
```

Scaling of the array s[0..159].

```
IF ( scalauto > 0 ) THEN
                    | temp = 16384 >> sub( scalauto,1);
                    |== FOR k = 0 to 159:
                    |    s[k] = mult_r( s[k], temp);
                    |== NEXT k:
```

Compute the L-ACF[..].

```
|== FOR k=0 to 8:
|    L_ACF[k] = 0;
|==== FOR i=k to 159:
|      L_temp = L_mult( s[i], s[i-k] );
|      L_ACF[k] = L_add( L_ACF[k], L_temp );
|==== NEXT i:
|== NEXT k:
```

Rescaling of the array s[0..159].

```
        IF ( scalauto > 0 ) THEN
                                        |== FOR k = 0 to 159:
                                        |     s[k] = s[k] << scalauto;
                                        |== NEXT k:
```

4.2.5. Computation of the reflection coefficients

Schur recursion with 16 bits arithmetic.

```
        IF( L_ACF[0] == 0 ) THEN
                                        |== FOR i = 1 to 8:
                                        |     r[i] = 0;
                                        |== NEXT i:
                                        |     EXIT; /continue with section
                                                          4.2.6/
        temp = norm( L_ACF[0] );
        |== FOR k=0 to 8:
        |     ACF[k] = ( L_ACF[k] << temp ) >> 16;
        |== NEXT k:
```

Initialize array P[..] and K[..] for the recursion.

```
        |== FOR i=1 to 7:
        |     K[9-i] = ACF[i];
        |== NEXT i:
```

```
|== FOR i=0 to 8:
|    P[i] = ACF[i];
|== NEXT i:
```

Compute reflection coefficients.

```
|== FOR n=1 to 8:
|    IF( P[0] < abs( P[1] ) ) THEN
|                                        |== FOR i = n to 8:
|                                        |    r[i] = 0;
|                                        |== NEXT i:
|                                        | EXIT; /continue with
|                                        |       section 4.2.6/
|    r[n] = div( abs( P[1] ), P[0] );
|    IF ( P[1] > 0 ) THEN r[n] = sub( 0, r[n] );
|
|    IF (  n ==  8 ) THEN EXIT; /continue with
                                 section 4.2-6/
|    Schur recursion.
|    P[0] = add( P[0], mult_r( P[1], r[n] ) );
|==== FOR m=1 to 8-n:
|      P[m] = add( P[m+1], mult_r( K[9-m], r[n] ) );
|      K[9-m] = add( K[9-m], mult_r( P[m+1], r[n] ) );
|==== NEXT m:
|
|== NEXT n:
```

NOTE: The following lines gives one correct implementation of the div(num, denum) arithmetic operation. Compute div which is the integer division of num by denum: with denum >= num > 0.

```
L_num = num;

L_denum = denum;

div =0;

|== FOR k = 0 to 14:
|    div= div << 1;
|    L_num = L_num << 1;
|    IF ( L_num >= L_denum) THEN
|                            | L_num=L_sub(L_num, L_denum);
|                            | div = add( div ,1 );
|== NEXT k:
```

4.2.6. Transformation of reflection coefficients to Log.-Area Ratios

The following scaling for r[..] and LAR[..] has been used:

```
/* r[..] = integer( real_r[..]*32768. );  -1. <= real_r <1. */
/*                                                          */
/* LAR[..] = integer( real_LAR[..]*16384. );                */
/*                                                          */
/* with  -1.625 <= real_LAR <= 1.625                        */
```

Computation of the LAR[1..8] from the r[1..8].

```
|== FOR i = 1 to 8:
|    temp = abs( r[i] );
|    IF ( temp < 22118 ) THEN temp = temp >> 1;
|       ELSE IF ( temp < 31130 ) THEN
|              temp= sub(temp, 11059);
|          ELSE temp = sub( temp, 26112 ) << 2;
|    LAR[i] = temp;
|    IF ( r[i] < 0 ) THEN LAR[i] = sub( 0, LAR[i] );
|== NEXT i:
```

4.2.7. Quantization and coding of the Log.-Area Ratios

This procedure needs four tables; the following equations give the optimum scaling for the constants:

```
/* A[1..8]= integer( real_A[1..8]*1024); 8 values (see table4.1)*/
/*                                                              */
/* B[1..8]= integer( real_B[1..8]*512);  8 values (see table4.1)*/
/*                                                              */
/* MAC[1..8]= maximum of the LARc[1..8]; 8 values (see table4.1)*/
/*                                                              */
/* MIC[1..8]= minimum of the LARc[1..8]; 8 values (see table4.1)*/
```

Computation for quantizing and coding the LAR[1..8].

```
|== FOR i =1 to 8:
|    temp= mult( A[i], LAR[i] );
|    temp= add( temp, B[i] );
|    temp= add( temp, 256);     for rounding
|    LARc[i]= temp >> 9;
|  Check IF LARc[i] lies between MIN and MAX
|     IF ( LARc[i] > MAC[i] ) THEN LARc[i] = MAC[i];
|     IF ( LARc[i] < MIC[i] ) THEN LARc[i] = MIC[i];
|    LARc[i] = sub( LARc[i], MIC[i] );  /See note below/
|== NEXT i:
```

NOTE: The equation is used to make all the LARc[i] positive.

SHORT TERM ANALYSIS FILTERING SECTION

4.2.8. Decoding of the coded Log.-Area Ratios

This procedure requires for efficient implementation two tables.

```
/* INVA[1..8]=integer((32768*8)/(real_A[1..8]);                 */
/*                                       8 values (table 4.2 )  */
/* MIC[1..8]=minimum value of the LARc[1..8];                   */
/*                                       8 values (table 4.1)   */
```

Compute the LARpp[1..8].

```
|== FOR i=1 to 8:
|     temp1 = add( LARc[i], MIC[i] ) << 10; /See note below/
|     temp2 = B[i] << 1;
|     temp1 = sub( temp1, temp2);
|     temp1 = mult_r( INVA[i], temp1);
|     LARpp[i] = add( temp1, temp1);
|== NEXT i:
```

NOTE: The addition of MIC[i] is used to restore the sign of LARc[i].

4.2.9. Computation of the quantized reflection coefficients

Within each frame of 160 analysed speech samples the short term analysis and synthesis filters operate with four different sets of coefficients, derived from the previous set of decoded LARs(LARpp(j-1)) and the actual set of decoded LARs (LARpp(j)).

4.2.9.1. Interpolation of the LARpp[1..8] to get the LARp[1..8]

For k_start = 0 to k_end = 12.

```
|==== FOR i= 1 to 8:
|      LARp[i] = add((LARpp(j-1)[i] >> 2),(LARpp(j)[i] >> 2));
|      LARp[i] = add( LARp[i] , ( LARpp(j-1)[i]  >> 1 ) );
|==== NEXT i:
```

For k_start = 13 to k_end = 26.

```
|==== FOR i= 1 to 8:
|      LARp[i] = add((LARpp(j-1)[i] >> 1),(LARpp(j)[i] >> 1 ));
|==== NEXT i:
```

For k_start = 27 to k_end = 39.

```
|==== FOR i= 1 to 8:
|     LARp[i] = add((LARpp(j-1)[i] >> 2),(LARpp(j)[i] >> 2 ));
|     LARp[i] = add( LARp[i] , ( LARpp(j)[i]  >> 1 ) );
|==== NEXT i:
```

For k_start = 40 to k_end = 159.

```
|==== FOR i= 1 to 8:
|     LARp[i] =  LARpp(j)[i];
|==== NEXT i:
```

Initial value: LARpp(j-1)[1..8]=0;

4.2.9.2. Computation of the rp[1..8] from the interpolated LARp[1..8]

The input of this procedure is the interpolated LARp[1..8] array. The reflection coefficients, rp[i], are used in the analysis filter and in the synthesis filter.

```
|== FOR i=1 to 8:
|    temp = abs( LARp[i] );
|    IF ( temp < 11059 ) THEN temp = temp << 1;
|        ELSE IF (temp < 20070) THEN
|                 temp = add(temp, 11059);
|           ELSE temp = add( (temp >> 2), 26112 );
|    rp[i] = temp;
|    IF ( LARp[i] < 0 ) THEN rp[i] = sub( 0, rp[i] );
|== NEXT i:
```

4.2.10. Short term analysis filtering

This procedure computes the short term residual signal d[..] to be fed to the RPE-LTP loop from the s[..] signal and from the local rp[..] array (quantized reflection coefficients). As the call of this procedure can be done in many ways (see the interpolation of the LAR coefficient), it is assumed that the computation begins with index k_start (for arrays d[..] and s[..]) and stops with index k_end (k_start and k_end are defined in 4.2.9.1). This procedure also needs to keep the array u[0..7] in memory for each call.

```
|== FOR k = k_start to k_end:
|    di = s[k]
|    sav = di;
|==== FOR i = 1 to 8:
|      temp = add( u[i-1], mult_r( rp[i], di ) );
|      di = add( di, mult_r( rp[i], u[i-1] ) );
|      u[i-1] = sav;
|      sav = temp;
|==== NEXT i:
|    d[k] = di;
|== NEXT k:
```

Keep the array u[0..7] in memory.

Initial value: u[0..7]=0;

LONG TERM PREDICTOR (LTP) SECTION

4.2.11. Calculation of the LTP parameters

This procedure computes the LTP gain (bc) and the LTP lag (Nc) for the long term analysis filter. This is done by calculating a maximum of the cross-correlation function between the current sub-segment short term residual signal d[0..39] (output of the short term analysis filter; for simplification the index of this array begins at 0 and ends at 39 for each sub-segment of the RPE-LTP analysis) and the previous reconstructed short term residual signal dp[-120..-1]. A dynamic scaling must be performed to avoid overflow.

Search of the optimum scaling of d[0..39].

```
dmax = 0;
|== FOR k = 0 to 39:
|     temp = abs( d[k] );
|     IF ( temp > dmax ) THEN dmax = temp;
|== NEXT k:

temp = 0;
IF ( dmax == 0 ) THEN scal = 0;
  ELSE temp = norm( dmax << 16 );
IF ( temp > 6 ) THEN scal = 0;
  ELSE scal = sub( 6, temp );
```

Initialisation of a working array wt[0..39].

```
|== FOR k = 0 to 39:
|     wt[k] = d[k] >> scal;
|== NEXT k:
```

Search for the maximum cross-correlation and coding of the LTP lag.

```
L_max = 0;
Nc = 40;    (index for the maximum cross-correlation)

|== FOR lambda = 40 to 120:
|    L_result = 0;
|==== FOR k = 0 to 39:
|      L_temp = L_mult( wt[k], dp[k-lambda] );
|      L_result = L_add( L_temp, L_result );
|==== NEXT k:
|    IF ( L_result > L_max) THEN
|                                       | Nc = lambda;
|                                       | L_max = L_result ;
|== NEXT lambda:
```

Rescaling of L-max.

```
L_max = L_max >> ( sub( 6, scal ) );
```

Initialisation of a working array wt[0..39].

```
|== FOR k = 0 to 39:
|    wt[k] = dp[k-Nc] >> 3;
|== NEXT k:
```

Compute the power of the reconstructed short term residual signal dp[..].

```
L_power = 0;
|== FOR k =0 to 39:
|     L_temp = L_mult( wt[k], wt[k] );
|     L_power = L_add( L_temp, L_power );
|== NEXT k:
```

Normalization of L-max and L-power.

```
IF ( L_max <= 0 ) THEN
                               | bc = 0;
                               | EXIT; /cont. with 4.2-12/
IF ( L_max >= L_power ) THEN
                                     | bc = 3;
                                     | EXIT; /cont. with 4.2-12/
temp = norm( L_power );
R = ( L_max << temp ) >> 16;
S = ( L_power << temp ) >> 16;
```

Coding of the LTP gain.

Table 4.3a must be used to obtain the level DLB[i] for the quantization of the LTP gain b to get the coded version bc.

```
|== FOR bc = 0 to 2:
|       IF (R <= mult(S, DLB[bc])) THEN EXIT; /cont. with
                                                4.2.12/
|== NEXT bc;
bc = 3;
```

Initial value: dp[-120..-1]=0;

4.2.12. Long term analysis filtering

In this part, we have to decode the bc parameter to compute the samples of the estimate dpp[0..39]. The decoding of bc needs the use of table 4.3b. The long term residual signal e[0..39] is then calculated to be fed to the RPE encoding section.

Decoding of the coded LTP gain.

```
        bp = QLB[bc];
```

Calculating the array e[0..39] and the array dpp[0..39].

```
        |== FOR k = 0 to 39:
        |     dpp[k] = mult_r( bp, dp[k-Nc] );
        |     e[k] = sub( d[k], dpp[k] );
        |== NEXT k:
```

RPE ENCODING SECTION

4.2.13. Weighting filter

The coefficients of the weighting filter are stored in a table (see table 4.4). The following scaling is used:

```
        /* H[0..10] = integer( real_H[0..10]*8192 ); */
```

Initialisation of a temporary working array wt[0..49].

```
        |== FOR k= 0 to 4:
        |     wt[k] = 0;
        |== NEXT k:
```

```
|== FOR k = 5 to 44:
|    wt[k] = e[k-5];
|== NEXT k:


|== FOR k= 45 to 49:
|    wt[k] = 0;
|== NEXT k:
```

Compute the signal x[0..39].

```
|== FOR k= 0 to 39:
|       L_result =  8192; /rounding of the output
                                  of the filter/
|==== FOR i = 0 to 10:
|       L_temp = L_mult( wt[k+i], H[i] );
|       L_result = L_add( L_result, L_temp );
|==== NEXT i:
|       L_result = L_add(L_result,L_result); /scaling (x2)/
|       L_result = L_add(L_result,L_result); /scaling (x4)/
|    x[k] = L_result >> 16;
|== NEXT k:
```

4.2.14. RPE grid selection

The signal x[0..39] is used to select the RPE grid which is represented by Mc.

```
EM =0;

Mc = 0;


|== FOR m = 0 to 3:
|    L_result = 0;
|==== FOR i = 0 to 12:
|      temp1 = x[m+(3*i)] >> 2;
|      L_temp = L_mult( temp1, temp1 );
|      L_result = L_add( L_temp, L_result );
|==== NEXT i:
|    IF ( L_result > EM) THEN
|                                            | Mc = m;
|                                            | EM = L_result;
|== NEXT m:
```

<u>Down-sampling by a factor 3 to get the selected xM[0..12] RPE sequence</u>.

```
|== FOR i = 0 to 12:
|    xM[i] = x[Mc +(3*i)];
|== NEXT i:
```

4.2.15. APCM quantization of the selected RPE sequence

Find the maximum absolute value xmax of xM[0..12].

```
xmax = 0;
|== FOR i = 0 to 12:
|     temp = abs( xM[i] ) ;
|     IF ( temp > xmax ) THEN xmax = temp;
|== NEXT i:
```

Quantizing and coding of xmax to get xmaxc.

```
exp = 0;
temp = xmax >> 9;
itest = 0;
|== FOR i = 0 to 5:
|     IF ( temp <= 0 ) THEN itest = 1;
|     temp = temp >> 1;
|     IF ( itest == 0 ) THEN exp = add( exp, 1 ) ;
|== NEXT i:


temp = add( exp, 5 ) ;
xmaxc = add( ( xmax >> temp ), ( exp << 3 ) ) ;
```

Quantizing and coding of the xM[0..12] RPE sequence to get the xMc[0..12].

This computation uses the fact that the decoded version of xmaxc can be calculated by using the exponent and the mantissa part of xmaxc (logarithmic table).

So, this method avoids any division and uses only a scaling of the RPE samples by a function of the exponent. A direct multiplication by the inverse of the mantissa (NRFAC[0..7] found in table 4.5) gives the 3 bit coded version xMc[0..12] of the RPE samples.

Compute exponent and mantissa of the decoded version of xmaxc.

```
exp = 0 ;

IF ( xmaxc > 15 ) THEN exp = sub( ( xmaxc >> 3 ), 1 ) ;

mant = sub( xmaxc , ( exp << 3 ) );
```

Normalize mantissa 0 <= mant <= 7.

```
IF ( mant == 0 ) THEN | exp = -4;
                      | mant = 15;

ELSE | itest = 0;

     |== FOR i = 0 to 2:

     |     IF ( mant > 7 ) THEN itest = 1;

     |     IF (itest == 0) THEN mant = add((mant << 1),1);

     |     IF ( itest == 0 ) THEN exp = sub( exp, 1 );

     |== NEXT i:

mant = sub( mant, 8 );
```

Direct computation of xMc[0..12] using table 4.5.

```
temp1= sub( 6, exp );   /normalization by the exponent/

temp2  =  NRFAC[mant];  /see table 4.5 (inverse mantissa)/

|== FOR i = 0 to 12:
|    temp = xM[i] << temp1;
|    temp = mult( temp , temp2 );
|    xMc[i] = add( ( temp >> 12 ), 4 );  /See note below/
|== NEXT i:
```

NOTE: This equation is used to make all the xMc[i] positive.

Keep in memory exp and mant for the following inverse APCM quantizer.

4.2.16. APCM inverse quantization

This part is for decoding the RPE sequence of coded xMc[0..12] samples to obtain the xMp[0..12] array. Table 4.6 is used to get the mantissa of xmaxc (FAC[0..7]).

```
temp1 = FAC[mant];  see 4.2-15 for mant

temp2= sub( 6, exp );  see 4.2-15 for exp

temp3= 1 << sub( temp2, 1 );
```

```
|== FOR i =0 to 12:
|    temp = sub( ( xMc[i] << 1 ), 7 );  /See note below/
|    temp = temp << 12;
|    temp = mult_r( temp1, temp );
|    temp = add( temp, temp3 );
|    xMp[i] = temp >> temp2;
|== NEXT i;
```

NOTE: This subtraction is used to restore the sign of xMc[i].

4.2.17. RPE grid positioning

This procedure computes the reconstructed long term residual signal ep[0..39] for the LTP analysis filter. The inputs are the Mc which is the grid position selection and the xMp[0..12] decoded RPE samples which are upsampled by a factor of 3 by inserting zero values.

```
|== FOR k = 0 to 39:
|    ep[k] = 0;
|== NEXT k:


|== FOR i = 0 to 12:
|    ep[Mc +(3*i)] = xMp[i];
|== NEXT i:
```

4.2.18. Update of the reconstructed short term residual signal dp[-120..-1]

This procedure adds the reconstructed long term residual signal ep[0..39] to the estimated signal dpp[0..39] from the long term analysis filter to compute the recontsructed short term residual signal dp[-40..-1]; also the reconstructed short term residual array dp[-120..-41] is updated.

```
|== FOR k = 0 to 79:
|     dp[-120+k] = dp[-80+k];
|== NEXT k:


|== FOR k = 0 to 39:
|     dp[-40+k] = add( ep[k], dpp[k] );
|== NEXT k:
```

Keep the array dp[-120..-1] in memory for the next sub-segment.

Initial value: dp[-120..-1]=0;

4.3. FIXED POINT IMPLEMENTATION OF THE RPE-LTP DECODER

Only the synthesis filter and the de-emphasis procedure are different from the procedures found in the RPE-LTP coder. Procedures 4.3.1 and 4.3.2 are executed for each sub-segment (four times per frame). Procedures 4.3.3, 4.3.4 and 4.3.5 are executed once per frame.

4.3.1. RPE decoding section

Procedures 4.2.15 (only the part to get mant and exp of xmaxc), 4.2.16 and 4.2.17 are used to obtain the reconstructed long term residual signal erp[0..39] signal from the received parameters for each sub-segment (ie Mcr, xmaxcr, xmcr[0..12]).

4.3.2. Long term synthesis filtering

This procedure uses the bcr and Ncr parameter to realize the long term synthesis filtering. The decoding of bcr needs the use of table 4.3b.

- Nr is the received and decoded LTP lag.
- An array drp[-120..39] is used in this procedure.

The elements for -120 to -1 of the array drp are kept in memory for the long term synthesis filter. For each sub-segment (40 samples), this procedure computes the drp[0..39] to be fed to the synthesis filter.

<u>Check the limits of Nr</u>.

```
        Nr = Ncr;

        IF ( Ncr < 40 ) THEN Nr = nrp;

        IF ( Ncr > 120 ) THEN Nr = nrp;

        nrp= Nr;
```

<u>Keep the nrp value for the next sub-segment</u>.

<u>Initial value</u>: nrp=40;

<u>Decoding of the LTP gain bcr</u>.

```
        brp = QLB[bcr]
```

<u>Computation of the reconstructed short term residual signal drp[0..39]</u>.

```
        |== FOR k = 0 to 39:
        |     drpp = mult_r( brp, drp[k-Nr] );
        |     drp[k] = add( erp[k], drpp );
        |== NEXT k:
```

<u>Update of the reconstructed short term residual signal drp[-1..-120]</u>.

```
        |== FOR k = 0 to 119:
        |     drp[-120+k] = drp[-80+k];
        |== NEXT k:
```

<u>Keep the array drp[-120..-1] for the next sub-segment</u>.

<u>Initial value</u>: drp[-120..-1]=0;

4.3.3. Computation of the decoded reflection coefficients

This procedure (which is executed once per frame) is the same as the one described in the CODER part.

For decoding of the received LARcr[1..8], see procedure 4.2.8.

For the interpolation of the decoded Log.-Area Ratios, see procedure 4.2.9.1 and for the computation of the reflection coefficients rrp[1..8], see procedure 4.2.9.2.

4.3.4. Short term synthesis filtering section

This procedure uses the drp[0..39] signal and produces the sr[0..159] signal which is the output of the short term synthesis filter. For ease of explanation, a temporary array wt[0..159] is used.

Initialisation of the array wt[0..159].

For the first sub-segment in a frame:

```
|== FOR k = 0 to 39:
|      wt[k] = drp[k];
|== NEXT k:
```

For the second sub-segment in a frame:

```
|== FOR k = 0 to 39:
|      wt[40+k] = drp[k];
|== NEXT k:
```

For the third sub-segment in a frame:

```
|== FOR k = 0 to 39:
|      wt[80+k] = drp[k];
|== NEXT k:
```

For the fourth sub-segment in a frame:

```
|== FOR k = 0 to 39:
|      wt[120+k] = drp[k];
|== NEXT k:
```

As the call of the short term synthesis filter procedure can be done in many ways (see the interpolation of the LAR coefficient), it is assumed that the computation begins with index k_start (for arrays wt[..] and sr[..]) and stops with index k_end (k_start and k_end are defined in 4.2.9.1). The procedure also needs to keep the array v[0..8] in memory between calls.

```
|== FOR k = k_start to k_end:
|    sri = wt[k];
|==== FOR i = 1 to 8:
|      sri = sub( sri, mult_r( rrp[9-i], v[8-i] ) );
|      v[9-i] = add( v[8-i], mult_r( rrp[9-i], sri ) );
|==== NEXT i:
|    sr[k] = sri;
|    v[0] = sri;
|== NEXT k:
```

Keep the array v[0..8] in memory for the next call.

Initial value: v[0..8]=0;

POSTPROCESSING

4.3.5. Deemphasis filtering

```
|== FOR k = 0 to 159:
|    temp = add( sr[k], mult_r( msr, 28180 ) );
|    msr = temp;
|    sro[k] = msr;
|== NEXT k:
```

Keep msr in memory for the next frame.

Initial value: msr=0;

4.3.6. Upscaling of the output signal

```
|== FOR k = 0 to 159:
|    srop[k] = add( sro[k], sro[k] );
|== NEXT k:
```

4.3.7. Truncation of the output variable

```
|== FOR k = 0 to 159:
|    srop[k] = srop[k] >> 3;
|    srop[k] = srop[k] << 3;
|== NEXT k:
```

The output format is the following:

S.v.v.v.v.v.v.v.v.v.v.v.v.0.0.0 (2's complement).

Where S is the sign bit, v a valid bit.

NOTE: When a linear to A-law compression is needed, then the sub-block COMPRESS of CCITT G721 recommendation must be used with inputs:

```
SR = srop[k] >> 3;
LAW = 1;
```

4.4. TABLES USED IN THE FIXED POINT IMPLEMENTATION OF THE RPE-LTP CODER AND DECODER

i	A[i]	B[i]	MIC[i]	MAC[i]
1	20480	0	-32	31
2	20480	0	-32	31
3	20480	2048	-16	15
4	20480	-2560	-16	15
5	13964	94	-8	7
6	15360	-1792	-8	7
7	8534	-341	-4	3
8	9036	-1144	-4	3

Table 4.1. Quantization of the Log.-Area Ratios

i	INVA[i]
1	13107
2	13107
3	13107
4	13107
5	19223
6	17476
7	31454
8	29708

Table 4.2. Tabulation of 1/A[1..8]

bc	DLB[bc]
0	6554
1	16384
2	26214
3	32767

Table 4.3a. Decision level of the LTP gain quantizer

bc	QLB[bc]
0	3277
1	11469
2	21299
3	32767

Table 4.3b. Quantization levels of the LTP gain quantizer

i	H[i]
0	-134
1	-374
2	0
3	2054
4	5741
5	8192
6	5741
7	2054
8	0
9	-374
10	-134

Table 4.4. Coefficients of the weighting filter

i	NRFAC[i]
0	29128
1	26215
2	23832
3	21846
4	20165
5	18725
6	17476
7	16384

Table 4.5. Normalized inverse mantissa used to compute xM/xmax

i	FAC[i]
0	18431
1	20479
2	22527
3	24575
4	26623
5	28671
6	30719
7	32767

Table 4.6. Normalized direct mantissa used to compute xM/xmax

5. DIGITAL TEST SEQUENCES

This chapter provides information on the digital test sequences that have been designed to help in the verification of implementations of the RPE-LTP codec. Copies of these sequences are available (see annex 3).

5.1. INPUT AND OUTPUT SIGNALS

Table 5.1 defines the input and output signals for the test sequences. The words defined in this table use 16 bits. The left or right justification is indicated in the table.

The codewords described in the table correspond to one frame of coder input or decoder output signal; i.e for 20 ms of input signal the 76 codewords are obtained at the output of the coder and 76 codewords provided at the input of the decoder will yield 20 ms of output signal in the decoder.

5.2. CONFIGURATION FOR THE APPLICATION OF THE TEST SEQUENCES

Two configurations are appropriate in order to test an implementation of the RPE-LTP coder. The first one is for testing the coder part of the RPE-LTP; it means that a sop[..] signal is provided at the input of the encoder that furnishes frames of coded parameters. This output has to be checked against some reference file. The other configuration is for testing the decoder part of the RPE-LTP; in this case frames of coded parameters (see table 5.1) are sent to the RPE-LTP decoder that furnishes the srop[..] signal. These samples have to be checked against a reference file.

5.2.1. Configuration 1 (encoder only)

A reset signal (RS) must be applied to the RPE-LTP encoder under test to set all internal variables to the exact states specified in section 4 of this recommendation prior to the start of an input test sequence in order to obtain the correct output values for this test. This test must be done in real time with a sampling rate of 8 kHz at the input of the encoder under test (see figure 5.1). All the necessary hardware and software should be installed by the user in order to capture in real time the output coded parameters of the RPE-LTP encoder and to compare them to the dedicated reference file.

```
                    | 8 kHz sampling rate
                    v
                 --------------
        RS --->| RPE-LTP       |  Coded parameters
               | Encoder       |-------------/--------> Comparison
Input:   ----/->| under test    |          2-7 bits
       13 bits   --------------
```

Figure 5.1. Configuration 1: RPE-LTP encoder under test

5.2.2. Configuration 2 (Decoder only)

Figure 5.2 shows a RPE-LTP decoder under test. In the same way as described in the coder part, a reset signal (RS) must be used before the processing of the first frame of coded parameters. The decoder must be tested for a continuous output with a sampling rate of 8 kHz. At the input of the decoder, the 76 parameters must be sent in a time interval of 20 ms.

```
                           | 8 kHz sampling rate
                           v
                       ------------
           RS ---->| RPE-LTP     |  Output signal: srop[k]
                   | Decoder     |-----------/--------> Comparison
Input: coded ---/-->| under test  |           13 bits
     parameters        ------------
     2-7 bits
```

Figure 5.2. Configuration 2: RPE-LTP decoder under test

5.3. TEST SEQUENCES

5.3.1. Test sequences for configuration 1

For configuration 1, four types of input test sequence are provided:

1. Sequence for testing the overflow controls in the encoder
2. Sequence for testing the LPC part of the encoder
3. Sequence for testing the LTP part of the encoder
4. Sequence for testing various critical parts of the algorithm

Sequence 1 uses a large number of saturated samples. The residual LPC signal reaches very high values, which has two effects on the processing:

- occurrence of a large number of overflows in addition/subtraction operations. Table 5.2 describes each overflow point and the number of occurrences for each.

- the excitation RPE samples have a large dynamic range and the 64 codewords of the sub-block maximum are each obtained at least once on output.

Sequence 2 focuses successively on each reflection coefficient calculated in the Schur recursion. Table 5.3 shows which frames deal with which reflection coefficient and its dynamic range. The Log.-Area codewords output by the coder cover the full range of their possible values except the 2nd LARc that does not reach the value 0 and 63 (min and max). The maximum value (63) is however obtained in sequence 4.

Sequence 3 tests the long term predictor part of the algorithm. It has been generated by exciting a sharply resonant filter with a periodic train of impulses; this produces a pitched signal. Each part corresponding to a given pitch is 128 ms (4 blocks of 256 words) long. The pitch periods have been randomly drawn in the range [2,15] ms and the random order is shown in table 5.4.

Sequence 4 accounts for various remaining non tested points of the algorithm where implementing errors may be suspected. Table 5.5 and 5.6 summarize the critical points that this sequence has been designed to check (ie where the three previous sequences were ineffective). Table 5.5 shows the list of tested points where errors can be detected. Each tested error is described and the frame number corresponding to the first occurrence of a divergence between the exact and the degraded algorithm is also indicated.

Table 5.6 illustrates three paths of the algorithm that are never explored during the processing of the three previous sequences; the table shows which condition leads to each path and the number of associated occurrences in sequence 4.

Notice finally one point where special care must be taken:

- A small degradation (ie +/- 1) of DLB[2] (the third decision level of the LTP gain quantizer (see table 4.3a) is unable to provide any noticeable effect on the output of the four sequences described above).

5.3.2. Test sequences for configuration 2

Five types of input test sequence are provided for this configuration. Four sequences obtained in configuration 1 at the output of the encoder (coded parameters) are used as input for the decoder under test in configuration 2.

Table 5.7 gives the list of tested overflow points and their occurrence on sequence 1 for this configuration.

Sequence 5 is provided to scan all possible codes for each parameter. This sequence is an artificial sequence and does not correspond to any encoder output. The codewords have been randomly generated and cover the entire range of codewords values. Moreover, the delay value Nr belonging to [40,120] in an error-free transmission condition, takes in this sequence its value in [0,127]. In this case the decoder behaviour on non-allowed values of Nr will be tested.

Name	Description	Justification
	ENCODER INPUT	
sop[k]	13 bits: encoder input signal.	left
	OUTPUT PARAMETERS	
LARc[1]	6 bits : 1st Log.-Area Ratio	right
LARc[2]	6 bits : 2nd Log.-Area Ratio	right
LARc[3]	5 bits : 3rd Log.-Area Ratio	right
LARc[4]	5 bits : 4th Log.-Area Ratio	right
LARc[5]	4 bits : 5th Log.-Area Ratio	right
LARc[6]	4 bits : 6th Log.-Area Ratio	right
LARc[7]	3 bits : 7th Log.-Area Ratio	right
LARc[8]	3 bits : 8th Log.-Area Ratio	right
	Sub-frame no 1	
Nc	7 bits : LTP lag	right
bc	2 bits : LTP gain	right
Mc	2 bits : RPE grid position	right
xmaxc	6 bits : Block amplitude	right
xMc[0..12]	3 bits : RPE pulses index 0 to 12	right
	Sub-frame no 2	
Nc	7 bits : LTP lag	right
bc	2 bits : LTP gain	right
Mc	2 bits : RPE grid position	right
xmaxc	6 bits : Block amplitude	right
xMc[0..12]	3 bits : RPE pulses index 0 to 12	right
	Sub-frame no 3	
Nc	7 bits : LTP lag	right
bc	2 bits : LTP gain	right
Mc	2 bits : RPE grid position	right
xmaxc	6 bits : Block amplitude	right
xMc[0..12]	3 bits : RPE pulses index 0 to 12	right
	Sub-frame no 4	
Nc	7 bits : LTP lag	right
bc	2 bits : LTP gain	right
Mc	2 bits : RPE grid position	right
xmaxc	6 bits : Block amplitude	right
xMc[0..12]	3 bits : RPE pulses index 0 to 12	right

Table 5.1a. Signals used in digital test sequences

DECODER OUTPUT		
srop[k]	13 bits: decoder output signal. The 3 LSB's of the 16 bits are equal to 0.	left

Table 5.1b. Signals used in digital test sequences

Overflow point	No of occurrences
Short term analysis filter (4.2.10)	
1st add	1059
2nd add	134
LTP parameters computation (4.2.11)	
Abs(d[k])	5
Long term analysis filter (4.2.12)	
sub	11
Weighting filter (4.2.13)	
scaling the result (both x2 and x4)	302
APCM quantizer (4.2.15)	
Find max abs of xm: Abs	49
Update of Array dp of the long term analysis filter (4.2.18)	
add	126

Table 5.2. List of tested overflow points for sequence 1 (coder part)

Reflection Coeff.	Frames	Dynamic range
1	1-135	-32564,32558
2	136-311	-32356,32242
3	316-423	-32157,32744
4	424-524	-31594,31960
5	525-633	-31697,31735
6	634-738	-30055,31575
7	739-839	-29090,31386
8	840-944	-31052,31208

Table 5.3.

Table 5.3 gives the position of the frames dedicated to the study of each reflection coefficient and dynamic range of the coefficient for sequence 2 in configuration 2.

86	56	68	120	52	93	20	66	82	115	114	60	42	45	17	64	16
88	83	63	90	73	23	77	100	33	29	106	35	67	57	103	116	30
71	69	81	47	32	97	65	62	111	49	109	25	96	50	54	91	85
99	70	76	46	26	34	104	108	107	22	119	48	58	37	72	110	27
24	36	87	51	59	38	21	44	113	39	61	53	18	40	94	105	55
112	75	98	118	41	80	31	74	28	84	89	79	43	101	95	19	78
117	92	102														

Table 5.4. Pitch periods of sequence 3 (configuration 1)

Test point	Error checked : incorrect statement / correct statement	No of the 1st frame with error
Autocorrelation function (4.2.4)	k=0 to 158 / k=0 to 159	27
Computation of the reflect. coefficients (4.2.5)	if(P[0] <= abs(P[1])) / if(P[0] < abs(P[1]))	514
Quantization and coding of the LARs (4.2.7)	A[4] + 1 / A[4]	21
	A[5] - 1 / A[5]	35
	A[5] + 1 / A[5]	430
	A[6] - 1 / A[6]	427
	A[8] - 1 / A[8]	8
	MAC[2] - 1 / MAC[2]	24
	MAC[2] + 1 / MAC[2]	516
Comput. of the rp from the interp. LARp (4.2.9)	11058 / 11059	19
	20069 / 20070	25
Calc. of the LTP parameters		
->Search of the opt scaling (4.2.11)	k= 0 to 38 / k= 0 to 39	32
->Coding of the LTP gain (4.2.11)	mult_r / mult	373
	DLB[0] + 1 / DLB[0]	511
	DLB[1] + 1 / DLB[1]	373
ADPCM inverse quantizer (4.2.16)	FAC[2] + 1 / FAC[2]	422
	FAC[3] - 1 / FAC[3]	179
	FAC[4] + 1 / FAC[4]	74
	FAC[5] - 1 / FAC[5]	439
	FAC[5] + 1 / FAC[5]	74
	FAC[6] - 1 / FAC[6]	479
	FAC[6] + 1 / FAC[6]	330
	FAC[7] - 1 / FAC[7]	139

Table 5.5. Errors specially detected by sequence 4/Config 1

Test point	Number of occurrences
Autocorrelation function (4.2.4) condition smax == 0	8
Computation of the reflection coefficients :	
-> condition L_ACF[0] == 0 (4.2.5)	8
-> condition P[0] < abs(P[1]) (4.2.5)	4

Table 5.6. Paths specially explored by sequence 4/Config 1

Overflow Point	Nb of occurrences Sequence 1	Sequence 2	Sequence 3
Long term synthesis filter (4.3.2) : add	126	0	0
Short term synthesis filter (4.3.4) :			
1st add:	4499	0	0
2nd add:	405	1	0
De-emphasize filter (4.3.5) : add	89	0	0
Scaling of the output signal (4.3.6) : add	16691	339	19

Table 5.7. List of tested overflows points for sequence 1 (decoder part)

ANNEX 1. CODEC PERFORMANCE

A1.1. INTRODUCTION

The purpose of this annex is to give a broad outline of the performance of the RPE-LTP codec with other parts of the digital network. Some general guidance is also offered on non-voice services.

A1.2. SPEECH PERFORMANCE

Planning rules for digital processes are defined in terms of quantizing distortion units (qdu) which can be realized from the following formula (reference 1) using the assumption that the formula accuracy represents the determination of qdus from QN measurements:

$$QN = 37 - 15 \log_{10}(n) \quad \text{,where n is the qdu} \qquad \text{(A1.1)}$$

By definition 1 qdu is the quantization distortion arising from one commercial PCM codec.

NOTE: The subjective testing methodology to determine QN for the RPE-LTP codec was consistent with current CCITT methods (reference 2).

A1.2.1. Single encoding

Under error-free transmission conditions the perceived quality of the RPE-LTP codec (see fig A1.1) is lower than both codecs conforming to recommendations CCITT G.711 and CCITT G.721.

Table A1.1 indicates the relative performance of the codec and can be compared with codecs conforming to recommendations CCITT G.711 and CCITT G.721.

The performance of the RPE-LTP codec has been found to be substantially unaffected down to a carrier to interference (C/I) ratio of 10 dB, but may be considered to have acceptable performance down to 7 dB. Smaller C/I ratios produce unacceptable degradation of speech performance and should be avoided.

NOTE 1: It should be noted that there are doubts as to whether the simulations which generated the error pattern properly represent real operating conditions. The C/I values quoted should therefore only be considered as parameters of this simulation. They may not correspond to real radio interference conditions. Results from early GSM validation hardware show that the C/I values which give the performance quoted may be several dBs higher. Some error statistics of the simulations are shown in table A1.2.

NOTE 2: The real condition C/I = 10 dB is believed to correspond to about 90 % coverage.

Codec	QN (dB)	qdu
G.711 (64 kbit/s, A-law PCM)	37	1
G.721 (32 kbit/s, ADPCM)	29	3.5 (*)
RPE-LTP	23-25	7-8 (*)

(*) Commercial A-law PCM input and output circuitry included.

NOTE: The qdu value for the RPE-LTP codec is a conservative estimate. At present there are no specific CCITT rules for determining qdus for encoding below 32 kbit/s.

Table A1.1. Relative levels of speech performance under error-free conditions

Simulated C/I ratio:	10 dB	7 dB	4 dB
Total number of errors in class I (182 bits protected by a 1/2 rate code)	0.016%	0.61%	4.1%
Total number of errors in class II (78 bits unprotected)	4.5%	8.3%	13.0%
Number of "frame erasure" indications by CRC	1	15	95
Number of "frame erasures" not detected by CRC	1	14	76

NOTE: The total number of frames was 750. CRC means Cyclic Redundancy Check.

Table A1.2. Bit error statistics for C/I test conditions

A1.2.2. Speech performance when interconnected with coding systems on an analogue basis

A1.2.2.1. Performance with 32 kbit/s ADPCM (G.721)

The speech performance of the RPE-LTP codec when interconnected with encoding at 32 kbit/s (see fig A1.3 and A1.4) decreases in accordance with the formula in section A1.2, and appears to obey the law of additivity when qdus have been determined for the individual codecs.

A1.2.2.2. Performance with another RPE-LTP codec

The speech performance of the RPE-LTP codec when interconnected with another codec of the same type (see fig A1.2) is lower than that of A1.2.2.1. It again appears to obey the law of additivity when qdus have been determined for the individual codecs.

A1.2.2.3. Performance with encoding other than RPE-LTP and 32 kbit/s ADPCM (G.721)

No information is available on this point, so great care must be exercised when interconnection is made to codecs with encoding different from that of A1.2.2.1 and A1.2.2.2.

A1.3. NON-SPEECH PERFORMANCE

It should be noted that the RPE-LTP speech codec is an adaptive system which has been optimised for speech inputs. Great care must be taken when making measurements with non-speech signals beause the normal assumptions of time invariance and linearity cannot be made.

A.1.3.1. Performance with single sine waves

Detailed experiments have shown that the RPE-LTP codec will pass sine waves with segmental signal to noise ratios generally in excess of 20 dB in the frequency range of 100 - 2000 Hz. However, in some cases reproduction above 2000 Hz is not as good.

It should be noted that sine waves above 1300 Hz may be reproduced with significant fluctuations in amplitude and frequency due to the adaptive sub-sampling technique employed. This results in irregularities in the measured frequency response.

A typical frequency response measured with A-law PCM input circuitry is shown in fig A1.5. If 13 bit linear PCM input circuitry is used, the irregularity is less.

A1.3.2. Performance with DTMF tones

It has been shown that the RPE-LTP codec transfers DTMF signals of 80 ms duration. However, questions like minimum allowable signal duration, pause duration and the behaviour in the presence of transmission errors have not been investigated.

A1.3.3. Performance with information tones

Experiments have shown that network originated signalling tones, conforming to recommendation CCITT Q.35 (reference 3), are easily recognizable when passed through the RPE-LTP codec.

A1.3.4. Performance with voice-band data

Tests have shown that voice-band data transmission does not work satisfactorily with 1200 bit/s modems according to recommendation CCITT V.23. Voice-band data according to recommendation CCITT V.21 (300 bit/s) will not be subject to any significant degradation.

This behaviour has been tested for one RPE-LTP link (encoder-decoder). The effect of transmission errors has not been tested.

A1.4. DELAY

The theoretical minimum delay of the RPE-LTP codec is 20 ms. However, practical realizations may have an additional processing time in the order of 3 - 8 ms.

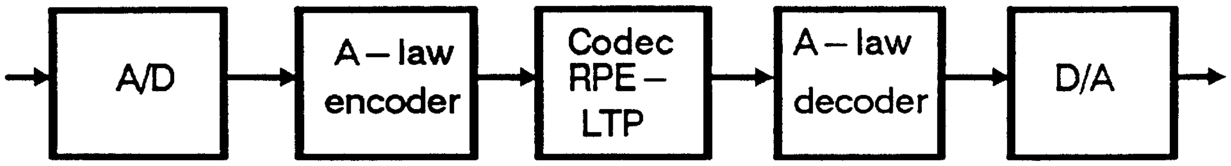


Fig A.1.1. One-transcoding scheme

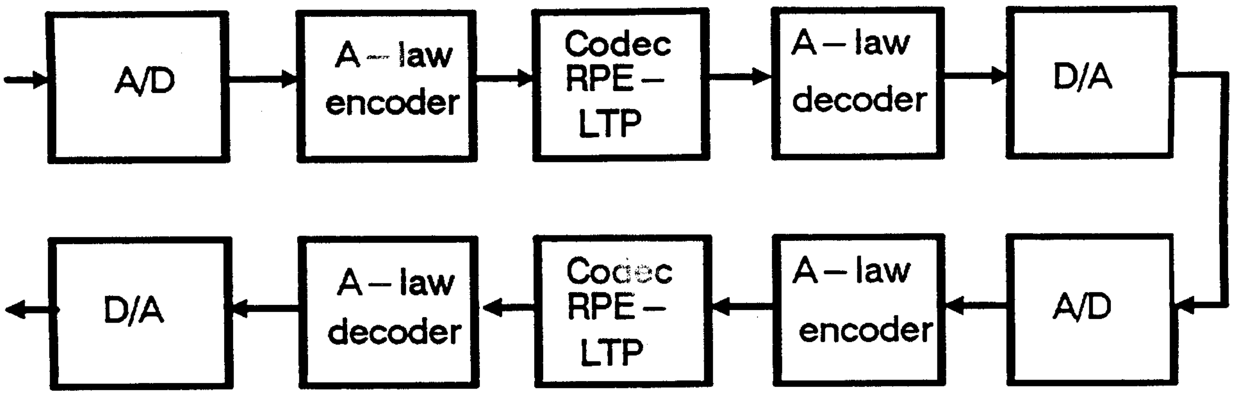


Fig A1.2. Two-transcodings scheme

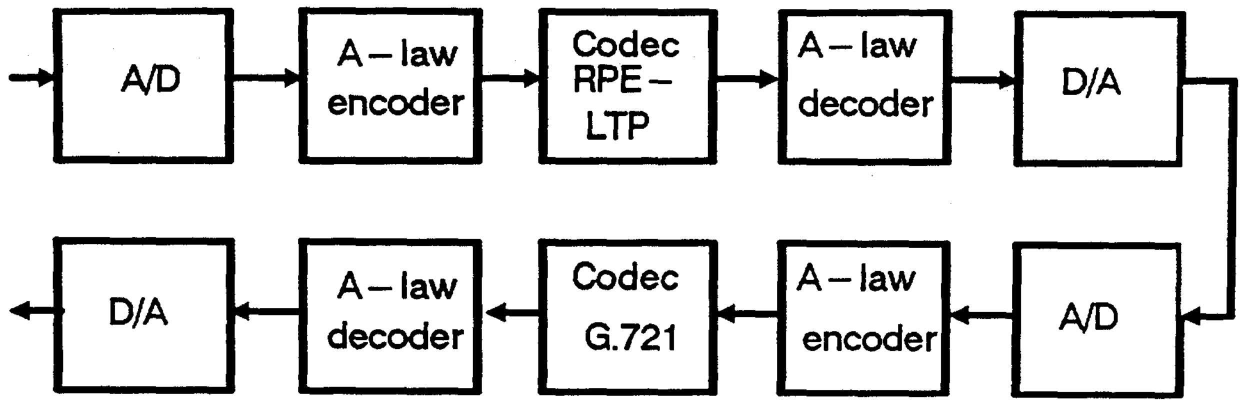


Fig A1.3. Mixed transcodings – scheme 1

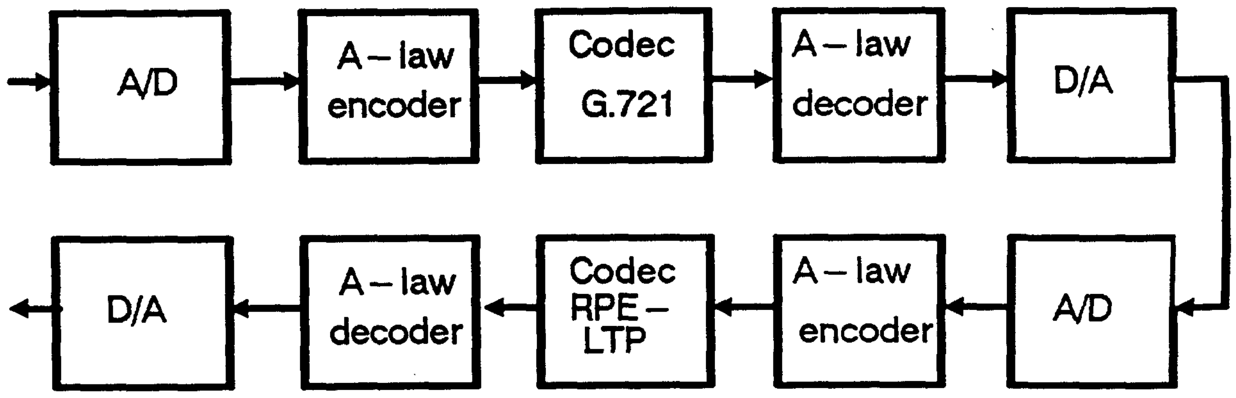


Fig A1.4. Mixed transcodings – scheme 2

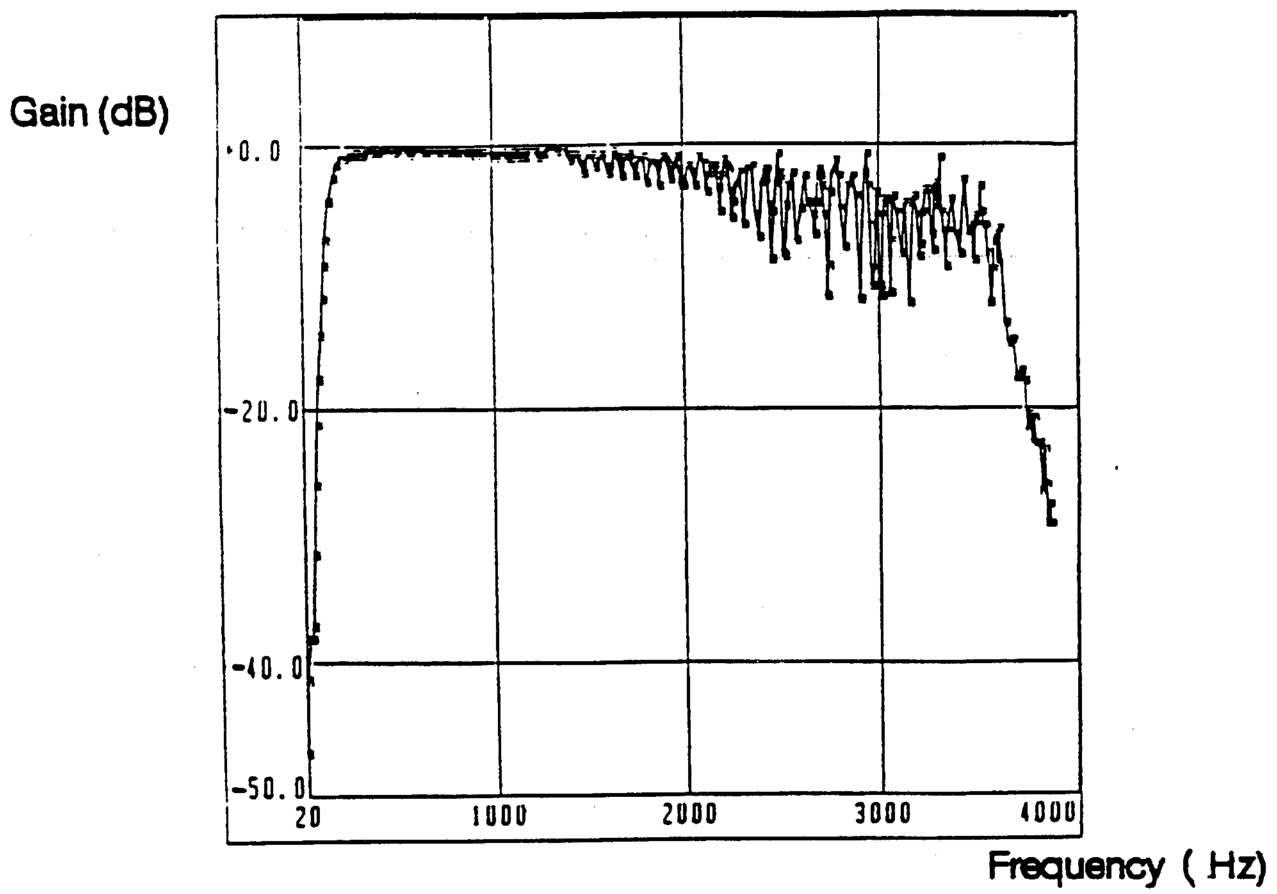


Fig A1.5. Frequency response for RPE-LTP codec (with commercial A-law PCM input and output circuitry)

A1.5. REFERENCES

1. CCITT: "Subjective performance assessment of digital processes using the Modulated Noise Reference Unit (MNRU)", Annex C, Supplement no 14, Red book, volume V, 1985.

2. CCITT: "Subjective performance assessment of digital processes using the Modulated Noise Reference Unit (MNRU)", Annex A, Supplement no 14, Red book, volume V, 1985.

3. CCITT: "Technical characteristics of tones for the telephone service", recommendation Q.35, Red book, volume VI.1, 1985.

ANNEX 2. SUBJECTIVE RELEVANCE OF THE SPEECH CODER OUTPUT BITS

Since no valid objective quality criterion for speech signals is available, the only way to build up such a relevance table is to perform listening tests. The procedure described below was used to obtain the relevance classification given in table A2.1 of the recommendation.

To classify a single bit, say bit i of parameter k, a short speech signal (2 sec) was encoded, then this bit was inverted in each frame (the other bits were left unchanged) and the resulting bit stream was fed into the speech decoder. The listeners had to compare the quality of the signal with the quality of six reference signals with different levels of distortion. Repeating this procedure for all bits would result in a subdivision of the 260 bits into six relevance classes.

It can be observed that many of the bits have the same physical meaning and it can be expected that bits with the same meaning have the same relevance (eg the MSB's of the RPE samples). Relying on this assumption, only one of the equivalent parameters was considered. Since there are 13 parameters with different physical meaning with 56 bits in total, the number of tests is reduced from 260 to 56.

The reference signals were the same speech signal distorted by inverting one of the six bits of LAR coefficient number one. This resulted in an adequate quantization of distortion levels ranging from "not intelligible" (MSB inverted) to "negligible distortion" (LSB inverted).

The test was carried out using three listeners and one female speaker. Since the three listeners came to rather similar results, no more listeners were considered to be required. Averaging the three outcomes led to the relevance table given in table A2.1, where the order of all bits between two successive bits of the first parameter (LAR 1) are arbitrarily chosen.

Importance class	Parameter name	Parameter number	Bit number
1	Log.area ratio 1	1	b6
	Block amplitude	12,29,46,63	b53,b109,b165,b221
2	Log.area ratio 1	1	b5
	Log.area ratio 2	2	b12
	Log.area ratio 3	3	b17
3	Log.area ratio 1	1	b4
	Log.area ratio 2	2	b11
	Log.area ratio 3	3	b16
	Log.area ratio 4	4	b22
	LTP lag	9,26,43,60	b43,b99,b155,b211
	Block amplitude	12,29,46,63	b52,b108,b164,b220
	Log.area ratio 2,5,6	2,5,6	b10,b26,b30
	LTP lag	9,26,43,60	b42,b98,b154,b210
	LTP lag	9,26,43,60	b41,b97,b153,b209
	LTP lag	9,26,43,60	b40,b96,b152,b208
	LTP lag	9,26,43,60	b39,b95,b151,b207
4	Block amplitude	12,29,46,63	b51,b107,b163,b219
	Log.area ratio 1	1	b3
	Log.area ratio 4	4	b21
	Log.area ratio 7	7	b33
	LTP lag	9,26,43,60	b38,b94,b150,b206
	Log.area ratio 5,6	5,6	b25,b29
	LTP gain	10,27,44,61	b45,b101,b157,b213
	LTP lag	9,26,43,60	b37,b93,b149,b205
	Grid position	11,28,45,62	b47,b103,b159,b215

Table A2.1a. Subjective importance of encoded bits (the parameter and bit numbers refer to table 1.1)

Importance class	Parameter name	Parameter number	Bit number
	Log.area ratio 1	1	b2
	Log.area ratio 2,3,8,4	2,3,8,4	b9,b15,b36,b20
	Log.area ratio 5,7	5,7	b24,b32
	LTP gain	10,27,44,61	b44,b100,b156,b212
	Block amplitude	12,29,46,63	b50,b106,b162,b218
	RPE pulses	13..25	b56,b59,..,b92
	RPE pulses	30..42	b112,b115,..,b148
	RPE pulses	47..59	b168,b171,..,b204
5	RPE pulses	64..76	b224,b227,..,b260
	Grid position	11,28,45,62	b46,b102,b158,b214
	Block amplitude	12,29,46,63	b49,b105,b161,b217
	RPE pulses	13..25	b55,b58,..,b91
	RPE pulses	30..42	b111,b114,..,b147
	RPE pulses	47..59	b167,b170,..,b203
	RPE pulses	64..67	b223,b226,b229,b232
	RPE pulses	68..76	b235,b238,..,b259
	Log.area ratio 1	1	b1
	Log.area ratio 2,3,6	2,3,6	b8,b14,b28
	Log.area ratio 7	7	b31
	Log.area ratio 8	8	b35
	Log.area ratio 8,3	8,3	b34,b13
	Log.area ratio 4	4	b19
6	Log.area ratio 4,5	4,5	b18,b23
	Block amplitude	12,29,46,63	b48,b104,b160,b216
	RPE pulses	13..25	b54,b57,..,b90
	RPE pulses	30..42	b110,b113,..,b146
	RPE pulses	47..59	b166,b169,..,b202
	RPE pulses	64..76	b222,b225,..,b258
	Log.area ratio 2,6	2,6	b7,b27

Table A2.1b. Subjective importance of encoded bits (the parameter and bit numbers refer to table 1.1)

ANNEX 3. FORMAT FOR TEST SEQUENCE DISTRIBUTION

A3.1. TYPE OF FILES PROVIDED

Three types of files are provided:

- Files for input of the encoder	: SEQxx.INP
- Files for input of decoder or comparison with encoder output:	: SEQxx.COD
- Files for comparison with the decoder output	: SEQxx.OUT

Two diskettes are provided containing all the digital test sequences.

The 1st flexible disk contains the SEQ01.INP, SEQ01.COD, SEQ01.OUT, SEQ02.INP, SEQ02.COD, SEQ02.OUT files. The 2nd flexible disk contains the SEQ03.INP, SEQ03.COD, SEQ03.OUT, SEQ04.INP, SEQ04.COD, SEQ04.OUT, SEQ05.COD, SEQ05.OUT files. Table A3.1 gives the contents of the two distribution flexible disks and also the size in bytes and the number of frames for each test sequence file.

A3.2. FILE FORMAT DESCRIPTION

All the files are written in binary using 16 bit words. This means that input samples (sop[k]), output samples (srop[k]) and coded parameters use 2 bytes each. Hence the sizes of the files are directly related to the number of processed frames.

For files with .INP or .OUT extension type:

```
    Size (in bytes) = No of frames * 160 * 2 ;
```

For files with .COD extension type:

```
    Size (in bytes) = No of frames * 76 * 2;
```

Table A3.1 shows the size of all the files written in direct binary format.

The diskette is formatted according to the high capacity (1.2 Mb) specifications for MS/DOS PC-AT compatible computers.

DISK No	FILE	FRAMES	SIZE (bytes)
1	SEQ01.INP SEQ01.COD SEQ01.OUT	584	186880 88768 186880
1	SEQ02.INP SEQ02.COD SEQ02.OUT	947	303040 143944 303040
2	SEQ03.INP SEQ03.COD SEQ03.OUT	673	215360 102296 215360
2	SEQ04.INP SEQ04.COD SEQ04.OUT	520	166400 79040 166400
2	SEQ05.COD SEQ05.OUT	64	9728 20480

Table A3.1. Contents of diskettes and size of files