Technical Specification Group Services and System Aspects *TSGS#14(01)0693*
Meeting #14, Kyoto, Japan, 17-20 December 2001

Source: **TSG-SA WG4**

Title: 3GPP Draft TS 26.204 version 1.0.0 “ANSI-C code for the floating-point AMR wideband speech codec" (Release 5)

Agenda Item: **7.4.3**

Presentation of Specification to TSG SA Plenary

Presentation to: **TSG SA Meeting #14**

Document for presentation: **TS 26.204, Version 1.0.0**

Presented for: **Information**

Abstract of document:

This floating-point codec specification is mainly targeted to be used in multimedia applications or in packet-based applications.

Changes since last presentation:

None, it is version 1.0.0, presented for the first time to TSG SA Plenary.

Outstanding Issues:

Verification phase is under way.

Contentious Issues:

None.

3GPP TS 26.204 V1.0.0 (2001-12)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
ANSI-C code for the Floating-point Adaptive Multi Rate (AMR) Wideband speech codec;
(Release 5)

GLOBAL SYSTEM FOR
MOBILE COMMUNICATIONS

Keywords

GSM, UMTS, codec

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

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 this specification is the only standard conforming non-bit-exact implementation of the Adaptive Multi Rate 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 this specification is a non-bit-exact implementation of the fixed-point encoder producing quality indistinguishable from that of the fixed-point encoder. The decoder in this specification 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 Wideband Speech Codec; Test sequences".

[2] 3GPP TS 26.190: "AMR Wideband Speech Codec; Speech transcoding".

[3] 3GPP TS 26.191: "AMR Wideband Speech Codec; Substitution and muting of lost frames".

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

[5] 3GPP TS 26.193: "AMR Wideband Speech Codec; Source controlled rate operation".

[6] 3GPP TS 26.194: "AMR Wideband Speech Codec; Voice Activity Detection".

3 Definitions and abbreviations

3.1 Definitions

Definition of terms used in the present document, can be found in 3GPP TS 26.190 [2], 3GPP TS 26.191 [3], 3GPP TS 26.192 [4], 3GPP TS 26.193 [5] and 3GPP TS 26.194 [6].

3.2 Abbreviations

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

AMR-WB	Adaptive Multi-Rate Wideband
ANSI	American National Standards Institute
ETS	European Telecommunication Standard
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 this 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, TBA.

4.2 Program execution

The GSM 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 neighboring 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_normalised_inverse_sqrt		
		E_UTIL_l_extract		
		E_UTIL_saturate		
		E_UTIL_mpy_32_16		
		E_UTIL_log2_32	E_UTIL_norm_l	
			E_UTIL_normalised_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_normalised_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_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_normalised_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_normalised_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_normalised_log2	
	D_UTIL_signal_up_scale			
	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_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
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 section 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
TBA	E_ROM_acos[128]	128	table to compute acos(x)
.	E_ROM_cdown_unusable[7]	7	attenuation factors for codebook gain in lost frames
.	E_ROM_cdown_usable[7]	7	attenuation factors for codebook gain in bad frames
.	E_ROM_corrweight[199]	199	weighting of the correlation function in open loop LTP search
	E_ROM_cos[129]	129	table of cos(x)
	E_ROM_dico1_isf[SIZE_BK1 * 9]	9*256	1st ISF quantizer of the 1st stage
	E_ROM_dico1_isf_noise[SIZE_BK_NOISE1 * 2]	2*64	1st ISF quantizer for comfort noise
	E_ROM_dico21_isf[SIZE_BK21 * 3]	3*64	1st ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
	E_ROM_dico21_isf_36b[SIZE_BK21_36b * 5]	5*128	1st ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
	E_ROM_dico22_isf[SIZE_BK22 * 3]	3*128	2nd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
	E_ROM_dico22_isf_36b[SIZE_BK22_36b * 4]	4*128	2nd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
	E_ROM_dico23_isf[SIZE_BK23 * 3]	3*128	3rd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
	E_ROM_dico23_isf_36b[SIZE_BK23_36b * 7]	7*64	3rd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
	E_ROM_dico24_isf[SIZE_BK24 * 3]	3*32	4th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
	E_ROM_dico25_isf[SIZE_BK25 * 4]	4*32	5th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
	E_ROM_dico2_isf[SIZE_BK2 * 7]	7*256	2nd ISF quantizer of the 1st stage
	E_ROM_dico2_isf_noise[SIZE_BK_NOISE2 * 3]	3*64	2nd ISF quantizer for comfort noise
	E_ROM_dico3_isf_noise[SIZE_BK_NOISE3 * 3]	3*64	3rd LSF quantizer for comfort noise
	E_ROM_dico4_isf_noise[SIZE_BK_NOISE4 * 4]	4*32	4th LSF quantizer for comfort noise
	E_ROM_dico5_isf_noise[SIZE_BK_NOISE5 * 4]	4*32	5th LSF quantizer for comfort noise
	E_ROM_en_adjust[9]	9	Energy scaling factor for each mode during comfort noise
	E_ROM_fir_6k_7k[L_FIR]	31	Bandpass FIR filter coefficients for higher band generation
	E_ROM_fir_7k[L_FIR]	31	Bandpass FIR filter coefficients for higher band in 23.85 kbit/s mode
	E_ROM_fir_down[120]	120	Downsample FIR filter coefficients
	E_ROM_fir_up[120]	120	Upsample FIR filter coefficients
	E_ROM_grid[GRID_POINTS + 1]	101	Chebyshev polynomial grid points
	E_ROM_hamming_cos[L_WINDOW]	384	LP analysis window
	E_ROM_hp_gain[16]	16	High band gain table for 23.85 kbit/s mode
	E_ROM_inter4_1[UP_SAMP * 2 * L_INTERPOL1]	4*2*4	interpolation filter coefficients
	E_ROM_inter4_2[UP_SAMP * 2 * L_INTERPOL2]	4*2*16	interpolation filter coefficients
	E_ROM_interpol_frac[NB_SUBFR]	4	interpolation filter coefficients
	E_ROM_isf[M]	16	isf table for initialization
	E_ROM_isp[M]	16	isp table for initialization
	E_ROM_isqrt[49]	49	table used in inverse square root computation
	E_ROM_lag_h[M]	16	high part of the lag window table
	E_ROM_lag_l[M]	16	low part of the lag window table
	E_ROM_log2[33]	33	table used in logarithm computation
	E_ROM_mean_isf[ORDER]	16	ISF mean
	E_ROM_mean_isf_noise[ORDER]	16	ISF mean for comfort noise
	E_ROM_pdown_unusable[7]	7	attenuation factors for adaptive codebook gain in lost frames
	E_ROM_pdown_usable[7]	7	attenuation factors for adaptive codebook gain in bad frames
	E_ROM_ph_imp_low[L_SUBFR]	64	phase dispersion impulse response
	E_ROM_ph_imp_mid[L_SUBFR]	64	phase dispersion impulse response
	E_ROM_pow2[33]	33	table used in power of two computation
	E_ROM_qua_gain6b[64 * 2]	2*64	gain quantization table for 6-bit gain quantization
	E_ROM_qua_gain7b[128 * 2]	2*128	gain quantization table for 7-bit gain quantization
	E_ROM_tipos[36]	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 section 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_decim		Decimation filter memory
	mem_sig_in		Prefilter memory
	mem_preemph		Preemphasis filter memory
	old_speech		speech buffer
	old_wsp		buffer holding spectral weighted speech
	old_exc		excitation vector
	mem_levinson		Levinson memories
	Ispold		Old ISP vector
	ispold_q		Old quantized ISP vector
	past_isfq		past quantized ISF prediction error
	mem_wsp		Open-loop LTP deemphasis filter memory
	mem_decim2		Open-loop LTP decimation filter memory
	mem_w0		weighting filter memory (applied to error signal)
	mem_syn		synthesis filter memory
	tilt_code		Preemhasis filter memory
	old_wsp_max		Open loop scaling factor
	old_wsp_shift		Maximum open loop scaling factor
	Q_old		Old scaling factor
	Q_max		Maximum scaling factor
	gp_clip		memory of pitch clipping
	qua_gain		Gain quantization memory
	old_T0_med		weighted open loop pitch lag
	ol_gain		Open-loop gain
	ada_w		weigthing level depeding on open loop pitch gain
	ol_wght_flg		switches lag weighting on and off
	old_ol_lag		Open loop lag history
	hp_wsp_mem		Open-loop lag gain filter memory

Struct name	Variable	Type[Length]	Description
	old_hp_wsp		Open-loop lag
	vadSt		see below in this table
	dtx_encSt		see below in this table
	first_frame		First frame indicator
	Isfold		Old ISF vector
	L_gc_thres		Noise enhancer threshold
	mem_syn_hi		synthesis filter memory (most significant word)
	mem_syn_lo		synthesis filter memory (least significant word)
	mem_deemph		Deemphasis filter memory
	mem_sig_out		HP filter memory in the synthesis
	mem_hp400		HP filter memory
	mem_oversamp		Oversampling filter memory
	mem_syn_hf		Higher band synthesis filter memory
	mem_hf		Estimated BP filter memory (23.85 kbit/s mode)
	mem_hf2		Input BP filter memory (23.85 kbit/s mode)
	mem_hf3		Input LP filter memory (23.85 kbit/s mode)
	seed2		Random generation seed
	disp_mem		Phase dispersion memory
	vad_hist		VAD history
	Gain_alpha		Higher band gain weighting factor (23.85 kbit/s mode)
dtx_encState	Isf_hist		LSP history (8 frames)
	Log_en_hist		logarithmic frame energy history (8 frames)
	Hist_ptr		pointer to the cyclic history vectors
	Log_en_index		Index for logarithmic energy
	Cng_seed		Comfort noise excitation seed
	D		ISF history distance matrix
	sumD		Sum of ISF history distances
	dtxHangoverCount		is decreased in DTX hangover period
	decAnaElapsedCount		counter for elapsed speech frames in DTX
vadState1	bckr_est		background noise estimate
	ave_level		averaged input components for stationary estimation
	old_level		input levels of the previous frame
	sub_level		input levels calculated at the end of a frame (lookahead)
	a_data5		memory for the filter bank
	a_data3		memory for the filter bank
	burst_count		counts length of a speech burst
	Hang_count	Word16	hangover counter
	Stat_count	Word16	stationary counter
	Vadreg	Word16	15 flags for intermediate VAD decisions
	Tone_flag	Word16	15 flags for tone detection
	sp_est_cnt	Word16	Speech level estimation counter
	Sp_max	Word16	Maximum signal level
	sp_max_cnt	Word16	Maximum level estimation counter
	Speech_level	Word16	Speech level
	prev_pow_sum	Word16	Power of previous frame

Table 6: Speech decoder static variables

Struct name	Variable	Type[Length]	Description
Decoder_State	old_exc	Word16[248]	excitation vector
	ispold	Word16[16]	Old ISP vector
	isfold	Word16[16]	Old ISF vector
	isf_buf	Word16[48]	ISF vector history
	past_isfq	Word16[16]	past quantized ISF prediction error
	tilt_code	Word16	Preemhasis filter memory
	Q_old	Word16	Old scaling factor
	Qsubfr	Word16	Scaling factor history
	L_gc_thres	Word16	Noise enhancer threshold
	mem_syn_hi	Word16[16]	synthesis filter memory (most significant word)
	mem_syn_lo	Word16[16]	synthesis filter memory (least significant word)
	mem_deemph	Word16	Deemphasis filter memory
	mem_sig_out	Word16[6]	HP filter memory in the synthesis
	mem_oversamp	Word16[24]	Oversampling filter memory
	mem_syn_hf	Word16[20]	Higher band synthesis filter memory
	mem_hf	Word16[30]	Estimated BP filter memory (23.85 kbit/s mode)
	mem_hf2	Word16[30]	Input BP filter memory (23.85 kbit/s mode)
	mem_hf3	Word16[30]	Input LP filter memory (23.85 kbit/s mode)
	seed	Word16	Random code generation seed for bad frames
	seed2	Word16	Random generation seed for higher band
	old_T0	Word16	Old LTP lag (integer part)
	old_T0_frac	Word16	Old LTP lag (fraction part)
	lag_hist	Word16[5]	LTP lag history
	dec_gain	Word16[23]	Gain decoding memory
	seed3	Word16	Random LTP lag generation seed for bad frames
	disp_mem	Word16[8]	Phase dispersion memory
	mem_hp400	Word16[6]	HP filter memory
	prev_bfi	Word16	Previous BFI
	state	Word16	BGH state machine memory
	first_frame	Word16	First frame indicator
	dtx_decSt	dtx_decState*	see below in this table
	Vad_hist	Word16	VAD history
dtx_decState	Since_last_sid	Word16	number of frames since last SID frame
	true_sid_period_inv	Word16	inverse of true SID update rate
	log_en	Word16	logarithmic frame energy
	old_log_en	Word16	previous value of log_en
	isf	Word16[16]	ISF vector
	Isf_old	Word16[16]	Previous ISF vector
	Cng_seed	Word16	Comfort noise excitation seed
	Isf_hist	Word16[128]	ISF vector history (8 frames)
	Log_en_hist	Word16[8]	logarithmic frame energy history
	Hist_ptr	Word16	index to beginning of LSF history
	dtxHangoverCount	Word16	counts down in hangover period
	DecAnaElapsedCount	Word16	counts elapsed speech frames after DTX
	sid_frame	Word16	flags SID frames
	valid_data	Word16	flags SID frames containing valid data
	log_en_adjust	Word16	mode-dependent frame energy adjustment
	dtxHangoverAdded	Word16	flags hangover period at end of speech
	dtxGlobalState	Word16	DTX state flags
	data_updated	Word16	flags CNI updates

5 Homing procedure

The principles of the homing procedures are described in [2]. This specification 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 9 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 section 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 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**
2001-12	14	SP-010693			Version 1.0.0 (for information)		