ETSI TS 126 304 V13.0.0 (2016-01)

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Extended Adaptive Multi-Rate - Wideband (AMR-WB+) codec;
Floating-point ANSI-C code
(3GPP TS 26.304 version 13.0.0 Release 13)

Reference
RTS/TSGS-0426304vd00

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

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2016.
All rights reserved.

DECT™, **PLUGTESTS**™, **UMTS**™ and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
3GPP™ and **LTE**™ are Trade Marks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

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

Intellectual Property Rights 2
Foreword 2
Modal verbs terminology 2
Foreword 4
1 Scope 5
2 References 5
3 Definitions, symbols and abbreviations 5
3.1 Definitions 5
3.2 Abbreviations 5
4 C code structure 6
4.1 Contents of the C source code 6
4.2 Program execution 6
4.3 Code hierarchy 6
4.4 Variables, constants and tables 19
4.4.1 Description of fixed tables used in the C-code 19
4.4.2 Static variables used in the C-code 20
5 File formats 24
5.1 Audio file (encoder input/decoder output) 24
5.2 Parameter bitstream file (encoder output/decoder input) 25
Annex A (informative): AMR-WB+ user guide 26
A.1 Encoder usage 26
A.1.1 Simple mode 26
A.1.2 Flexible mode 27
A.2 Decoder usage 31
Annex B (informative): Change history 32
History 33

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 26.274: " Audio codec processing functions; Extended Adaptive Multi-Rate - Wideband (AMR-WB+) codec; Conformance testing ".

[4] 3GPP TS 26.244: "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)"

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.

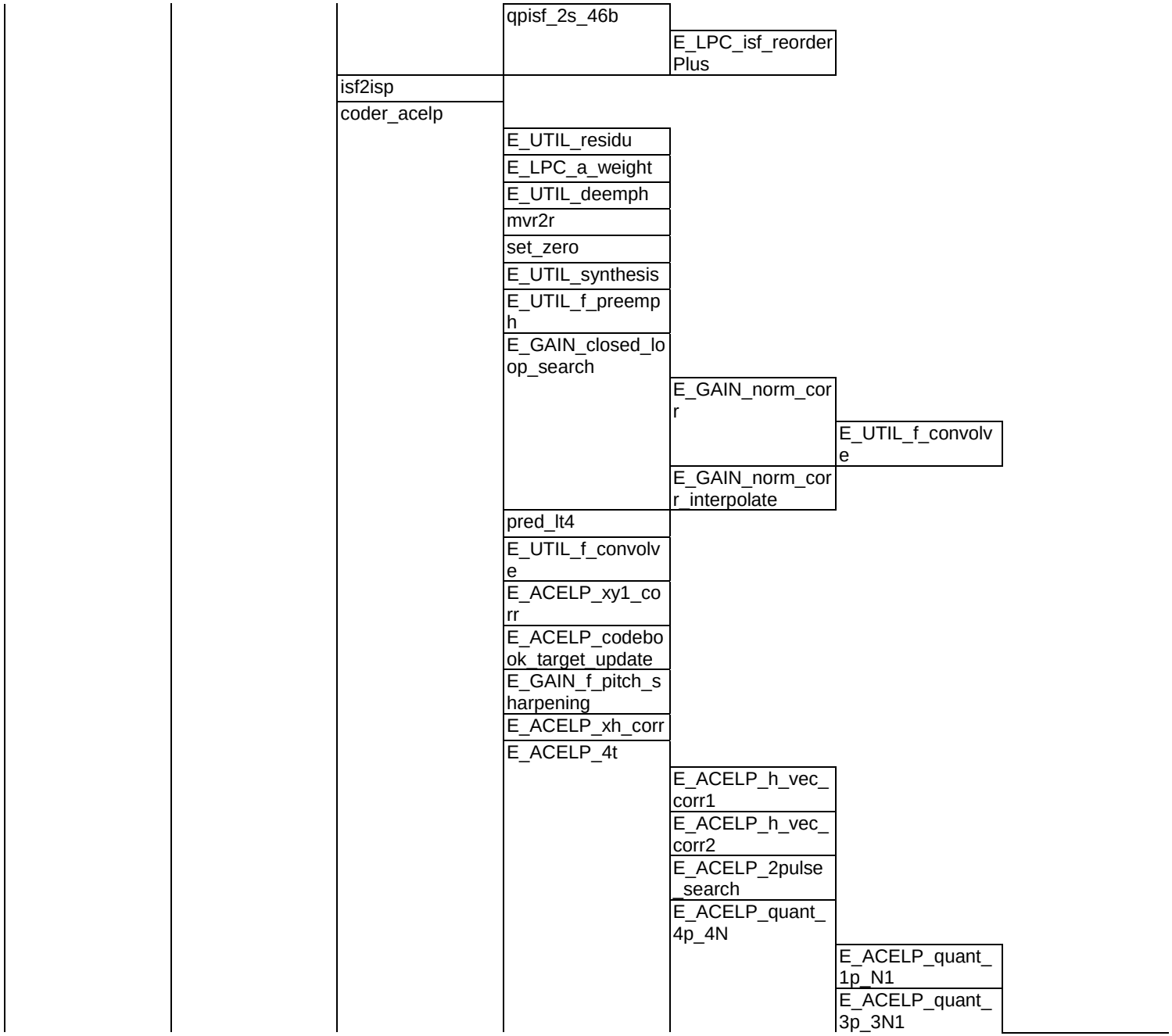
4.3 Code hierarchy

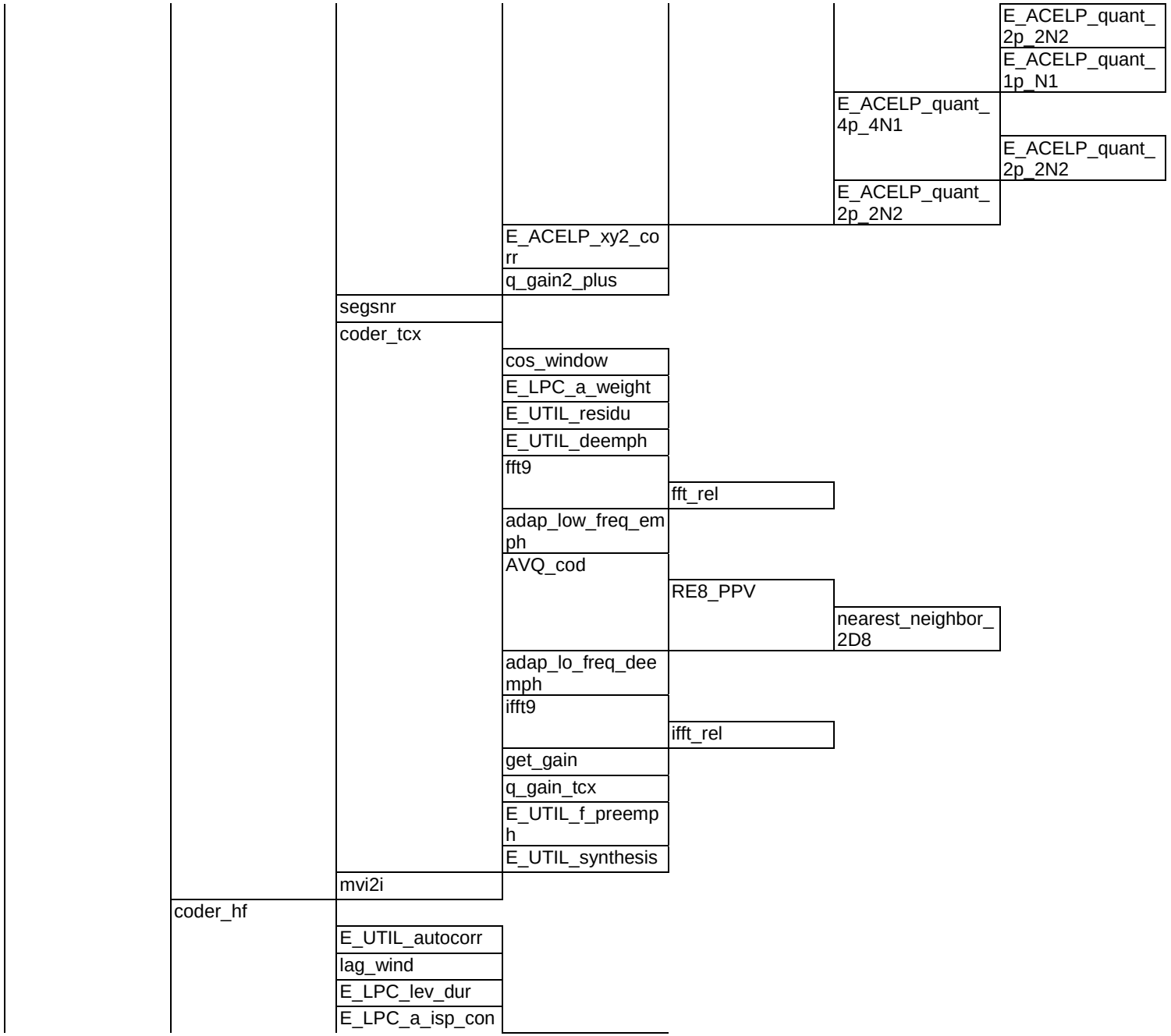
Tables 1 and 2 are call graphs that show the functions used in the audio codec.

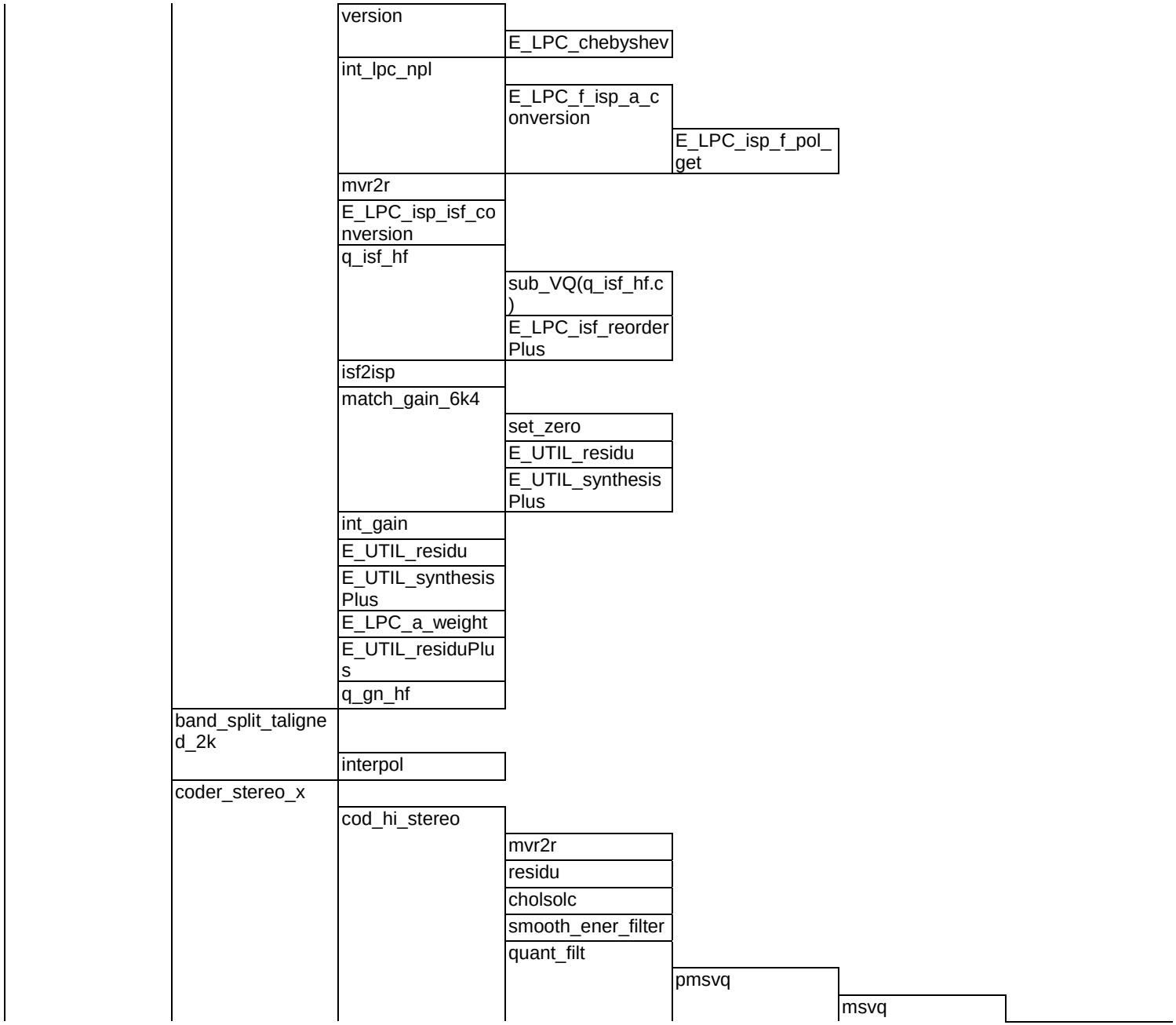
Each column represents a call level and each cell a function. The functions contain calls to the functions in rightwards neighbouring cells. The time order in the call graphs is from the top downwards as the processing of a frame advances. All standard C functions: memcpy(), fwrite(), etc. have been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

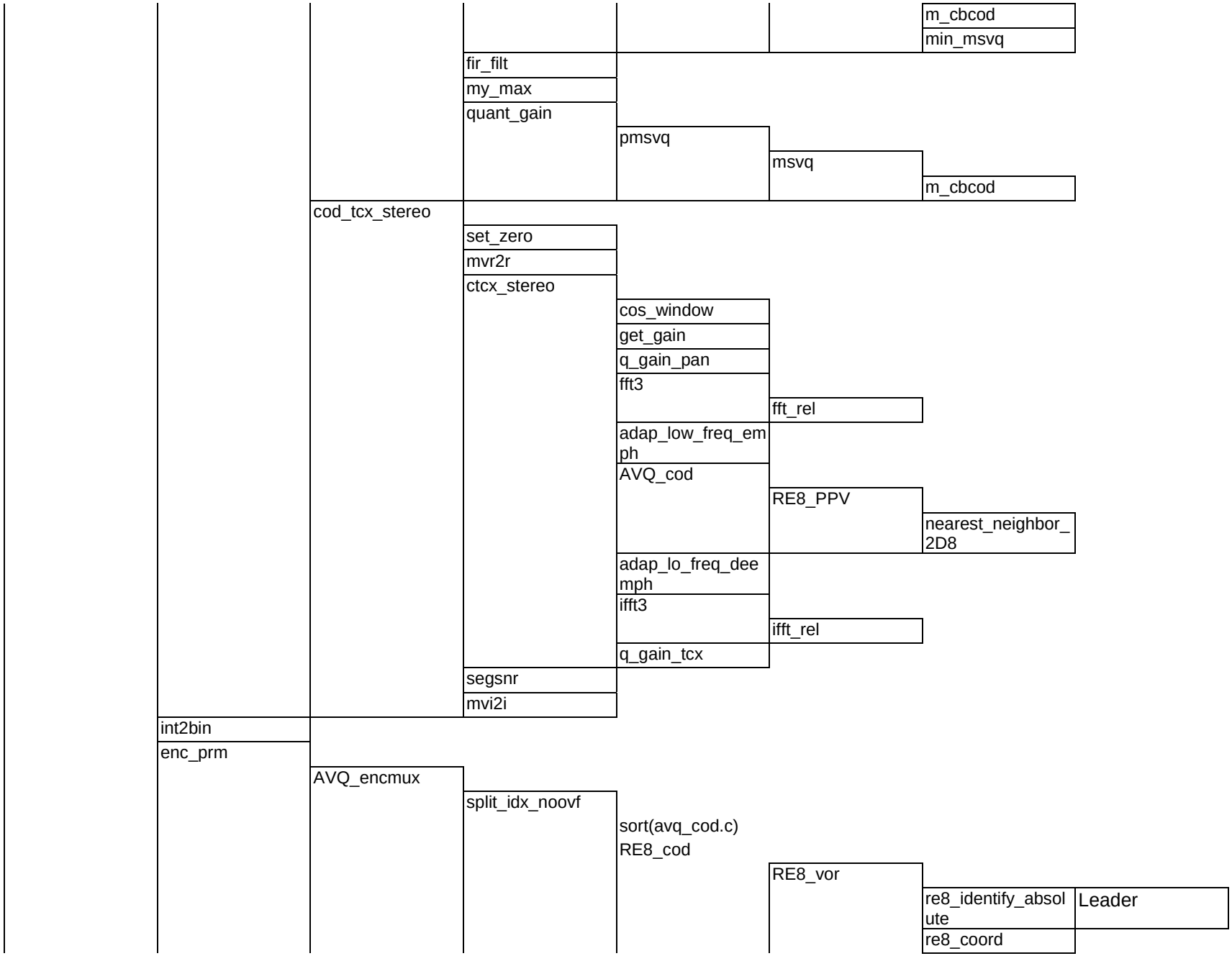
Table 1: Encoder call structure

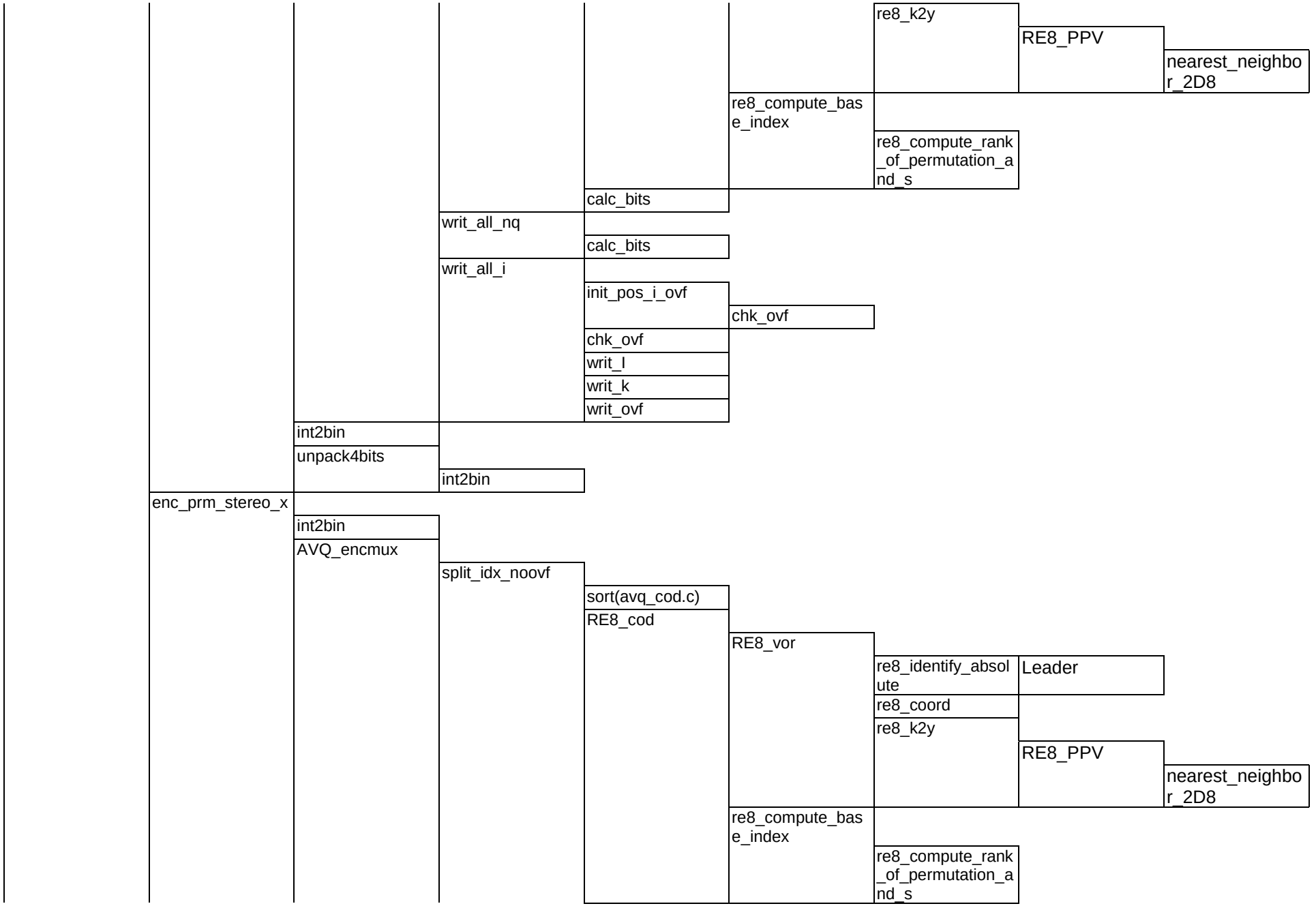
- coder_amrwb_plus_stereo
  - decim_12k8
    - interpol
  - hp50_12k8
  - mix_ch
  - E_UTIL_f_preemph
  - mvr2r
  - coder_lf
    - mvr2r
    - E_UTIL_autocorr
    - lag_wind
    - E_LPC_lev_dur
    - E_LPC_a_isp_conversion
      - E_LPC_chebyshev
    - int_lpc_npl
      - E_LPC_f_isp_a_conversion
        - 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_con


version
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

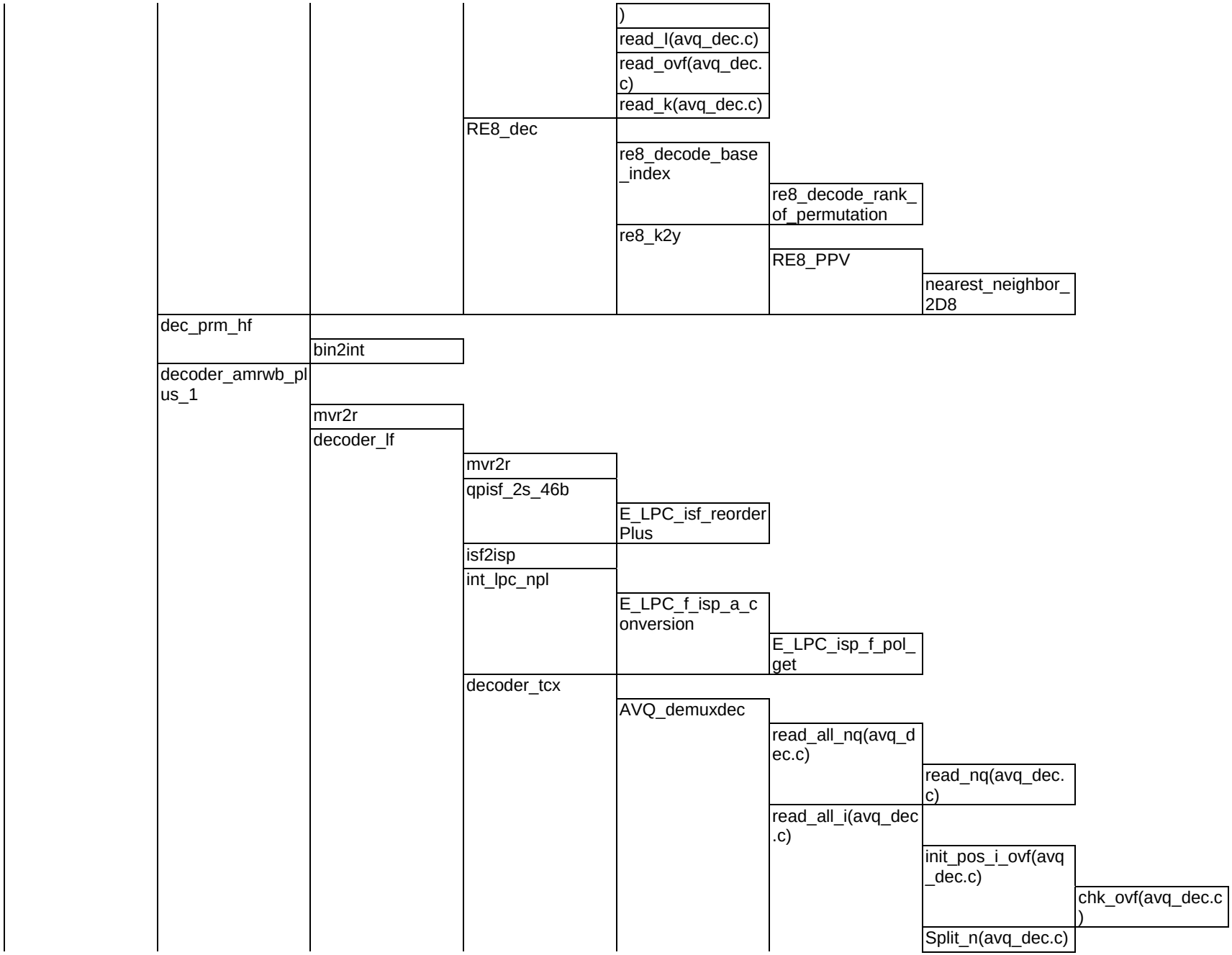

m_cbcod
min_msvq
fir_filt
my_max
quant_gain
pmsvq
msvq
m_cbcod
cod_tcx_stereo
set_zero
mvr2r
ctcx_stereo
cos_window
get_gain
q_gain_pan
fft3
fft_rel
adap_low_freq_emph
AVQ_cod
RE8_PPV
nearest_neighbor_2D8
adap_lo_freq_deemph
ifft3
ifft_rel
q_gain_tcx
segsnr
mvi2i
int2bin
enc_prm
AVQ_encmux
split_idx_noovf
sort(avq_cod.c)
RE8_cod
RE8_vor
re8_identify_absolute
Leader
re8_coord

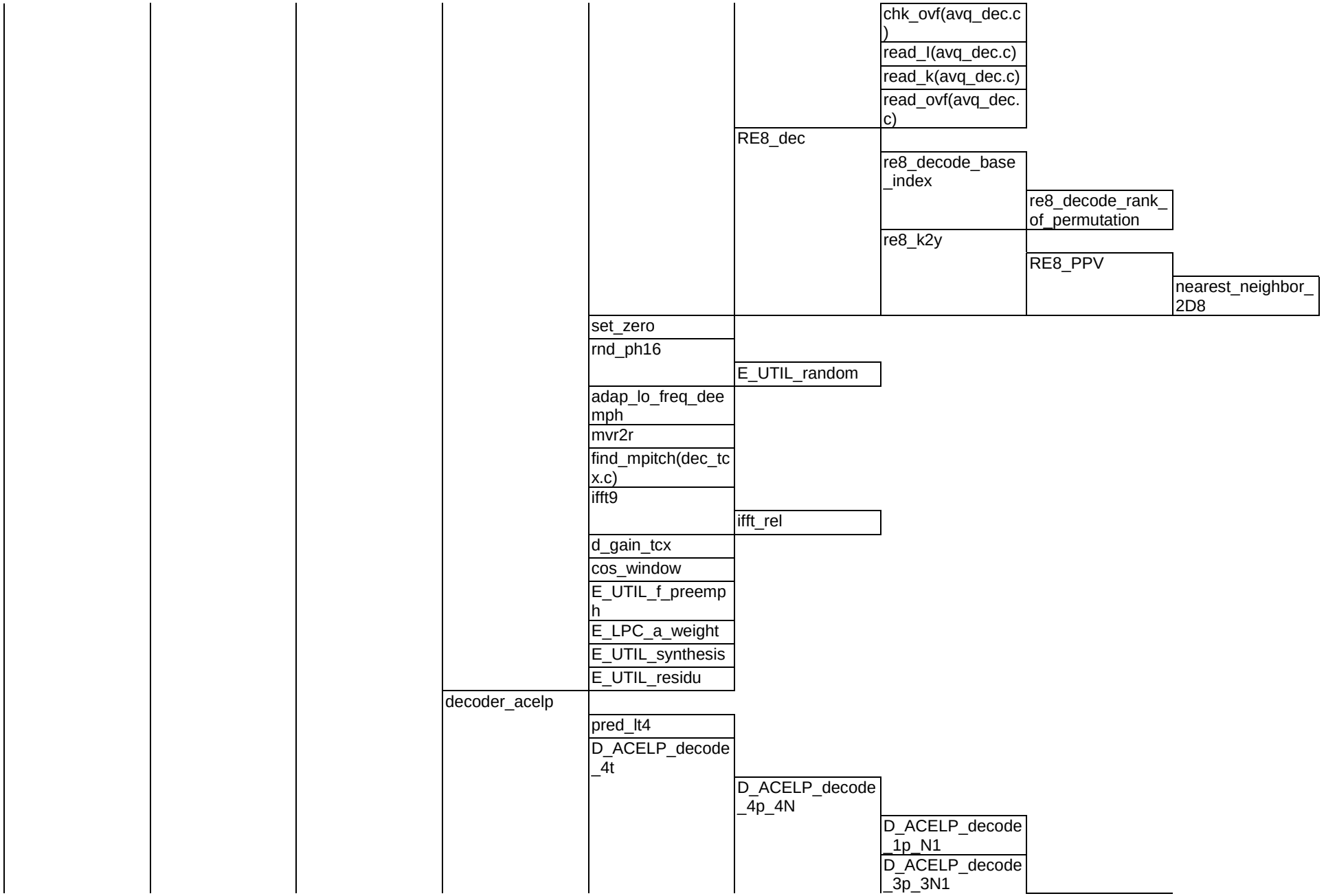

re8_k2y
RE8_PPV
nearest_neighbor_2D8
re8_compute_base_index
re8_compute_rank_of_permutation_and_s
calc_bits
writ_all_nq
calc_bits
writ_all_i
init_pos_i_ovf
chk_ovf
chk_ovf
writ_l
writ_k
writ_ovf
int2bin
unpack4bits
int2bin
enc_prm_stereo_x
int2bin
AVQ_encmux
split_idx_noovf
sort(avq_cod.c)
RE8_cod
RE8_vor
re8_identify_absolute
Leader
re8_coord
re8_k2y
RE8_PPV
nearest_neighbor_2D8
re8_compute_base_index
re8_compute_rank_of_permutation_and_s

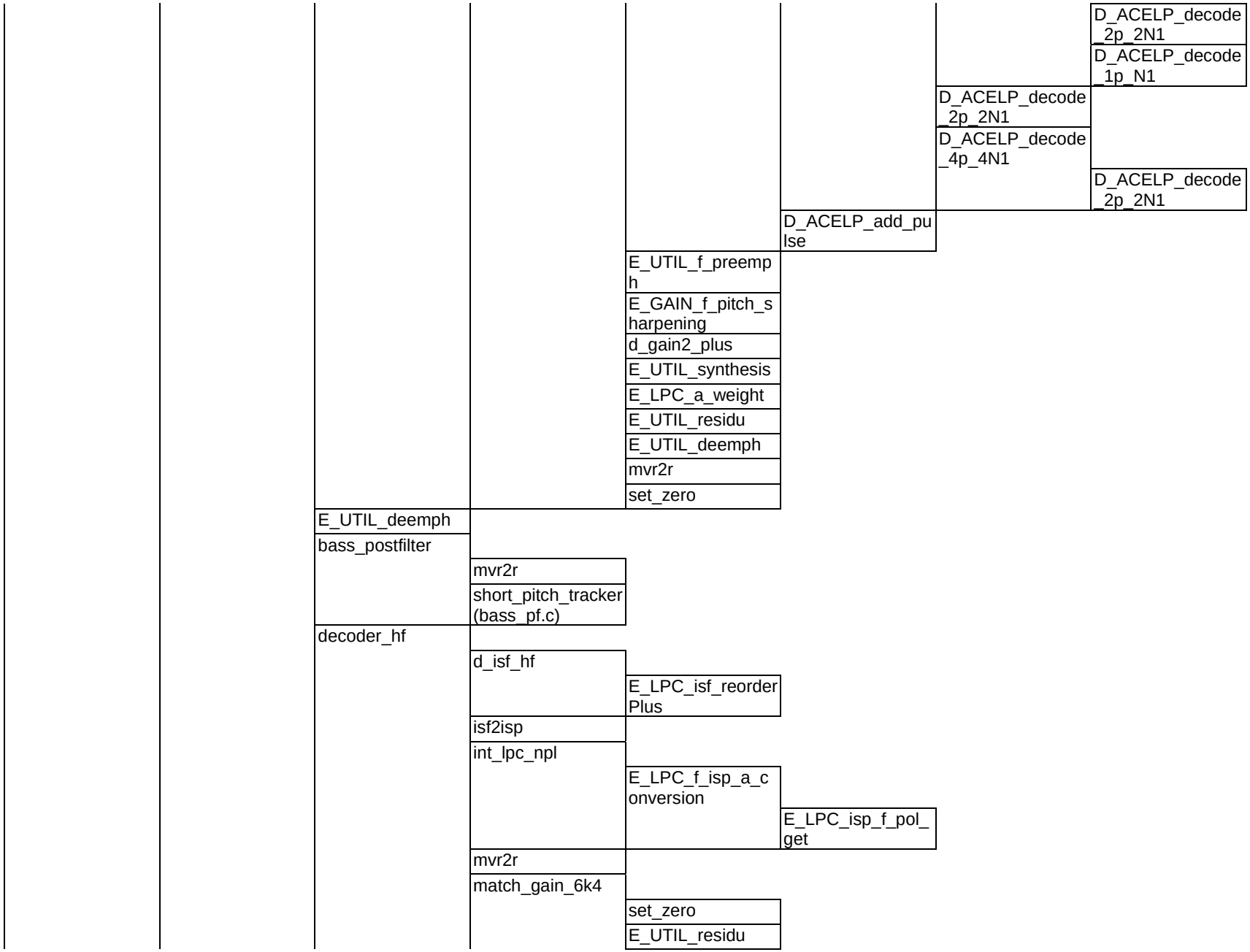
				calc_bits	
			writ_all_nq		
				calc_bits	
			writ_all_i		
				init_pos_i_ovf	
					chk_ovf
				chk_ovf	
				writ_l	
				writ_ovf	
				writ_k	
		unpack4bits_d			
			int2bin		
	enc_prm_hf				
		int2bin			

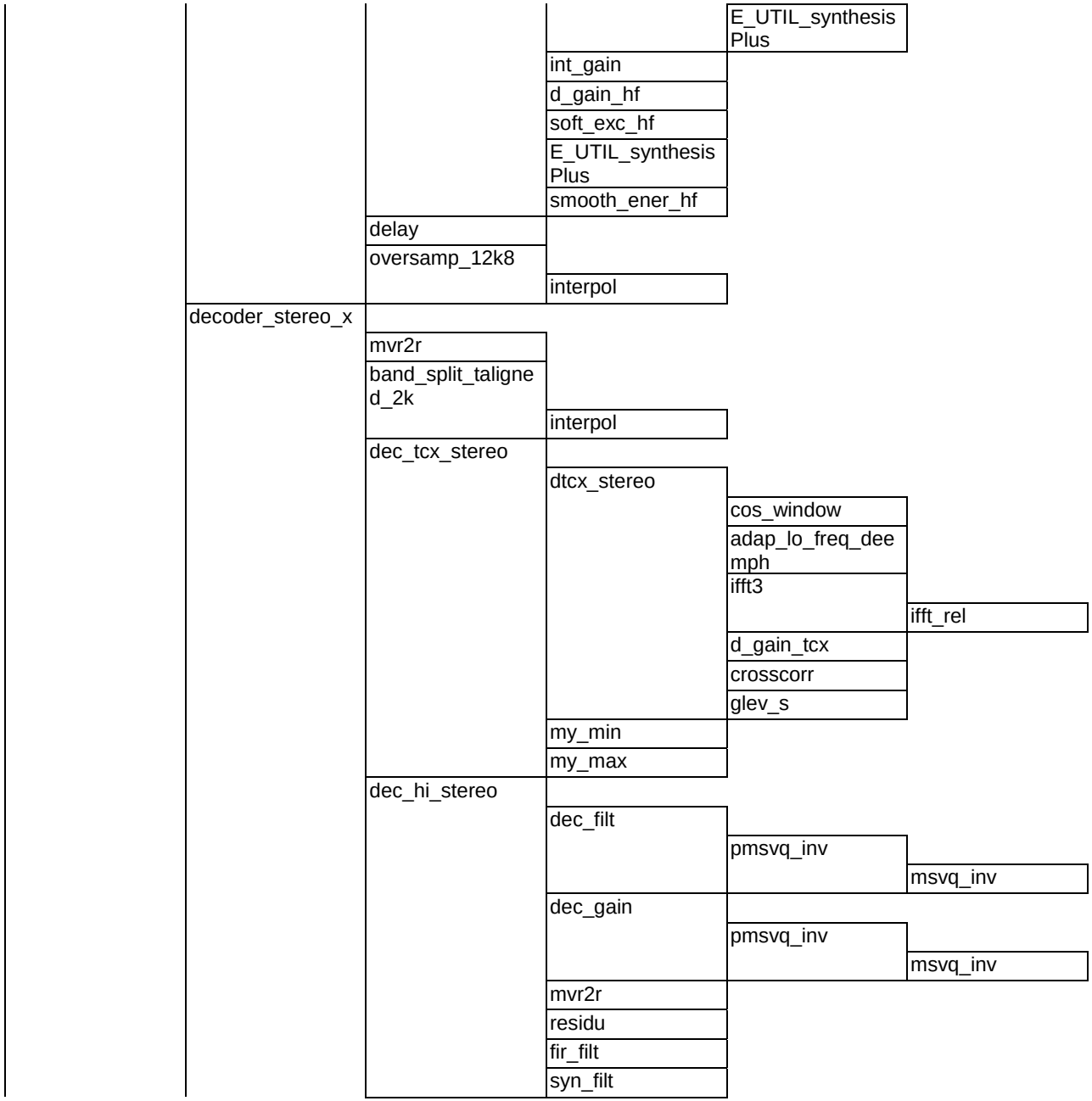
Table 2: Decoder call structure

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_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_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_synthesis Plus
int_gain
d_gain_hf
soft_exc_hf
E_UTIL_synthesis Plus
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_hf_ramp	64	HF gain ramp for wb->wb+ switching
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 TBC	Gain_hi_mscb_2a Gain_hi_mscb_5a	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 are either according to the raw format defined in Reference [2] Section 8.2, or according to the 3GP file format [4], whereby the storage sample definition is found in Reference [2] Section 8.3.

Annex A (informative): AMR-WB+ user guide

This Annex contains a user guide on how to configure AMR-WB+ codec parameters. Due to the codec flexibility, different configurations can be used to operate the codec at a certain bit rate. The aim of this annex is ease the understanding of how to set the parameters of the AMR-WB+ audio codec.

There are two ways to set the AMR-WB+ parameters: in simple mode and flexible ('expert') mode. The simple mode uses a predefined set of configurations at different bit rates. In the flexible mode, the user chooses the core bit rate and the stereo extension rate, in case of stereo operation, and the internal sampling frequency, to attain a certain bit rate.

The AMR-WB+ audio codec processes input frames always equal to 2048 samples at an internal sampling frequency F_s. The internal sampling frequency (ISF) is independent from the audio sampling rate and is limited to the range from 12800-38400 Hz. The 2048-sample frames are split into two critically sampled equal frequency bands. This results in two superframes of 1024 samples corresponding to the low frequency (LF) and high frequency (HF) bands. Each superframe is divided into four 256-sample frames. Because the codec always requires 2048 samples then the input frame size or frame duration will vary for different ISFs.

A.1 Encoder usage

A.1.1 Simple mode

Simple mode is easy to use and requires no knowledge of AMR-WB+ to use the full capacity of the codec. The usage is as follows:

```
AmrwbPlusEncode -rate <bit rate> [-mono] [-ff <3gp|raw>] -if <infile.wav> -of <outfile.wb+>
```

Where

`AmrwbPlusEncode`	Name of the AMR-WB+ encoder program either compiled from the floating-point C-code of this specification or from the fixed-point C-code of [1].
`-rate`	Bit rate from 6-36 kbps for mono encoding or 7-48 kbps for stereo encoding
`-mono`	Forces mono encoding for stereo inputs. If this option is not used the encoder performs mono encoding for mono WAV files and stereo encoding for stereo WAV files.
`-ff`	File format: 3gp raw The default is 3gp file format.
`-if`	Input audio WAV file with one (mono) or two (stereo) channels Supported audio sampling rates are 8, 16, 24, 32, 48, 11.025, 22.05, 44.1 kHz
`-of`	Output file (according to the -ff argument)

The codec will use the best combination of mono and stereo bit rates and internal sampling frequency (ISF) according to the bit rate specified by the user. In this simple mode, the codec uses a set of predefined configurations in the bit rate range from 6-48 kbps. The codec will choose the closest configuration to the required bit rates. The configurations have been chosen to optimize the quality/bandwidth trade-off at a certain bit rate for 48 kHz sampled input.

Tables A.1 and A.2 show the default codec configurations for mono and stereo operation, respectively. In the tables, the mapping from selected bit rate to mode index and ISF index are given, and in addition the resulting bit rate factor and coded audio bandwidth (BW). The mode index (or frame type) is defined and explained in Table 25 of 26.290 [2] (modes 16 to 23 are mono modes and modes 24 to 47 are stereo modes). The mapping from ISF index to ISF, corresponding frame size and bit rate factor is defined in Table 24 of 26.290 [2].

Table A.1 Mono default configurations

bit rate (kbps)	Mode Index (bit rate at nominal ISF of 25.6 kHz)	ISF index (sampling frequency)	Bit rate factor	BW (kHz)
5.85	16 (10.4 kbps)	2 (14.4 kHz)	9/16	7.2
6.933	16 (10.4 kbps)	4 (17.067)	2/3	8.533
7.8	16 (10.4 kbps)	5 (19.2 kHz)	3/4	9.6
8.667	16 (10.4 kbps)	6 (21.33 kHz)	5/6	10.67
9.75	16 (10.4 kbps)	7 (24 kHz)	15/16	12
11.25	17 (12 kbps)	7 (24 kHz)	15/16	12
12	17 (12 kbps)	8 (25.6 kHz)	1	12.8
13.6	18 (13.6 kbps)	8 (25.6 kHz)	1	12.8
15.2	19 (15.2 kbps)	8 (25.6 kHz)	1	12.8
17.1	19 (15.2 kbps)	9 (28.8 kHz)	9/8	14.4
18.9	20 (16.8 kbps)	9 (28.8 kHz)	9/8	14.4
21.6	21 (19.2 kbps)	9 (28.8 kHz)	9/8	14.4
24	21 (19.2 kbps)	10 (32 kHz)	5/4	16
26	22 (20.8 kbps)	10 (32 kHz)	5/4	16
30	23 (24 kbps)	10 (32 kHz)	5/4	16
32	23 (24 kbps)	11 (34.13 kHz)	4/3	17.06
33.75	23 (24 kbps)	12 (36 kHz)	45/32	18
36	23 (24 kbps)	13 (38.4 kHz)	3/2	19.2

Table A.2: Stereo default configurations

bit rate (kbps)	Mode Index (bit rate at nominal ISF of 25.6 kHz)	ISF index (sampling frequency)	Bit rate factor	BW (kHz)
6.975	24 (12.4 kbps)	2 (14.4 kHz)	9/16	7.2
8.267	24 (12.4 kbps)	4 (17.067 kHz)	2/3	8.533
9.3	24 (12.4 kbps)	5 (19.2 kHz)	3/4	9.6
10.33	24 (12.4 kbps)	6 (21.33 kHz)	5/6	10.67
11.65	24 (12.4 kbps)	7 (24 kHz)	15/16	12
13.125	26 (14 kbps)	7 (24 kHz)	15/16	12
14.25	28 (15.2 kbps)	7 (24 kHz)	15/16	12
15	29 (16 kbps)	7 (24 kHz)	15/16	12
16	29 (16 kbps)	8 (25.6 kHz)	1	12.8
17.2	31 (17.2 kbps)	8 (25.6 kHz)	1	12.8
18	32 (18 kbps)	8 (25.6 kHz)	1	12.8
19.2	34 (19.2 kbps)	8 (25.6 kHz)	1	12.8
20	35 (20 kbps)	8 (25.6 kHz)	1	12.8
22.5	35 (20 kbps)	9 (28.8 kHz)	9/8	14.4
23.85	37 (21.2 kbps)	9 (28.8 kHz)	9/8	14.4
25.2	38 (22.4 kbps)	9 (28.8 kHz)	9/8	14.4
27	40 (24 kbps)	9 (28.8 kHz)	9/8	14.4
30	40 (24 kbps)	10 (32 kHz)	5/4	16
32	40 (24 kbps)	11 (34.13 kHz)	4/3	17.06
34.13	41 (25.6 kbps)	11 (34.13 kHz)	4/3	17.06
36	41 (25.6 kbps)	12 (36 kHz)	45/32	18
37.6875	43 (26.8 kbps)	12 (36 kHz)	45/32	18
40.2	43 (26.8 kbps)	13 (38.4 kHz)	3/2	19.2
43.2	44 (28.8 kbps)	13 (38.4 kHz)	3/2	19.2
45	46 (30 kbps)	13 (38.4 kHz)	3/2	19.2
48	47 (32 kbps)	13 (38.4 kHz)	3/2	19.2

A.1.2 Flexible mode

Flexible ('expert') mode requires more knowledge of AMR-WB+ to use the full capacity of the codec. In this mode, the user can choose the core bit rate and the stereo extension rate, in case of stereo operation, and the internal sampling frequency, to attain a certain bit rate.

The usage is as follows:

```
AmrwbPlusEncode -mi <mode> [-isf <factor>] [-lc] [-dtx] [-ff <3gp/raw>]
–if <infile.wav> -of <outfile.wb+> [-cf <configfile.txt>]
```

Where

`AmrwbPlusEncode`	Name of the AMR-WB+ encoder program either compiled from the floating-point C-code of this specification or from the fixed-point C-code of [1].
`-mi`	Mode Index 0..8 AMR-WB 10..13 AMR-WB+ special modes 16..23 AMR-WB+ mono modes 24..47 AMR-WB+ stereo modes
`-isf`	Internal Sampling Frequency factor 0.5..1.5 If this option is missing, the default is 1.0
`-lc`	Low complexity (for AMR-WB+ modes) If this option is missing, the default is to use normal encoding. **Note:** This option is designed for terminal-based encoding. For optimal quality, it is recommended to not use this parameter.
`-dtx`	Enables VAD/DTX functionality (only for AMR-WB modes) If this option is missing, the default is NO DTX
`-ff`	File format: 3gp raw If this option is missing, the default is 3gp file format.
`-if`	Input audio WAV file Supported audio sampling rates are 8, 16, 24, 32, 48, 11.025, 22.05, 44.1 kHz Modes 0..8 require 16 kHz input audio sampling rate. Modes 10..13 require 16 or 24 kHz audio sampling rate.
`-of`	Output file (according to the -ff argument)
`-cf`	Configuration file: an auxiliary file that can be used for bit rate switching

The mode index (`mi`) can be found in Tables 21 and 25 of 26.290 [2] and the Internal Sampling Frequency (`ISF`) in Table 24 of 26.290 [2].

Table 21 of 26.290 [2] contains the AMR-WB–compatible modes and four AMR-WB+ special modes (mode index 10-13). The AMR-WB+ special modes have a fixed ISF (ISF index = 0 from Table 24 of 26.290 [2]). The codec can switch dynamically between the AMR-WB and AMR-WB+ special modes (Table 21 of 26.290 [2]) if AMR-WB+ is operated at 16 kHz.

Table 25 of 26.290 [2] contains the AMR-WB+ mono and stereo modes. The core and stereo mode indices of Table 25 of 26.290 [2] correspond to Tables 22 and 23 of 26.290 [2]. The bit rates specified in Table 25 of 26.290 [2] are for a nominal ISF of 25600 Hz (bit rate factor = 1.0). The output bit rate can be computed by multiplying the bit rate value from Table 25 of 26.290 [2] and the bit rate factor from Table 24 of 26.290 [2].

Stereo flexibility

In case of stereo operation, the flexible mode provides some degree of flexibility for trade-off between mono and stereo extension bit rates. This can be content dependent where higher or lower stereo extension bit rates can be used depending on the correlation between the two channels. In Table 25 of 26.290 [2], there are 24 stereo modes (mode

indices 24 to 47). These modes correspond to the 8 core mono modes where 3 different extension rates are combined with each core mode. For example, mode indices 39, 40, and 41 correspond to a core bit rate of 19.2 kbps combined with stereo extension rates of 4.0, 4.8, and 6.4 kbps, respectively. This results in total bit rates of 23.2, 24.0, and 25.6 kbps. In these stereo modes, the ratio between the stereo extension rate and the total bit rate is 17.1%, 20%, and 25%, respectively.

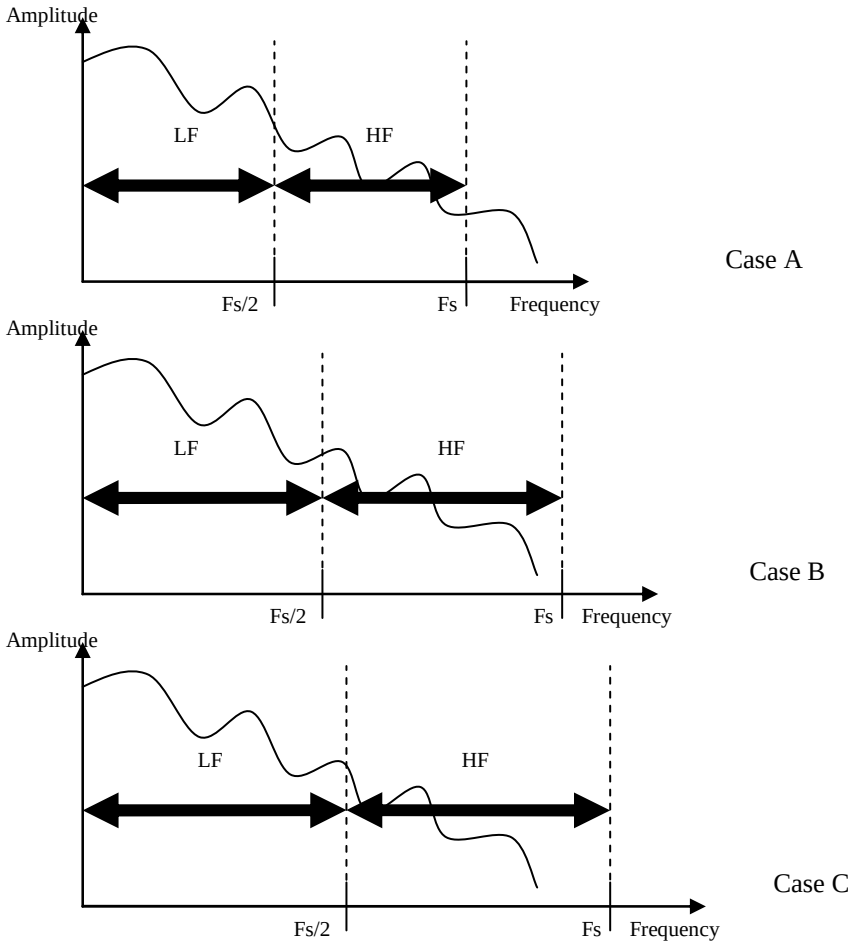
Choosing encoded bandwidth

The flexibility in choosing the ISF gives the user the choice to adjust the coded audio bandwidth depending on the input signal. For instance, in case of speech signals, a bandwidth up to 14 kHz is sufficient to attain transparent quality.

To determine the AMR-WB+ mode, one approach could be to first determine the bandwidth of the signal that required to be encoded. Table 24 of 26.290 [2] can be used to choose the ISF. Then the core and stereo rates are chosen from Tables 25, 22, and 23 of 26.290 [2]. (Note that these bit rates should be scaled with the bit rate factor from Table 24 of 26.290 [2].)

Further tuning can be done by adjusting the ISF. The HF encoding uses relatively few bits compared to the LF; therefore, the LF part of the signal has a higher definition than the HF. Therefore, increasing the ISF can be considered even in cases where the resulting bandwidth might exceed the input signal bandwidth if the bit budget allows it.

The graph below explains the three different possibilities.



Case A is when the ISF is set to be smaller than the signal spectrum. This can be used when the bit rate or the CPU load has to be reduced.

Case B is probably the most usual situation. The signal spectrum is matched with the ISF.

Case C can be used when high quality is required and an adequate bit rate budget is available. In this case, the HF exceeds the signal bandwidth, but the bits allocated for the HF will be used to encode the active part only. For example, this case can be used to encode a signal with input bandwidth limited to 14 kHz at a bit rate of 36 kbps. If we use mode index 36 (24 kbps at nominal ISF) and ISF of 38.4 kHz (bit rate factor 1.5), the resulting bit rate will be 36 kbps with the LF encoded up to 9.6 kHz and the HF from 9.6 to 14 kHz.

Bit rate switching using a configuration file

Bit rate switching can be simulated using an auxiliary configuration file. The option -cf refers to a text file that allows for changing Mode Index and ISF dynamically during a program run. The configuration file contains a time reference, a specific extension (AMR-WB or AMR-WB+), mode index (mi), and an internal sampling frequency (ISF), used to encode at that specific time. The encoder keeps the last setting to encode the remaining part of the file. To use -cf option to switch between AMR-WB and AMR-WB+ special modes, input files at 16 kHz sampling rate need to be used and the decoder needs to use the option -fs 16000 (see Section A.2 below).

Each configuration file consists of 4 columns consisting of "time" "ext" "mode_index" "fscale".

"time" is specified in seconds and must always be > 0.

"extension" is 0 or 1 for choosing AMR-WB or AMR-WB+ modes.

"mi" = [0..47], where [0..15] is AMR-WB and AMR-WB+ special modes, and [16..47] is for AMR-WB+ extension modes

"isf"= [0.5..1.5] for AMR-WB+ extension modes, and represent the bit rate factor. 'isf' is set to zero for mode indices 0 to 15.

The following is an example of a configuration file for switching between AMR-WB and AMR-WB+ special modes.

#time	Extension	mi	isf
0.08	1	10	0.0
1.08	0	7	0.0
2.08	1	10	0.0
3.08	0	0	0.0

Here, 'time' is in seconds, and 'extension'=0 means AMR-WB and 'extension'=1 means AMR-WB+. The value of 'isf' for modes 0-15 must be zero. In this example, the encoder will use the initial configuration up to first 80 ms. At 80 ms it will start encoding with AMR-WB+ mode 10 (13.6 kbps). At time instant 1080 ms, it will start encoding with AMR-WB mode 7 (23.05 kbps). At time instant 2080 ms, it will start encoding with AMR-WB+ mode 10. Finally, at time instant 3080 ms, it will start encoding with AMR-WB mode 0 (6.6 kbps) till the end of the file. This can be seen as using the initial configuration for the first 80 ms, using mode 10 for 1 second, using mode 7 for 1 second, using mode 10 for 1 second, and using mode 0 for the remaining of the file.

In the above example, if the following command line is used:

```
AmrwbPlusEncode -ff raw -mi 12 -cf switch_amrwb.txt -if Input.wav -of bit_stream
```

then the first 80 ms will be encoded with AMR-WB+ mode 12.

The example below shows a configuration file where mode index and ISF are switched.

#time	Extension	mi	isf
0.5	1	16	1.0
1.0	1	20	1.5
2.0	1	23	1.5
3.0	1	35	0.8
4.0	1	40	1.0
5.0	1	47	1.5
10.0	1	23	1.0

A.2 Decoder usage

The decoder usage is as follows:

```
AmrwbPlusDecode [-ff <3gp/raw>] [-fs <fs_Hz>][-mono] [-limiter] -if
<infile.wb+> -of <outfile.wav> [-fer <error.txt>]
```

Where

`AmrwbPlusDecode`	Name of the AMR-WB+ decoder executable either compiled from the floating-point C-code of this specification or from the fixed-point C-code of [1].
`-ff`	File format: 3gp raw The default is 3gp file format.
`-fs`	Output sampling rate (if this option is missing the default value of 48 kHz is used)
`-mono`	Forces mono decoding for stereo encoded inputs.
`-limiter`	To avoid output clipping (recommended)
`-if`	Input AMR-WB+ bitstream file (according to the -ff argument)
`-of`	Output audio WAV file
`-fer`	Frame erasure file for simulating frame erasures.

The output sampling rate can be chosen in function of the hardware. The decoder is able to support 8, 16, 24, 32 and 48kHz or 11.025, 22.050 and 44.1kHz. The sampling rate cannot be changed at run time. The default value is 48 kHz which is recommended. If the decoder receives modes 0 to 15 the default sampling frequency is 16 kHz.

The limiter option will increase quality when the signal contains saturations and will not degrade otherwise. It is recommended to always activate the limiter option.

Frame erasure text files can be used for simulating frame erasures. They are constructed as text files (ASCII) with one flag "0" or "1" per line. There is one line per codec (transmission) frame. "0" indicates proper reception of the frame and "1" a frame erasure.

Annex B (informative): Change history

Change history							
Date	**TSG SA#**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2004-09	25	SP-040640	-	-	Approved at TSG SA#25	2.0.0	6.0.0
2004-12	26	SP-040841	001		Incorrect definition of mode index for SID frames	6.0.0	6.1.0
2004-12	26	SP-040841	002		Correction of TCX coding selection for MMS encoder	6.0.0	6.1.0
2004-12	26	SP-040841	003		Misread of energy buffer in coding mode selection in MMS encoder. Correction of energy buffer initialisation	6.0.0	6.1.0
2004-12	26	SP-040841	004		Correction of stereo bit allocation tables	6.0.0	6.1.0
2004-12	26	SP-040841	005	1	Optimization of error concealment operation	6.0.0	6.1.0
2004-12	26	SP-040841	006	1	Stereo operation of pre-echo mode, saturation of gain_shape	6.0.0	6.1.0
2004-12	26	SP-040841	007		Stereo operation of pre-echo mode, alignment of encoder and decoder	6.0.0	6.1.0
2004-12	26	SP-040841	008	1	Addition of support for file formats and improved command line	6.0.0	6.1.0
2004-12	26	SP-040841	009	1	Source code editorial changes	6.0.0	6.1.0
2004-12	26	SP-040841	010		Removal of complexity counters	6.0.0	6.1.0
2004-12	26	SP-040841	011	1	Editorial changes	6.0.0	6.1.0
2004-12	26	SP-040841	012		Void. This CR in S4-040722 (Title: Editorial changes) was meant to TS 26.290 (as CR 004) and not to TS 26.304 ! The CR was implemented instead in TS 26.290 v. 6.1.0 and a remark was put in the CR database.	6.0.0	6.1.0
2004-12	26	SP-040841	013		Removal of the eid tool	6.0.0	6.1.0
2004-12	26	SP-040841	014	1	Addition of frame erasure simulation at the decoder	6.0.0	6.1.0
2004-12	26	SP-040841	015		Removal of two unused stereo rates	6.0.0	6.1.0
2004-12	26	SP-040841	016		Removal of extStMode	6.0.0	6.1.0
2005-03	27	SP-050096	019	1	AMR-WB/AMR-WB+ switching	6.1.0	6.2.0
2005-03	27	SP-050096	020	2	Clean-up of unused C-code functions	6.1.0	6.2.0
2005-03	27	SP-050096	021	1	Correction of misbehaviour of constrained cholesky	6.1.0	6.2.0
2005-03	27	SP-050096	022	1	Source code bit exact editorial changes	6.1.0	6.2.0
2005-03	27	SP-050096	023	2	Correction of last frame processing	6.1.0	6.2.0
2005-03	27	SP-050096	024	1	Correction of frame erasure concealment	6.1.0	6.2.0
2005-03	27	SP-050096	025	1	Correction of references and terminology	6.1.0	6.2.0
2005-06	28	SP-050252	026		Correction of DTX handling in AMR-WB modes	6.2.0	6.3.0
2005-06	28	SP-050252	027		Remove IF2 header in AMR-WB bitstream	6.2.0	6.3.0
2005-06	28	SP-050252	028		Decoder synchronization after frame erasures	6.2.0	6.3.0
2005-06	28	SP-050252	029		Correction for buffer reading in low complexity encoder	6.2.0	6.3.0
2005-06	28	SP-050252	030		Correction to a wrong function call	6.2.0	6.3.0
2005-06	28	SP-050252	031		Correction of mode switching using configuration file	6.2.0	6.3.0
2005-06	28	SP-050252	032		Correction of information printed by decoder in DTX frames	6.2.0	6.3.0
2005-06	28	SP-050252	033		Correction of library function for 3GP file format	6.2.0	6.3.0
2005-06	28	SP-050252	034		Support for input files with sampling frequency other than 48 kHz	6.2.0	6.3.0
2005-09	29	SP-050425	0035		Correction to frame erasure concealment	6.3.0	6.4.0
2006-03	31	SP-060012	0036		Correction to end-of-file logic and initialisation in AMR-WB modes	6.4.0	6.5.0
2006-03	31	SP-060012	0037		Correction to unnecessary look ahead in encoder	6.4.0	6.5.0
2006-03	31	SP-060012	0038		Correction to memory initialization and memory overwrite when switching between AMR-WB and AMR-WB+ modes	6.4.0	6.5.0
2006-03	31	SP-060012	0039		Correction to VC 6.0 compilation warnings	6.4.0	6.5.0
2006-06	32	SP-060353	0040		Correction to switching between AMR-WB and AMR-WB+ modes	6.5.0	6.6.0
2006-06	32	SP-060353	0041		Correction to default stereo codec configurations	6.5.0	6.6.0
2007-03	35	SP-070029	0042		Reference to users guide	6.6.0	7.0.0
2008-12	42				Version for Release 8	7.0.0	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
2012-10					Correction of Change History	11.0.0	11.0.1
2014-09	65				Version for Release 12	11.0.1	12.0.0
2015-12	70				Version for Release 13	12.0.0	13.0.0

History

Document history		
V13.0.0	January 2016	Publication