ETSI TS 126 204 V13.0.0 (2016-01)

TECHNICAL SPECIFICATION

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Speech codec speech processing functions;
Adaptive Multi-Rate - Wideband (AMR-WB) speech codec;
ANSI-C code
(3GPP TS 26.204 version 13.0.0 Release 13)

Reference
RTS/TSGS-0426204vd00

Keywords
GSM,LTE,UMTS

ETSI

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

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

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

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

Intellectual Property Rights

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

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

Foreword

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

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

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

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

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

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

Version x.y.z

where:

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

1 Scope

The present document contains an electronic copy of the ANSI-C code for the Floating-point Adaptive Multi-Rate Wideband codec. This floating-point codec specification is mainly targeted to be used in multimedia applications or in packet-based applications. The bit-exact fixed-point ANSI-C code in 3GPP TS 26.173 remains the preferred implementation for all applications, but the floating-point codec may be used instead of the fixed-point codec when the implementation platform is better suited for a floating-point implementation. It has been verified that the fixed-point and floating-point codecs interoperate with each other without any artifacts.

The floating-point ANSI-C code in the present document is the only standard conforming non-bit-exact implementation of the Adaptive Multi-Rate Wideband speech transcoder (3GPP TS 26.190 [2]), Voice Activity Detection (3GPP TS 26.194 [6]), comfort noise generation (3GPP TS 26.192 [4]), and source controlled rate operation (3GPP TS 26.193 [5]). The floating-point code also contains example solutions for substituting and muting of lost frames (3GPP TS 26.191 [3]).

The fixed-point specification in 26.173 shall remain the only allowed implementation for the 3G AMR-WB speech service and the use of the floating-point codec is strictly limited to other services.

The floating-point encoder in the present document is a non-bit-exact implementation of the fixed-point encoder producing quality indistinguishable from that of the fixed-point encoder. The decoder in the present document is functionally a bit-exact implementation of the fixed-point decoder, but the code has been optimized for speed and the standard fixed-point libraries are not used as such.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

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

[1] 3GPP TS 26.174: "AMR speech codec, wideband; Test sequences".

[2] 3GPP TS 26.190: "Mandatory Speech Codec speech processing functions AMR Wideband speech codec; Transcoding functions".

[3] 3GPP TS 26.191: "AMR speech codec, wideband; Error concealment of lost frames".

[4] 3GPP TS 26.192: "Mandatory Speech Codec speech processing functions AMR Wideband Speech Codec; Comfort noise aspects".

[5] 3GPP TS 26.193: "AMR speech codec, wideband; Source controlled rate operation".

[6] 3GPP TS 26.194: "Mandatory Speech Codec speech processing functions AMR Wideband speech codec; Voice Activity Detector (VAD)".

[7] RFC 3267 'A Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs, June 2002.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 26.190 [2], TS 26.191 [3], TS 26.192 [4], TS 26.193 [5] and TS 26.194 [6].

3.2 Abbreviations

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

AMR-WB	Adaptive Multi-Rate WideBand
ANSI	American National Standards Institute
GSM	Global System for Mobile communications
I/O	Input/Output
RAM	Random Access Memory
ROM	Read Only Memory

4 C code structure

This clause gives an overview of the structure of the bit-exact C code and provides an overview of the contents and organization of the C code attached to the present document.

The C code has been verified on the following systems:

- IBM PC/AT compatible computers with Windows NT40 and Microsoft Visual C++ v.6.0 compiler.
- IBM PC/AT compatible computers with Windows NT40 and Intel C/C++ v.4.0 compiler.

ANSI-C was selected as the programming language because portability was desirable.

4.1 Contents of the C source code

The C code distribution has all files in the root level.

The distributed files with suffix "c" contain the source code and the files with suffix "h" are the header files. The ROM data is contained in "rom" files with suffix "c".

Makefiles are provided for the platforms in which the C code has been verified (listed above). Once the software is installed, this directory will have a compiled version of encoder and decoder and all the object files.

4.2 Program execution

The Adaptive Multi-Rate Wideband codec is implemented in two programs:

- (*encoder*) speech encoder;
- (*decoder*) speech decoder.

The programs should be called like:

- encoder [encoder options] <speech input file> <parameter file>;
- decoder <parameter file> <speech output file>.

The speech files contain 16-bit linear encoded PCM speech samples and the parameter files contain encoded speech data and some additional flags.

The encoder and decoder options will be explained by running the applications without input arguments. See the file readme.txt for more information on how to run the *encoder* and *decoder* programs.

4.3 Code hierarchy

Tables 1 and 2 are call graphs that show the functions used in the speech codec, including the functions of VAD, DTX, and comfort noise generation.

Each column represents a call level and each cell a function. The functions contain calls to the functions in rightwards neighbouring cells. The time order in the call graphs is from the top downwards as the processing of a frame advances. All standard C functions: memcpy(), fwrite(), etc. have been omitted. The initialization of the static RAM (i.e. calling the _init functions) is also omitted.

Table 1: Speech encoder call structure

E_MAIN_encode	E_UTIL_decim_12k8	E_UTIL_down_samp	E_UTIL_interpol	
	E_UTIL_decim_12k8			
	E_UTIL_hp50_12k8			
	E_UTIL_hp50_12k8			
	E_UTIL_f_preemph			
	E_DTX_vad	E_DTX_filter_bank	E_DTX_filter5	
			E_DTX_filter3	
			E_DTX_level_calculation	
		E_DTX_decision	E_DTX_noise_estimate_update	E_DTX_update_cntrl
			E_DTX_hangover_addition	
		E_DTX_speech_estimate		
	E_DTX_tx_handler			
	E_DTX_reset	E_LPC_isf_init		
	E_MAIN_parm_store			
	E_UTIL_autocorr			
	E_LPC_lag_wind			
	E_LPC_lev_dur			
	E_LPC_a_isp_conversion	E_LPC_chebyshev		
	E_LPC_f_int_isp_find	E_LPC_f_isp_a_conversion	E_LPC_f_isp_pol_get	
	E_LPC_isp_isf_conversion			
	E_GAIN_clip_isf_test			
	E_LPC_a_weight			
	E_UTIL_residu			
	E_UTIL_deemph			
	E_GAIN_lp_decim2			
	E_GAIN_open_loop_search			
	E_GAIN_olag_median	E_GAIN_sort		
	E_DTX_pitch_tone_detection			
	E_GAIN_open_loop_search			
	E_GAIN_olag_median			
	E_DTX_pitch_tone_detection			
	E_UTIL_residu			
	E_DTX_buffer			
	E_DTX_exe	E_DTX_frame_indices_find		
		E_DTX_isf_history_aver		
		E_DTX_isf_q	E_LPC_isf_sub_vq	
			E_LPC_isf_noise_d	E_LPC_f_isf_reorder
		E_DTX_dithering_control		
		E_UTIL_random		
	E_MAIN_reset	E_GAIN_clip_init		
		E_DTX_reset		
		E_DTX_vad_reset		
	E_LPC_isf_2s3s_quantise	E_LPC_stage1_isf_vq		
		E_LPC_isf_sub_vq		
		E_LPC_stage1_isf_vq		
		E_LPC_isf_sub_vq		
		E_LPC_isf_2s3s_decode	E_LPC_isf_reorder	
	E_LPC_isf_2s5s_quantise	E_LPC_stage1_isf_vq		
		E_LPC_isf_sub_vq		
		E_LPC_isf_2s5s_decode	E_LPC_isf_reorder	
	E_LPC_isf_isp_conversion			
	E_LPC_int_isp_find	E_LPC_isp_a_conversion	E_LPC_isp_pol_get	E_UTIL_l_extract
				E_UTIL_mpy_32_16
			E_UTIL_l_extract	
			E_UTIL_mpy_32_16	
	E_UTIL_residu			
	E_DTX_buffer			
	E_UTIL_residu			
	E_UTIL_synthesis			
	E_LPC_a_weight			
	E_UTIL_residu			
	E_UTIL_deemph			
	E_UTIL_f_preemph			

E_LPC_a_weight			
E_UTIL_synthesis			
E_UTIL_residu			
E_LPC_a_weight			
E_UTIL_synthesis			
E_UTIL_deemph			
E_GAIN_closed_loop_search	E_GAIN_norm_corr	E_UTIL_f_convolve	
	E_GAIN_norm_corr_interpolate		
E_GAIN_clip_test			
E_GAIN_adaptive_codebook_excitation			
E_UTIL_convolve			
E_ACELP_xy1_corr			
E_ACELP_codebook_target_update			
E_UTIL_convolve			
E_ACELP_xy1_corr			
E_ACELP_codebook_target_update			
E_UTIL_f_preemph			
E_GAIN_f_pitch_sharpening			
E_ACELP_xh_corr			
E_ACELP_2t			
E_ACELP_4t	E_ACELP_h_vec_corr1		
	E_ACELP_h_vec_corr2		
	E_ACELP_2pulse_search		
	E_ACELP_quant_1p_N1		
	E_ACELP_quant_2p_2N1		
	E_ACELP_quant_3p_3N1	E_ACELP_quant_2p_2N1	
		E_ACELP_quant_1p_N1	
	E_ACELP_quant_4p_4N	E_ACELP_quant_4p_4N1	E_ACELP_quant_2p_2N1
		E_ACELP_quant_1p_N1	
		E_ACELP_quant_3p_3N1	
		E_ACELP_quant_2p_2N1	
		E_ACELP_quant_3p_3N1	
	E_ACELP_quant_5p_5N	E_ACELP_quant_3p_3N1	
		E_ACELP_quant_2p_2N1	
	E_ACELP_quant_6p_6N_2	E_ACELP_quant_5p_5N	
		E_ACELP_quant_1p_N1	
		E_ACELP_quant_4p_4N	
		E_ACELP_quant_2p_2N1	
		E_ACELP_quant_3p_3N1	
E_UTIL_preemph			
E_GAIN_pitch_sharpening			
E_ACELP_xy2_corr			
E_ACELP_gains_quantise	E_UTIL_dot_product12	E_UTIL_saturate_31	
		E_UTIL_norm_l	
	E_UTIL_normalized_inverse_sqrt		
	E_UTIL_l_extract		
	E_UTIL_saturate		
	E_UTIL_mpy_32_16		
	E_UTIL_log2_32	E_UTIL_norm_l	
		E_UTIL_normalized_log2	
E_UTIL_signal_up_scale			
E_UTIL_signal_down_scale			
E_GAIN_clip_pit_test			
E_UTIL_signal_down_scale			
E_GAIN_voice_factor	E_UTIL_dot_product12		
	E_UTIL_norm_l		
	E_UTIL_norm_s		
E_UTIL_norm_s			
E_UTIL_synthesis			
E_UTIL_enc_synthesis	E_UTIL_synthesis		
	E_UTIL_deemph		
	E_UTIL_hp50_12k8		

		E_UTIL_random
		E_UTIL_hp400_12k8
		E_LPC_a_weight
		E_UTIL_synthesis
		E_UTIL_bp_6k_7k
		E_UTIL_bp_6k_7k

Table 2: Speech decoder call structure

D_MAIN_decode	D_DTX_rx_handler	D_LPC_isf_noise_d	D_LPC_isf_reorder	
	D_DTX_exe	D_DTX_cn_dithering	D_UTIL_random	
		D_UTIL_pow2		
		D_UTIL_norm_l		
		D_UTIL_random		
		D_UTIL_dot_product12	D_UTIL_norm_l	
		D_UTIL_normalized_inverse_sqrt		
	D_LPC_isf_isp_conversion			
	D_LPC_isp_a_conversion	D_LPC_isp_pol_get	D_UTIL_l_extract	
			D_UTIL_mpy_32_16	
		D_UTIL_l_extract		
		D_UTIL_mpy_32_16		
	D_UTIL_dec_synthesis	D_UTIL_synthesis_32		
		D_UTIL_deemph_32	D_UTIL_saturate	
		D_UTIL_hp50_12k8	D_UTIL_l_extract	
		D_UTIL_oversamp_16k	D_UTIL_up_samp	D_UTIL_interpol
		D_UTIL_random		
		D_UTIL_signal_down_scale		
		D_UTIL_dot_product12		
		D_UTIL_normalized_inverse_sqrt		
		D_UTIL_hp400_12k8	D_UTIL_l_extract	
		D_UTIL_norm_l		
		D_LPC_isf_extrapolation	D_UTIL_norm_s	
			D_UTIL_l_extract	
			D_UTIL_mpy_32	
			D_LPC_isf_isp_conversion	
		D_LPC_isp_a_conversion		
		D_LPC_a_weight		
		D_UTIL_synthesis		
		D_LPC_a_weight		
		D_UTIL_synthesis		
		D_UTIL_bp_6k_7k		
		D_UTIL_hp_7k		
	D_MAIN_reset	D_GAIN_init		
		D_GAIN_lag_concealment_init		
		D_DTX_reset		
	D_LPC_isf_2s3s_decode	D_LPC_isf_reorder		
	D_LPC_isf_2s5s_decode	D_LPC_isf_reorder		
	D_LPC_isf_isp_conversion			
	D_LPC_int_isp_find	D_LPC_isp_a_conversion		
	D_GAIN_lag_concealment	D_GAIN_sort_lag	D_GAIN_insert_lag	
		D_UTIL_random		
	D_GAIN_adaptive_codebook_excitation			
	D_UTIL_random			
	D_ACELP_decode_2t			
	D_ACELP_decode_4t	D_ACELP_decode_1p_N1		
		D_ACELP_add_pulse		
		D_ACELP_decode_2p_2N1		
		D_ACELP_decode_3p_3N1	D_ACELP_decode_2p_2N1	
			D_ACELP_decode_1p_N1	
		D_ACELP_decode_4p_4N	D_ACELP_decode_4p_4N1	D_ACELP_decode_2p_2N1
				D_ACELP_decode_2p_2N1
			D_ACELP_decode_1p_N1	

			D_ACELP_decode_3p_3N1
			D_ACELP_decode_2p_2N1
		D_ACELP_decode_5p_5N	D_ACELP_decode_3p_3N1
			D_ACELP_decode_2p_2N1
		D_ACELP_decode_6p_6N_2	D_ACELP_decode_5p_5N
			D_ACELP_decode_1p_N1
			D_ACELP_decode_4p_4N
			D_ACELP_decode_2p_2N1
			D_ACELP_decode_3p_3N1
	D_UTIL_preemph		
	D_GAIN_pitch_sharpening		
	D_GAIN_decode	D_UTIL_dot_product12	
		D_UTIL_normalized_inverse_sqrt	
		D_GAIN_median	
		D_UTIL_l_extract	
		D_UTIL_pow2	
		D_UTIL_mpy_32_16	
		D_UTIL_log2	D_UTIL_norm_l
			D_UTIL_normalized_log2
	D_UTIL_signal_up_scale	D_UTIL_saturate	
	D_UTIL_signal_down_scale		
	D_GAIN_find_voice_factor	D_UTIL_dot_product12	
		D_UTIL_norm_l	
		D_UTIL_norm_s	
	D_UTIL_norm_s		
	D_UTIL_l_extract		
	D_ACELP_phase_disper		
	D_UTIL_mpy_32_16		
	D_UTIL_l_extract		
	D_GAIN_adaptive_control	D_UTIL_norm_l	
		D_UTIL_inverse_sqrt	
	D_UTIL_dec_synthesis	D_UTIL_saturate	
	D_UTIL_signal_down_scale		
	D_DTX_activity_update	D_UTIL_log2	

4.4 Variables, constants and tables

The data types of variables and tables used in the floating-point implementation are signed integers in 2's complement representation, defined by:

Word8 8 bit variable
UWord8 8 bit unsigned variable

Word16 16 bit variable
Word16 16 bit unsigned variable
Word32 32 bit variable

Floating-point numbers use the IEEE (Institute of Electrical and Electronics Engineers) format:

Float32 8 bit exponent, 23 bit mantissa, 1 bit sign
Float64 11 bit exponent, 52 bit mantissa, 1 bit sign

4.4.1 Description of fixed tables used in the C-code

This clause contains a listing of all fixed tables declared in enc_rom.c and dec_rom.c files.

Table 3: Encoder fixed tables

Format	Table name	Size	Description
Word16	E_ROM_cdown_unusable	7	Attenuation factors for codebook gain in lost frames
Word16	E_ROM_cdown_usable	7	Attenuation factors for codebook gain in bad frames
Float32.	E_ROM_corrweight	199	Weighting of the correlation function in open loop LTP search
Word16	E_ROM_cos	129	Table of cos(x)
Float32	E_ROM_dico1_isf	9*256	1st ISF quantizer of the 1st stage
Float32	E_ROM_dico1_isf_noise	2*64	1st ISF quantizer for comfort noise
Float32	E_ROM_dico21_isf	3*64	1st ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Float32	E_ROM_dico21_isf_36b	5*128	1st ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Float32	E_ROM_dico22_isf	3*128	2nd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Float32	E_ROM_dico22_isf_36b	4*128	2nd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Float32	E_ROM_dico23_isf	3*128	3rd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Float32	E_ROM_dico23_isf_36b	7*64	3rd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Float32	E_ROM_dico24_isf	3*32	4th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Float32	E_ROM_dico25_isf	4*32	5th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Float32	E_ROM_dico2_isf	7*256	2nd ISF quantizer of the 1st stage
Float32	E_ROM_dico2_isf_noise	3*64	2nd ISF quantizer for comfort noise
Float32	E_ROM_dico3_isf_noise	3*64	3rd LSF quantizer for comfort noise
Float32	E_ROM_dico4_isf_noise	4*32	4th LSF quantizer for comfort noise
Float32	E_ROM_dico5_isf_noise	4*32	5th LSF quantizer for comfort noise
Float32	E_ROM_en_adjust	9	Energy scaling factor for each mode during comfort noise
Float32	E_ROM_f_interpol_frac	4	LPC interpolation coefficients
Float32	E_ROM_fir_6k_7k	31	Bandpass FIR filter coefficients for higher band generation
Word16	E_ROM_fir_down	120	Downsample FIR filter coefficients
Float32	E_ROM_fir_ipol	61	Interpol FIR filter coefficients
Word16	E_ROM_fir_up	120	Upsample FIR filter coefficients
Float32	E_ROM_grid	101	Chebyshev polynomial grid points
Float32	E_ROM_hamming_cos	384	LP analysis window
Float32	E_ROM_hp_gain	16	High band gain table for 23.85 kbit/s mode
Float32	E_ROM_inter4_1	4*2*4	Interpolation filter coefficients
Word16	E_ROM_inter4_2	4*2*16	Interpolation filter coefficients
Word16	E_ROM_interpol_frac	4	Interpolation filter coefficients
Float32	E_ROM_isf	16	ISF table for initialization
Word16	E_ROM_isp	16	ISP table for initialization
Word16	E_ROM_isqrt	49	Table used in inverse square root computation
Float32	E_ROM_lag_window	16	Lag window table
Word16	E_ROM_log2	33	Table used in logarithm computation
Float32	E_ROM_f_mean_isf	16	ISF mean
Word16	E_ROM_mean_isf	16	ISF mean
Float32	E_ROM_mean_isf_noise	16	ISF mean for comfort noise
Word16	E_ROM_pdown_unusable	7	Attenuation factors for adaptive codebook gain in lost frames
Word16	E_ROM_pdown_usable	7	Attenuation factors for adaptive codebook gain in bad frames
Word16	E_ROM_pow2	33	Table used in power of two computation
Float32	E_ROM_qua_gain6b	2*64	Gain quantization table for 6-bit gain quantization
Float32	E_ROM_qua_gain7b	2*128	Gain quantization table for 7-bit gain quantization
Uword8	E_ROM_tipos	36	Starting point for codebook search

Table 4: Decoder fixed tables

Format	Table name	Size	Description
Word16	D_ROM_cdown_unusable	7	Attenuation factors for codebook gain in lost frames
Word16	D_ROM_cdown_usable	7	Attenuation factors for codebook gain in bad frames
Word16	D_ROM_cos	129	Table of cos(x)
Word16	D_ROM_dico1_isf	9*256	1st ISF quantizer of the 1st stage
Word16	D_ROM_dico1_isf_noise	2*64	1st ISF quantizer for comfort noise
Word16	D_ROM_dico21_isf	3*64	1st ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Word16	D_ROM_dico21_isf_36b	5*128	1st ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Word16	D_ROM_dico22_isf	3*128	2nd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Word16	D_ROM_dico22_isf_36b	4*128	2nd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Word16	D_ROM_dico23_isf	3*128	3rd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Word16	D_ROM_dico23_isf_36b	7*64	3rd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Word16	D_ROM_dico24_isf	3*32	4th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Word16	D_ROM_dico25_isf	5*32	5th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Word16	D_ROM_dico2_isf	7*256	2nd ISF quantizer of the 1st stage
Word16	D_ROM_dico2_isf_noise	3*64	2nd ISF quantizer for comfort noise
Word16	D_ROM_dico3_isf_noise	3*64	3rd LSF quantizer for comfort noise
Word16	D_ROM_dico4_isf_noise	4*32	4th LSF quantizer for comfort noise
Word16	D_ROM_dico5_isf_noise	4*32	5th LSF quantizer for comfort noise
Word16	D_ROM_fir_6k_7k	31	Bandpass FIR filter coefficients for higher band generation
Word16	D_ROM_fir_7k	31	Bandpass FIR filter coefficients for higher band in 23.85 kbit/s mode
Word16	D_ROM_fir_down	120	Downsample FIR filter coefficients
Word16	D_ROM_fir_up	120	Upsample FIR filter coefficients
Word16	D_ROM_hp_gain	16	High band gain table for 23.85 kbit/s mode
Word16	D_ROM_inter4_2	4*2*16	Interpolation filter coefficients
Word16	D_ROM_interpol_frac	4	LPC interpolation coefficients
Word16	D_ROM_isf	16	ISF table for initialization
Word16	D_ROM_isp	16	ISP table for initialization
Word16	D_ROM_isqrt	49	Table used in inverse square root computation
Word16	D_ROM_log2	33	Table used in logarithm computation
Word16	D_ROM_mean_isf	16	ISF mean
Word16	D_ROM_mean_isf_noise	16	ISF mean for comfort noise
Word16	D_ROM_pdown_unusable	7	Attenuation factors for adaptive codebook gain in lost frames
Word16	D_ROM_pdown_usable	7	Attenuation factors for adaptive codebook gain in bad frames
Word16	D_ROM_ph_imp_low	64	Phase dispersion impulse response
Word16	D_ROM_ph_imp_mid	64	Phase dispersion impulse response
Word16	D_ROM_pow2	33	Table used in power of two computation
Word16	D_ROM_qua_gain6b	2*64	Gain quantization table for 6-bit gain quantization
Word16	D_ROM_qua_gain7b	2*128	Gain quantization table for 7-bit gain quantization

4.4.2 Static variables used in the C-code

In this clause two tables that specify the static variables for the speech encoder and decoder respectively are shown. All static variables are declared within a C **struct.**

Table 5: Speech encoder static variables

Struct name	Variable	Type	Length	Description
Coder_State	mem_speech	Float32	384	speech buffer
	mem_wsp	Float32	371	buffer holding spectral weighted speech
	mem_hp_wsp	Float32	243	highpass wsp
	mem_decim	Float32	30	Open-loop LTP decimation filter memory
	mem_hf	Float32	30	Estimated BP filter memory (23.85 kbit/s mode)
	mem_hf2	Float32	30	Input BP filter memory (23.85 kbit/s mode)
	mem_hf3	Float32	30	Input LP filter memory (23.85 kbit/s mode)
	mem_isp	Float32	16	Old ISP vector
	mem_syn	Float32	16	synthesis filter memory
	mem_syn2	Float32	16	modified synthesis memory
	mem_syn_hf	Float32	16	Higher band synthesis filter memory
	mem_isf	Float32	16	Old ISF vector
	mem_hf_wsp	Float32	9	Open-loop lag gain filter memory
	mem_sig_in	Float32	4	Prefilter memory
	mem_sig_out	Float32	4	HP filter memory in the synthesis
	mem_hp400	Float32	4	HP filter memory
	mem_decim2	Float32	3	Open-loop LTP decimation filter memory
	mem_gp_clip	Float32	2	Memory of pitch clipping
	mem_preemph	Float32	1	Preemphasis filter memory
	mem_deemph	Float32	1	Deemphasis filter memory
	mem_wsp_df	Float32	1	Open-loop LTP deemphasis filter memory
	mem_w0	Float32	1	Weighting filter memory (applied to error signal)
	mem_ol_gain	Float32	1	Open-loop gain
	mem_ada_w	Float32	1	Weighting level depeding on open loop pitch gain
	mem_gc_threshold	Float32	1	Noise enhancer threshold
	mem_gain_alpha	Float32	1	Higher band gain weighting factor (23.85 kbit/s mode)
	mem_ol_lag	Word32	5	Open loop lag history
	mem_T0_med	Word32	1	Weighted open loop pitch lag
	mem_exc	Word16	505	Excitation vector
	mem_isp_q	Word16	16	Old ISP vector
	mem_isf_q	Word16	16	Past quantized ISF prediction error
	mem_gain_q	Word16	4	Gain quantization memory
	mem_subfr_q	Word16	4	Scaling factor history
	mem_tilt_code	Word16	1	Preemhasis filter memory
	mem_q	Word16	1	Old scaling factor
	mem_seed	Word16	1	Random generation seed
	*vadSt	E_DTX_Vad_State	1	See below in this table
	*dtx_encSt	E_DTX_State	1	See below in this table
	mem_first_frame	UWord8	1	First frame indicator
	mem_ol_wght_flg	UWord8	1	Switches lag weighting on and off
	mem_vad_hist	UWord8	1	VAD history
E_DTX_State	mem_isf	Float32	128	LSP history
	mem_distance	Float32	28	ISF history distance matrix
	mem_distance_sum	Float32	8	Sum of ISF history distances
	mem_log_en	Float32	8	Logarithmic frame energy history
	mem_hist_ptr	Word16	1	Pointer to the cyclic history vectors
	mem_log_en_index	Word16	1	Index for logarithmic energy
	mem_cng_seed	Word16	1	Comfort noise excitation seed
	mem_dtx_hangover_count	Word16	1	DTX hangover period
	mem_dec_ana_elapsed_count	Word16	1	Counter for elapsed speech frames in DTX
E_DTX_Vad_State	mem_pow_sum	Float64	1	Power of previous frame
	mem_bckr_est	Float32	12	Background noise estimate
	mem_ave_level	Float32	12	Averaged input components for stationary estimation
	mem_leve	Float32	12	Input levels of the previous frame
	mem_sub_level	Float32	12	Input levels calculated at the end of a frame (lookahead)

Struct name	Variable	Type	Length	Description
	mem_a_data5	Float32	10	Memory for the filter bank
	mem_a_data3	Float32	6	Memory for the filter bank
	mem_sp_max	Float32	1	Maximum level
	mem_speech_level	Float32	1	Estimated speech level
	mem_burst_count	Word16	1	Counts length of a speech burst
	mem_hang_count	Word16	1	Hangover counter
	mem_stat_count	Word16	1	Stationary counter
	mem_vadreg	Word16	1	Flags for intermediate VAD decisions
	mem_pitch_tone	Word16	1	Flags for pitch and tone detection
	mem_sp_est_cnt	Word16	1	Counter for speech level estimation
	mem_sp_max_cnt	Word16	1	Counts frames that contains speech

Table 6: Speech decoder static variables

Struct name	Variable	Type	Length	Description
Decoder_State	mem_gc_thres	Word32	1	Threshold for noise enhancer
	mem_exc	Word16	505	INTERPOL]; /* old excitation vector
	mem_isf_buf	Word16	48	ISF buffer(frequency domain)
	mem_hf	Word16	30	HF band-pass filter memory
	mem_hf2	Word16	30	HF band-pass filter memory
	mem_hf3	Word16	30	HF band-pass filter memory
	mem_oversamp	Word16	24	Synthesis oversampled filter memory
	mem_gain	Word16	23	Gain decoder memory
	mem_syn_hf	Word16	20	HF synthesis memory
	mem_isp	Word16	16	Old ISP (immittance spectral pairs)
	mem_isf	Word16	16	Old ISF (frequency domain)
	mem_isf_q	Word16	16	Past ISF quantizer
	mem_syn_hi	Word16	16	Modified synthesis memory (MSB)
	mem_syn_lo	Word16	16	Modified synthesis memory (LSB)
	mem_ph_disp	Word16	8	Phase dispersion memory
	mem_sig_out	Word16	6	Hp50 filter memory for synthesis
	mem_hp400	Word16	6	Hp400 filter memory for synthesis
	mem_lag	Word16	5	LTP lag history
	mem_subfr_q	Word16	4	Old maximum scaling factor
	mem_tilt_code	Word16	1	Tilt of code
	mem_q	Word16	1	Old scaling factor
	mem_deemph	Word16	1	Speech deemph filter memory
	mem_seed	Word16	1	Random memory for frame erasure
	mem_seed2	Word16	1	Random memory for HF generation
	mem_seed3	Word16	1	Random memory for lag concealment
	mem_T0	Word16	1	Old pitch lag
	mem_T0_frac	Word16	1	Old pitch fraction lag
	mem_vad_hist	UWord16	1	VAD history
	dtx_decSt	D_DTX_State	1	See below in this table
	mem_bfi	UWord8	1	Previous BFI
	mem_state	UWord8	1	BGH state machine memory
	mem_first_frame	UWord8	1	First frame indicator
dtx_decState	mem_isf_buf	Word16	128	ISF vector history (8 frames)
	mem_isf	Word16	16	ISF vector
	mem_isf_prev	Word16	16	Previous ISF vector
	mem_log_en_buf	Word16	8	Logarithmic frame energy history
	mem_true_sid_period_inv	Word16	1	Inverse of true SID update rate
	mem_log_en	Word16	1	Logarithmic frame energy
	mem_log_en_prev	Word16	1	Previous logarithmic frame energy
	mem_cng_seed	Word16	1	Comfort noise excitation seed
	mem_hist_ptr	Word16	1	Index to beginning of LSF history
	mem_dither_seed	Word16	1	Comfort noise dithering seed
	mem_cn_dith	Word16	1	Background noise stationarity information
	mem_since_last_sid	Word16	1	Number of frames since last SID frame
	mem_dec_ana_elapsed_count	UWord8	1	Counts elapsed speech frames after DTX
	mem_dtx_global_state	UWord8	1	DTX state flags
	mem_data_updated	UWord8	1	Flags CNI updates
	mem_dtx_hangover_count	UWord8	1	Counts down in hangover period
	mem_sid_frame	UWord8	1	Flags SID frames
	mem_valid_data	UWord8	1	Flags SID frames containing valid data
	mem_dtx_hangover_added	UWord8	1	Flags hangover period at end of speech

5 Homing procedure

The principles of the homing procedures are described in [2]. The present document only includes a description of the 9 decoder homing frames. For each AMR-WB codec mode, the corresponding decoder homing frame has a fixed set of speech parameters. Table 7 shows the homing frame speech parameters for different modes.

Table 7: Table values for the decoder homing frame parameters for different modes

Mode	Speech Parameters
0	0, 49, 131, 84, 5, 50, 29, 2015, 8,0, 2061, 8,1, 3560, 8,0, 2981, 8
1	0, 49, 131,55, 49, 38,26, 29, 29,3, 15, 7,15, 8, 16,13, 7, 17,16, 8, 0,16, 20, 16,27, 8, 23,0, 27, 0,27, 8
2	0, 49, 131, 55, 49, 38, 26, 29, 58, 1, 7, 63,127, 15, 70, 37, 1, 209, 210, 224, 96, 31, 7, 1, 256, 260, 271, 443, 31, 47, 0, 400, 238, 436, 347, 31
3	0, 49, 131, 55, 49, 38, 26, 29, 58, 1, 3847, 3845, 63, 127, 70, 34, 0, 3128, 4517, 192, 96, 0, 2, 1, 4160, 8036, 267, 443, 31, 46, 0, 3840, 7091, 432, 395, 31
4	0, 49, 131, 55, 49, 38, 26, 29, 58, 1, 3847, 3845, 3847, 3843, 70, 31, 0, 3648, 4764, 824, 2864, 0, 6, 1, 4160, 5220, 4319, 7131, 31, 47, 0, 112, 3764, 219, 211, 31
5	0, 49, 131, 55, 49, 38, 26, 29, 58, 1, 3, 2, 3, 2, 7223, 703, 7223, 703, 70, 0, 1, 3, 2, 2, 3, 9475, 9483, 3090, 8737, 0, 0, 1, 0, 0, 2, 0, 4112, 4400, 8415, 14047, 31, 38, 0, 2, 1, 3, 1, 91, 426, 13545, 12955, 0
6	0, 49, 131, 55, 49, 38, 26, 29, 58, 1, 161, 759, 3, 2, 127, 516, 6167, 447, 70, 11, 1, 264, 641, 2, 3, 123, 562, 8347, 4354, 0, 1, 1, 264, 408, 3, 0, 256, 308, 9487, 14047, 31, 46, 0, 320, 885, 2, 2, 464, 439, 11347, 12739, 0
7	0, 49, 131, 55, 49, 38, 26, 29, 58, 1, 1154, 1729, 1154, 1761, 447, 1519, 959, 495, 70, 27, 1, 1800, 1253, 665, 1960, 546, 164, 1043, 335, 0, 28, 1, 580, 196, 1187, 383, 1031, 1052, 359, 1531, 31, 45, 1, 1024, 893, 1272, 1920, 101, 876, 203, 1119, 31
8	0, 49, 131, 55, 49, 38, 26, 29, 58, 1, 1729, 1154, 1761, 1154, 1519, 959, 495, 447, 70, 3, 42, 1, 580, 1436, 1362, 1250, 901, 714, 24, 45, 0, 0, 0, 1, 68, 708, 1212, 383, 1048, 1611, 1756, 1467, 31, 1, 23, 0, 1536, 1460, 861, 1554, 410, 1368, 1008, 594, 31, 0

6 File formats

This clause describes the file formats used by the encoder and decoder programs. The test sequences defined in [1 also use the file formats described here.

6.1 Speech file (encoder input/decoder output)

Speech files read by the encoder and written by the decoder consist of 16-bit words where each word contains a 14-bit, left aligned speech sample. The byte order depends on the host architecture (e.g. MSByte first on SUN workstations, LSByte first on PCs etc.). Both the encoder and the decoder program process complete frames (of 320 samples) only.

This means that the encoder will only process *n* frames if the length of the input file is $n*320 + k$ words, while the files produced by the decoder will always have a length of $n*320$ words.

6.2 Mode control file (encoder input)

The encoder program can optionally read in a mode control file which specifies the encoding mode for each frame of speech processed. The file is a text file containing one number per speech frame. Each line contains one of the mode numbers 0-8.

6.3 Parameter bitstream file (encoder output/decoder input)

The files produced by the speech encoder/expected by the speech decoder are described in RFC 3267 [7], sections 5.1 and 5.3.

By using a preprocessor definition encoder/decoder can optionally use format described in TS26.201 that defines an octet-aligned frame format (Interface format 2) for the AMR-WB codec.

Annex A (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2002-03	15	SP-020073			Presented at TSG SA#15 for approval	2.0.0	5.0.0
2003-03	19	SP-030090	001	1	Correction to log(0) error in VAD decision with low SNR input signals	5.0.0	5.1.0
2003-03	19	SP-030090	002	1	Correction to decoder with input of long sequence of NO_DATA frames	5.0.0	5.1.0
2003-03	19	SP-030090	003	1	Correction to "D_UTIL_pow2" function to be bitexact with TS26.173 counterpart	5.0.0	5.1.0
2003-03	19	SP-030090	004	1	MMS compatible i/o format option	5.0.0	5.1.0
2003-03	19	SP-030090	005		Correction for handling of RX_NO_DATA frames	5.0.0	5.1.0
2003-03	19	SP-030090	006	1	Ambiguous expressions in the AMR-WB Floating-point C-Code	5.0.0	5.1.0
2003-09	21	SP-030447	008		Possible decoder LPC coefficients overflow	5.1.0	5.2.0
2004-12	26	SP-040844	009	1	Incorrect definition of vector nb_of_bits	5.2.0	6.0.0
2007-03	35	SP-070029	0011		Maintaining bit-exactness with TS 26.173 after Correction in AMR decoder to avoid division by zero in RX-DTX Handling	6.0.0	7.0.0
2007-03	35	SP-070029	0012		Bug fix to SID frame signaling in decoder	6.0.0	7.0.0
2007-09	37	SP-070626	0015	1	Robust operation of AMRWB-decoder	7.0.0	7.1.0
2008-12	42				Version for Release 8	7.1.0	8.0.0
2009-12	46				Version for Release 9	8.0.0	9.0.0
2011-03	51				Version for Release 10	9.0.0	10.0.0
2012-09	57				Version for Release 11	10.0.0	11.0.0
2014-09	65				Version for Release 12	11.0.0	12.0.0
2015-03	67	SP-150094	0016	1	Correction on AMR-WB (noise energy initialization)	12.0.0	12.1.0
2015-12	70				Version for Release 13	12.1.0	13.0.0

History

Document history		
V13.0.0	January 2016	Publication