Technical Specification Group Services and System Aspects *TSGS#25 (04)0640*
Meeting #25, Palm Springs, USA, 13-16 September 2004

Source: **TSG-SA WG4**

Title: 3GPP TS 26.304 version 2.0.0 "Extended Adaptive Multi-Rate - Wideband codec; Floating-point ANSI-C code" (Release 6)

Agenda Item: **7.4.3**

Presentation of Specification to TSG SA Plenary

Presentation to: **TSG SA Meeting #25**

Document for presentation: **TS 26.304 "Extended Adaptive Multi-Rate - Wideband codec; Floating-point ANSI-C code", Version 2.0.0 (Release 6)**

Presented for: **Discussion / Decision**

Abstract of document:

The present document contains an electronic copy of the ANSI-C code for the Floating-point Extended Adaptive Multi-Rate Wideband codec. Alternatively, fixed-point ANSI-C code is specified in 3GPP TS 26.273. The floating-point codec/encoder/decoder specified in this document or the fixed-point codec/encoder/decoder may be used depending on if the implementation platform is better suited for a floating-point or a fixed-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 defines, besides the fixed-point c-code, one valid reference implementation of the Extended Adaptive Multi-Rate Wideband transcoder (3GPP TS 26.290).

Changes since last presentation:

None.

Outstanding Issues:

At SA#24 the way forward for the selection of audio codecs was formulated in TD SP-040481. Following the guidance contained therein, SA4 agreed to forward this document to TSG SA#25, as one of the specifications for the Extended Adaptive Multi-Rate - Wideband codec.

Contentious Issues:

None.

Comment(s):

None.

3GPP TS 26.304 V2.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Extended AMR Wideband codec;
Floating-point ANSI-C code

(Release 6)

Keywords

AMR-WB, AMR-WB+, audio CODEC, Extended AMR Wideband 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 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 Extended Adaptive Multi-Rate Wideband codec. Alternatively, fixed-point ANSI-C code is specified in 3GPP TS 26.273 [1]. The floating-point codec/encoder/decoder specified in this document or the fixed-point codec/encoder/decoder specified in [1] may be used depending on if the implementation platform is better suited for a floating-point or a fixed-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 defines, besides the fixed-point c-code specified in [1], one valid reference implementation of the Extended Adaptive Multi-Rate Wideband transcoder (3GPP TS 26.290 [2]). Standard conformance is enforced by meeting the conformance criteria defined in [3].

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 26.273: "ANSI-C code for the Fixed-point Extended AMR Wideband codec".

[2] 3GPP TS 26.290: " Audio codec processing functions; Extended AMR Wideband codec; Transcoding functions ".

[3] 3GPP TS 26.xxx: "3GPP audio codecs, Conformance".

[4] 3GPP TS 26.201: " AMR Wideband speech codec; frame structure".

[5] IETF Internet Draft: "Real-Time Transport Protocol (RTP) Payload Format for Extended AMR Wideband (AMR-WB+) Audio Codec", Sjoberg J., Westerlund M. and Lakaniemi A., http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-amrwbplus-01.txt, July 2004.

[6] 3GPP TS 26193: " AMR Wideband speech codec; Source controlled rate operation".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions are given in TS 26.290 [2].

3.2 Abbreviations

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

AMR-WB+	Extended 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 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 2000 SP4 and Microsoft Visual C++ v.6.0 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 the files divided in five different directories, all present in the directory *c-code*. The directories are: *common, decoder, encoder, lib_amr* and *include*. The distributed files with suffix "c" contain the source code and the files with suffix "h" are the header files.

Project and workspace files are provided in the directory *MSVC*.

4.2 Program execution

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

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

The programs should be called like:

- encoder [encoder options] –if <audio input file> -of <parameter file>;
- decoder [decoder options] –if <parameter file> -of <audio output file>.

The input files contain one or two channels of 16-bit linear encoded PCM audio samples stored in the *wav* file format and the parameter files contain encoded audio 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 audio codec.

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: Encoder call structure

coder_amrwb_plus_stereo				
	decim_12k8			
		interpol		
	hp50_12k8			
	mix_ch			
	E_UTIL_f_preemph			
	mvr2r			
	coder_lf			
		mvr2r		
		E_UTIL_autocorr		
		lag_wind		
		E_LPC_lev_dur		
		E_LPC_a_isp_conversion		
			E_LPC_chebyshev	
		int_lpc_npl		
			E_LPC_f_isp_a_conversion	
				E_LPC_isp_f_pol_get
		find_wsp		
			E_LPC_a_weight	
			E_UTIL_residu	
			E_UTIL_deemph	
		E_GAIN_lp_decim2		
		E_GAIN_open_loop_search		
		E_GAIN_olag_med		
			E_GAIN_sort	
		E_LPC_isp_isf_conversion		
		qpisf_2s_46b		
			VQ_stage1(qpisf_2s.c)	
			sub_VQ(qpisf_2c.c	

			)		
			qpisf_2s_46b		
				E_LPC_isf_reorderPlus	
		isf2isp			
		coder_acelp			
			E_UTIL_residu		
			E_LPC_a_weight		
			E_UTIL_deemph		
			mvr2r		
			set_zero		
			E_UTIL_synthesis		
			E_UTIL_f_preemph		
			E_GAIN_closed_loop_search		
				E_GAIN_norm_corr	
					E_UTIL_f_convolve
				E_GAIN_norm_corr_interpolate	
			pred_lt4		
			E_UTIL_f_convolve		
			E_ACELP_xy1_corr		
			E_ACELP_codebook_target_update		
			E_GAIN_f_pitch_sharpening		
			E_ACELP_xh_corr		
			E_ACELP_4t		
				E_ACELP_h_vec_corr1	
				E_ACELP_h_vec_corr2	
				E_ACELP_2pulse_search	
				E_ACELP_quant_4p_4N	
					E_ACELP_quant_

					1p_N1	
					E_ACELP_quant_3p_3N1	
						E_ACELP_quant_2p_2N2
						E_ACELP_quant_1p_N1
					E_ACELP_quant_4p_4N1	
						E_ACELP_quant_2p_2N2
					E_ACELP_quant_2p_2N2	
			E_ACELP_xy2_corr			
			q_gain2_plus			
		segsnr				
		coder_tcx				
			cos_window			
			E_LPC_a_weight			
			E_UTIL_residu			
			E_UTIL_deemph			
			fft9			
				fft_rel		
			adap_low_freq_emph			
			AVQ_cod			
				RE8_PPV		
					nearest_neighbor_2D8	
			adap_lo_freq_deemph			
			ifft9			
				ifft_rel		
			get_gain			
			q_gain_tcx			
			E_UTIL_f_preemph			
			E_UTIL_synthesis			
		mvi2i				
	coder_hf					

		E_UTIL_autocorr		
		lag_wind		
		E_LPC_lev_dur		
		E_LPC_a_isp_conversion		
			E_LPC_chebyshev	
		int_lpc_npl		
			E_LPC_f_isp_a_conversion	
				E_LPC_isp_f_pol_get
		mvr2r		
		E_LPC_isp_isf_conversion		
		q_isf_hf		
			sub_VQ(q_isf_hf.c)	
			E_LPC_isf_reorderPlus	
		isf2isp		
		match_gain_6k4		
			set_zero	
			E_UTIL_residu	
			E_UTIL_synthesisPlus	
		int_gain		
		E_UTIL_residu		
		E_UTIL_synthesisPlus		
		E_LPC_a_weight		
		E_UTIL_residuPlus		
		q_gn_hf		
	band_split_taligned_2k			
		interpol		
	coder_stereo_x			
		cod_hi_stereo		
			mvr2r	
			residu	

			cholsolc			
			smooth_ener_filter			
			quant_filt			
				pmsvq		
					msvq	
						m_cbcod
						min_msvq
			fir_filt			
			my_max			
			quant_gain			
				pmsvq		
					msvq	
						m_cbcod
		cod_tcx_stereo				
			set_zero			
			mvr2r			
			ctcx_stereo			
				cos_window		
				get_gain		
				q_gain_pan		
				fft3		
					fft_rel	
				adap_low_freq_emph		
				AVQ_cod		
					RE8_PPV	
						nearest_neighbor_2D8
				adap_lo_freq_deemph		
				ifft3		
					ifft_rel	
				q_gain_tcx		
			segsnr			
			mvi2i			
	int2bin					
	enc_prm					
		AVQ_encmux				

- split_idx_noovf
  - sort(avq_cod.c)
  - RE8_cod
    - RE8_vor
      - re8_identify_absolute
        - Leader
      - re8_coord
      - re8_k2y
        - RE8_PPV
          - nearest_neighbor_2D8
    - re8_compute_base_index
      - re8_compute_rank_of_permutation_and_s
  - calc_bits
- writ_all_nq
  - calc_bits
- writ_all_i
  - init_pos_i_ovf
    - chk_ovf
  - chk_ovf
  - writ_I
  - writ_k
  - writ_ovf

int2bin

unpack4bits
- int2bin

enc_prm_stereo_x
- int2bin
- AVQ_encmux
  - split_idx_noovf
    - sort(avq_cod.c)
    - RE8_cod
      - RE8_vor
        - re8_identify_absolute
          - Leader
        - re8_coord

						re8_k2y		
							RE8_PPV	
								nearest_neighbor_2D8
					re8_compute_base_index			
						re8_compute_rank_of_permutation_and_s		
				calc_bits				
			writ_all_nq					
				calc_bits				
			writ_all_i					
				init_pos_i_ovf				
					chk_ovf			
				chk_ovf				
				writ_l				
				writ_ovf				
				writ_k				
		unpack4bits_d						
			int2bin					
	enc_prm_hf							
		int2bin						

Table 2: Decoder call structure

decoder_amrwb_plus				
	bin2int			
	dec_prm			
		bin2int		
		pack4bits(dec_prm.c)		
			bin2int	
	dec_prm_stereo_x			
		bin2int		
		pack4bits_d		
			bin2int	
		AVQ_demuxdec		
			read_all_nq(avq_d	

			ec.c)			
				read_nq(avq_dec.c)		
			read_all_i(avq_dec.c)			
				init_pos_i_ovf(avq_dec.c)		
					chk_ovf(avq_dec.c)	
				split_n(avq_dec.c)		
				chk_ovf(avq_dec.c)		
				read_I(avq_dec.c)		
				read_ovf(avq_dec.c)		
				read_k(avq_dec.c)		
			RE8_dec			
				re8_decode_base_index		
					re8_decode_rank_of_permutation	
				re8_k2y		
					RE8_PPV	
						nearest_neighbor_2D8
	dec_prm_hf					
		bin2int				
	decoder_amrwb_plus_1					
		mvr2r				
		decoder_lf				
			mvr2r			
			qpisf_2s_46b			
				E_LPC_isf_reorderPlus		
			isf2isp			
			int_lpc_npl			
				E_LPC_f_isp_a_conversion		
					E_LPC_isp_f_pol_get	
			decoder_tcx			

				AVQ_demuxdec				
					read_all_nq(avq_dec.c)			
						read_nq(avq_dec.c)		
					read_all_i(avq_dec.c)			
						init_pos_i_ovf(avq_dec.c)		
							chk_ovf(avq_dec.c)	
						Split_n(avq_dec.c)		
						chk_ovf(avq_dec.c)		
						read_l(avq_dec.c)		
						read_k(avq_dec.c)		
						read_ovf(avq_dec.c)		
					RE8_dec			
						re8_decode_base_index		
							re8_decode_rank_of_permutation	
						re8_k2y		
							RE8_PPV	
								nearest_neighbor_2D8
				set_zero				
				rnd_ph16				
					E_UTIL_random			
				adap_lo_freq_deemph				
				mvr2r				
				find_mpitch(dec_tcx.c)				
				ifft9				
					ifft_rel			
				d_gain_tcx				
				cos_window				
				E_UTIL_f_preemph				
				E_LPC_a_weight				

				E_UTIL_synthesis			
				E_UTIL_residu			
			decoder_acelp				
				pred_lt4			
				D_ACELP_decode_4t			
					D_ACELP_decode_4p_4N		
						D_ACELP_decode_1p_N1	
						D_ACELP_decode_3p_3N1	
							D_ACELP_decode_2p_2N1
							D_ACELP_decode_1p_N1
						D_ACELP_decode_2p_2N1	
						D_ACELP_decode_4p_4N1	
							D_ACELP_decode_2p_2N1
					D_ACELP_add_pulse		
				E_UTIL_f_preemph			
				E_GAIN_f_pitch_sharpening			
				d_gain2_plus			
				E_UTIL_synthesis			
				E_LPC_a_weight			
				E_UTIL_residu			
				E_UTIL_deemph			
				mvr2r			
				set_zero			
		E_UTIL_deemph					
		bass_postfilter					
			mvr2r				
			short_pitch_tracker (bass_pf.c)				
		decoder_hf					
			d_isf_hf				

				E_LPC_isf_reorderPlus	
			isf2isp		
			int_lpc_npl		
				E_LPC_f_isp_a_conversion	
					E_LPC_isp_f_pol_get
			mvr2r		
			match_gain_6k4		
				set_zero	
				E_UTIL_residu	
				E_UTIL_synthesisPlus	
			int_gain		
			d_gain_hf		
			soft_exc_hf		
			E_UTIL_synthesisPlus		
			smooth_ener_hf		
		delay			
		oversamp_12k8			
			interpol		
	decoder_stereo_x				
		mvr2r			
		band_split_taligned_2k			
			interpol		
		dec_tcx_stereo			
			dtcx_stereo		
				cos_window	
				adap_lo_freq_deemph	
				ifft3	
					ifft_rel
				d_gain_tcx	
				crosscorr	
				glev_s	
			my_min		

			my_max		
		dec_hi_stereo			
			dec_filt		
				pmsvq_inv	
					msvq_inv
			dec_gain		
				pmsvq_inv	
					msvq_inv
			mvr2r		
			residu		
			fir_filt		
			syn_filt		
		delay			
		band_join_2k			
			interpol		
	hp50_12k8				
	oversamp_12k8				
		interpol			

4.4 Variables, constants and tables

4.4.1 Description of fixed tables used in the C-code

This clause contains a listing of all fixed tables declared in tables_plus.c and tables_stereo.c files.

Table 3: Encoder fixed tables

Format	Table name	Size	Description
Float32	NBITS_CORE	8	Core bit-rates
Float32	T_sin	1152	FFT Sine table
Float32	T_cos	1152	FFT Cosine table
Float32	filter_32k	61	FIR table for decimation/oversampling
Float32	filter_32k_hf	61	FIR table for decimation/oversampling
Float32	filter_32k_7k	61	FIR table for decimation/oversampling
Float32	filter_48k	185	FIR table for decimation/oversampling
Float32	Filter_48k_hf	185	FIR table for decimation/oversampling
Float32	filter_8k	61	FIR table for decimation/oversampling
Float32	isf_init	16	Initial ISF memory
Float32	Mean_isf	16	Means of ISFs
Float32	Dico1_isf	2304	1st stage codebook, isf0 to isf8
Float32	Dico2_isf	1792	1st stage codebook, isf9 to isf15
Float32	Dico21_isf	192	2nd stage codebook, isf2_0 to isf 2_2
Float32	Dico22_isf	384	2nd stage codebook, isf2_3 to isf 2_5
Float32	Dico23_isf	384	2nd stage codebook, isf2_6 to isf 2_8
Float32	Dico24_isf	96	2nd stage codebook, isf2_9 to isf 2_11
Float32	Dico25_isf	128	2nd stage codebook, isf2_12 to isf 2_15
Float32	Dico21_isf_36b	640	1st stage codebook, (36b) split 1
Float32	Dico22_isf_36b	512	1st stage codebook, (36b) split 2
Float32	Dico23_isf_36b	448	1st stage codebook, (36b) split 3
Float32	Dico_gain_hf	512	Quantization table for one-stage HF gain
Float32	Mean_isf_hf_12k8	8	Means of ISFs (full band)
Float32	dico1_isf_hf_12k8	32	1nd stage isf codebook (full band)
Float32	mean_isf_hf_low_rate	8	Means of isfs
Float32	Dico1_isf_hf_low_rate	32	1st stage isf codebook
Float32	dico2_isf_hf	1024	2nd stage isf codebook
Float32	Lag_window	17	Lag window
Float32	Filt_lp	13	Low-pass fir filter for bass post filter
Float32	Sin20	20	Random phase
Float32	Inter4_2	65	¼ resolution interpolation filter
Float32	VadFiltBandFreqs	12	Open-loop classifier
Float32	Bw	12	Open-loop classifier
Float32	Lwg	8	Open-loop claissifier
Float32	Gain_jf_ramp	64	HF gain ramp for wb->wb+ switiching
Float32	Inter2_coef	12	Filter coefficients for band join/split
Float32	Filter_LP180	2341	Filter for 48 kHz interpolation
Float32	StereoNbits	18	Stereo bit-rates
Float32	Filter_2k	321	2k decimation filter
Float32	Cb_filt_hi_mean	9	Average filter
Float32	Filt_hi_mscb4a	16*9	
Float32	Filt_hi_mscb_7a	16*9	
Float32	Filt_hi_mscb_7b	8*9	
Float32	Cb_gain_hi_mean	2	Average gain vector
Float32 Float32	Gain_hi_mscb_2a Gain_hi_mscb_5a TBC	4*2 32*2	

Table 4: Decoder fixed tables

Format	Table name	Size	Description
Same as encoder			

4.4.2 Static variables used in the C-code

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

Table 5: Encoder static variables

struct name	type	variable	size	description
Coder_StState				
	float	mem_decim	1608	speech decimated filter memory
	int	decim_frac	1	Fractional decimation factor
	float	mem_sig_in	4	hp filter memory
	float	mem_preemph	1	speech preemphasis filter mem
	float	mem_decim_hf	46	HF filter memory
	float	old_speech_hf	528	HF old speech vector
	float	past_q_isf_hf	8	HF past quantized isf
	float	ispold_hf	8	HF old isp
	float	ispold_q_hf	8	HF quantized old isp
	float	old_gain;	1	HF old gain match
	float	mem_hf1	8	HF memory for gain 1
	float	mem_hf2	8	HF memory for gain 2
	float	mem_hf3	8	HF memory for gain 3
	float	old_exc	375	old excitation
	float*	mean_isf_hf	1	isf codebook mean
	float*	dico1_isf_hf	1	isf codebook first stage
Coder_State_Plus				
	Coder_StState	left	2614	state for left channel
	Coder_StState	right	2614	state for right channel
	float	old_chan	528	old left signal
	float	old_chan_2k	140	old left signal 2kHz sampl. rate
	float	old_chan_hi	448	old left signal HB
	float	old_speech_2k	140	old mono signal 2kHz sampl. rate
	float	old_speech_hi	448	old mono signal HB
	float	old_speech_pe	528	past pre-emphasised mono
	float	old_wh	9	past weighted filter
	float	old_wh_q	9	past quantized weighted filter
	float	old_gm_gain	2	past gain matching
	float	old_exc_mono	9	past mono excitation
	float	filt_energy_threshold	1	filter energy thershold
	float	w_window	64	weighting window
	PMSVQ*	*filt_hi_pmsvq	1	MSVQ quantizer
	PMSVQ*	*gain_hi_pmsvq	1	MSVQ quantizer
	int	mem_stereo_ovlp_size	1	past stereo overlap size
	float	mem_stereo_ovlp	32	past stereo overlap
	NCLASSDATA	*stClass	1	use case B classifier
	VadVars	*vadSt	1	VAD state
	short	vad_hist	1	VAD history
	float	old_speech	528	old speech
	float	old_synth	16	synthesis memory
	float	past_isfq	16	past isf quantizer
	float	old_wovlp	128	last tcx overlap
	float	old_d_wsp	187	Weighted speech vector
	float	old_exc	392	old excitation vector
	float	old_mem_wsyn	1	weighted synthesis memory
	float	old_mem_w0	1	weighted speech memory
	float	old_mem_xnq	1	quantized target memory

	int	old_ovlp_size	1	last tcx overlap size
	float	isfold	16	old isf frequency domain
	float	ispold	16	old isp
	float	ispold_q	16	quantized old isp
	float	mem_wsp	1	wsp vector mem
	float	mem_lp_decim2	3	wsp decimator filter mem
	float	ada_w	1	open loop LTP
	float	ol_gain	1	open loop LTP
	short	ol_wght_flg	1	open loop LTP
	long int	old_ol_lag	5	past openloop lag
	int	old_T0_med	1	past pitch
	float	hp_old_wsp	699	past HP weighted speech
	float	hp_ol_ltp_mem	7	past HP openloop long term prediction
	float	window	512	LP analysis window
	short	SwitchFlagPlusToWB	1	flag for switching to AMR-WB
	float	mem_gain_code	4	past code gain
	short	prev_mod	1	past frame type

Table 6: Decoder static variables

struct name	type	variable	size	description
Decoder_StState				
	float	mem_oversamp	72	Memory oversampling
	int	over_frac	1	Fractional overcloking factor
	float	mem_oversamp_hf	24	memory
	float	past_q_isf_hf	8	HF past quantized isf
	float	past_q_isf_hf_other	8	HF past quantized isf for the other channel when mono decoding stereo
	float	past_q_gain_hf	1	HF past quantized gain
	float	past_q_gain_hf_other	1	HF past quantized gain for the other channel when mono decoding stereo
	float	old_gain	1	HF old gain match
	float	ispold_hf	8	HF old isp
	float	threshold;	1	HF memory for smooth ener
	float	mem_syn_hf	8	HF synthesis memory
	float	mem_d_tcx	96	delay compensation memory
	float	mem_d_nonc	64	Non causality delay
	float	mem_synth_hi	16	High band sunthesis memory
	float	mem_sig_out	4	hp filter memory
	float	old_synth_hf	512	synch delay memory
	float	lp_amp	1	memory for soft exc
	float*	mean_isf_hf	1	isf codebook mean
	float*	dico1_isf_hf	1	isf codebook first stage
Decoder_State_Plus				
	Decoder_StState	left	828	State for left channel
	Decoder_StState	right	828	State for right channel
	float	mem_left_2k	20	2kHz memory on left chan
	float	mem_right_2k	20	2kHz memory on right chan
	float	mem_left_hi	64	HB memory left channel
	float	mem_right_hi	64	HB memory right channel
	float	my_old_synth_2k	35	old 2kHz synthesis
	float	my_old_synth_hi	128	old HB synthesis
	float	my_old_synth	148	old stereo synth
	float	old_AqLF	85	old quantized LPC
	float	old_wh	9	old decoded filter
	float	old_wh2	9	old decoded filter 2
	float	old_exc_mono	9	old mono excitation
	float	old_gain_left	4	old gain on left chan
	float	old_gain_right	4	old gain on right chan
	float	old_wh_q	9	past quantized filter
	float	old_gm_gain	2	past gain matching
	float	w_window	64	weighted synthesis window
	PMSVQ	*filt_hi_pmsvq	1	past MSVQ filter
	PMSVQ	*gain_hi_pmsvq	1	past MSVQ gain
	int	mem_stereo_ovlp_size	1	past stereo overlap size
	float	mem_stereo_ovlp	32	past stereo overlap
	int	last_stereo_mode	1	past stereo mode
	float	side_rms	1	side signal RMS
	float	h	9	current filter
	float	mem_balance	1	past balance factor

	int	fer_hist	500	frame errasure history
	int	fer_hist_ptr	1	frame erasure pointer
	float	fer_mean	1	frame erasure mean
	float	old_xri	1148	old spectral coefficeints
	int	last_mode	1	last mode in previous 80ms frame
	float	mem_sig_out	4	hp50 filter memory for synthesis
	float	mem_deemph	1	speech deemph filter memory
	int	prev_lpc_lost	1	previous lpc is lost when = 1
	float	old_synth	16	synthesis memory
	float	old_exc	392	old excitation vector
	float	isfold	16	old isf (frequency domain)
	float	ispold	16	old isp (immittance spectral pairs)
	float	past_isfq	16	past isf quantizer
	float	wovlp	128	last weighted synthesis for overlap
	int	ovlp_size	1	overlap size
	float	isf_buf	51	old isf (for frame recovery)
	int	old_T0	1	old pitch value (for frame recovery)
	int	old_T0_frac	1	old pitch value (for frame recovery)
	short	seed_ace	1	seed memory (for random function)
	float	mem_wsyn	1	TCX synthesis memory
	short	seed_tcx	1	seed memory (for random function)
	float	wsyn_rms	1	rms value of weighted synthesis
	float	past_gpit	1	past gain of pitch (for frame recovery)
	float	past_gcode	1	past gain of code (for frame recovery)
	int	pitch_tcx	1	for bfi
	float	gc_threshold	1	GC threshold
	float	old_synth_pf	503	Bass post-filter: old synthesis
	float	old_noise_pf	24	bass post-filter: noise memory
	int	old_T_pf	2	bass post-filter: old pitch
	float	old_gain_pf	2	Bass post-filter: old pitch gain
	float	*mean_isf_hf	1	HF isf codebook in-use
	float	*dico1_isf_hf	1	HF isf codebook in-use
	float	mem_gain_code	4	past code gain
	float	mem_lpc_hf	9	past HF lpc filter
	float	mem_gain_hf	1	past HF gain
	short	ramp_state	1	ramp state

5 File formats

This clause describes the file formats used by the encoder and decoder programs.

5.1 Audio file (encoder input/decoder output)

Audio files read by the encoder must be formatted as 16 bits PCM wave (*.wav) files. The decoder output is written as a 16 bit PCM wave file (*.wav).

Note that the decoder, with proper command line switch, can produce a mono file from a stereo bit-stream.

5.2 Parameter bitstream file (encoder output/decoder input)

For AMR-WB+ operation, the files produced by the audio encoder/expected by the audio decoder contain an arbitrary number of frames containing a header and data octets in the following format.

MONO RATE	STEREO EXTENSION RATE	FREQUENCY SCALE	B1	B2	…	Bmn	S1	S2	…	Ssn

Each box corresponds to one octet (Word8) value in the bitstream file, for a total of 3+mn+sn octets per frame, where mn is the number of encoded octets in the frame for the mono rate and sn is the number of encoded octets in the frame for the stereo extension rate. For mono encoding the value of sn is equal to zero.

For AMR-WB modes, the file has the following format:

MONO RATE	AMR-WB bitstream in IF2 format [4]

The header fields have the following meaning:

MONO_RATE:

The rate of AMR-WB or mono rate of Extended AMR-WB. The values of MONO_RATE are given in Table 7 below.

Table 7: Description of MONO_RATE header field.

MONO RATE MODE	*Mono rate(incl. BWE)* (bits/frame)	*Number of data bytes*
0x00	AMR-WB 6.60 kbit/s mode	18
0x01	AMR-WB 8.85 kbit/s mode	23
0x02	AMR-WB 12.65 kbit/s mode	33
0x03	AMR-WB 14.25 kbit/s mode	37
0x04	AMR-WB 15.85 kbit/s mode	41
0x05	AMR-WB 18.25 kbit/s mode	47
0x06	AMR-WB 19.85 kbit/s mode	51
0x07	AMR-WB 23.05 kbit/s mode	59
0x08	AMR-WB 23.85 kbit/s mode	61
0x09	AMR-WB SID	6
0x0A-0x0D	RESERVED	
0x0E	AMR-WB FRAME_ERASURE	0
0x0F	AMR-WB NO_DATA	0
0x10	AMR-WB+ 208 bit/frame	26
0x11	AMR-WB+ 240 bit/frame	30
0x12	AMR-WB+ 272 bit/frame	34

0x13	AMR-WB+ 304 bit/frame	38
0x14	AMR-WB+ 336 bit/frame	42
0x15	AMR-WB+ 384 bit/frame	48
0x16	AMR-WB+ 416 bit/frame	52
0x17	AMR-WB+ 480 bit/frame	60
0x18-0x1D	RESERVED	
0x1E	FRAME_ERASURE	0
0x1F	NO_DATA	0

STEREO_EXTENSION_RATE:

The mode of the stereo extension bit rate. The values of STEREO_EXTENSION_RATE are given in Table 8 below.

Table 8: Description of STEREO_EXTENSION_RATE header field.

STEREO EXTENSION RATE MODE	*Stereo extension rate(incl. BWE)* (bits/frame)	*Number of data octets*
0xFF	No Stereo Extension	0
0x00	40 bits/frame	5
0x01	48 bits/frame	6
0x02	56 bits/frame	7
0x03	64 bits/frame	8
0x04	72 bits/frame	9
0x05	80 bits/frame	10
0x06	88 bits/frame	11
0x07	96 bits/frame	12
0x08	104 bits/frame	13
0x09	112 bits/frame	14
0x0A	120 bits/frame	15
0x0B	128 bits/frame	16
0x0C	136 bits/frame	17
0x0D	144 bits/frame	18
0x0E	152 bits/frame	19
0x0F	160 bits/frame	20

FREQUENCY_SCALE

This field is related to the internal sampling frequency of the audio codec, which in its turn is related to the frame size in ms. The internal sampling frequency in kHz is given by

Fs = FREQUENCY_SCALE x 25.6/96 kHz.

For a value FREQUENCY_SCALE=96, the internal sampling frequency is 25.6 kHz and the 2048-sample encoded super frame corresponds to 80 ms, giving a packet size of 20 ms. For a value FREQUENCY_SCALE=120, the internal sampling frequency is 32 kHz and the 2048-sample encoded super frame corresponds to 64 ms, giving a packet size of 16 ms. The value of FREQUENCY_SCALE is limited to the range 48-144 corresponding to internal sampling frequency range of 12.8-38.4 kHz.

The AMR-WB+ packet is formed as a concatenation of AMR-WB+ Header and AMR-WB+ data (mono followed by stereo). The data octets in each packet are packetized according to the detailed bit allocation given in [2], tables 14 to 20.

For AMR-WB+ operation, the first three octets contain the header fields MONO_RATE, STEREO_EXTENSION_RATE and FREQUENCY_SCALE. The nm+ns data octets follow. The first bit of the AMR-WB+ data b0 is placed in bit 8 of octet 4. Table 9 shows the composition for the example of AMR-WB+ packet with 272 bits/frame mono rate, 88 bits/frame stereo extension rate, and FREQUENCY_SCALE=96 corresponding to 25.6 internal sampling frequency and 20 ms packets (80 ms superframe).

Table 9: AMR-WB+ packet for 272 bits/frame mono rate, 88 bits/frame stereo extension rate, and FREQUENCY_SCALE=96.

	MSB							LSB
Octet	**bit 8**	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**
1	MONO_RATE =18 (272bits/frame)							
	0	0	0	1	0	0	1	0
2	**STEREO_EXTENSION RATE =7 (88 bits/frame)**							
	0	0	0	0	0	1	1	1
3	FREQUENCY_SCALE=96							
	0	1	1	0	0	0	0	0
4	AMR-WB+ data (octet 1)							
	b0	b1	b2	b3	b4	b5	b6	b7
5..36	AMR-WB+ data (octets 2 to 33)							
	b8	…	…	…	…	…	…	…
37	AMR-WB+ data (octet 34)							
	b264	b265	b266	b267	b268	b269	b270	b271
38	AMR-WB+ data (octet 35)							
	s0	s1	s2	s3	s4	s5	s6	s7
39..47	AMR-WB+ data (octet 36 to 44)							
	s8	…	…	…	…	…	…	…
48	AMR-WB+ data (octet 45)							
	s80	s81	s82	s83	s84	s85	s86	s87

Annex A (informative): Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
2004-09	SP-25	SP-040640	-	-	Presentation to TSG SA#25	1.0.0	2.0.0