

3GPP TS 26.304 V1.0.0 (2004-06)

Technical Specification

**3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
ANSI-C code for the Floating-point;
Extended AMR Wideband codec
(Release 6)**



The present document has been developed within the 3rd Generation Partnership Project (3GPPTM) and may be further elaborated for the purposes of 3GPP.

The present document has not been subject to any approval process by the 3GPP Organizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPP only. The Organizational Partners accept no liability for any use of this Specification. Specifications and reports for implementation of the 3GPPTM system should be obtained via the 3GPP Organizational Partners' Publications Offices.

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>

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© 2004, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TTA, TTC).
All rights reserved.

Contents

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)	24
Annex A (informative): Change history	25

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".

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.

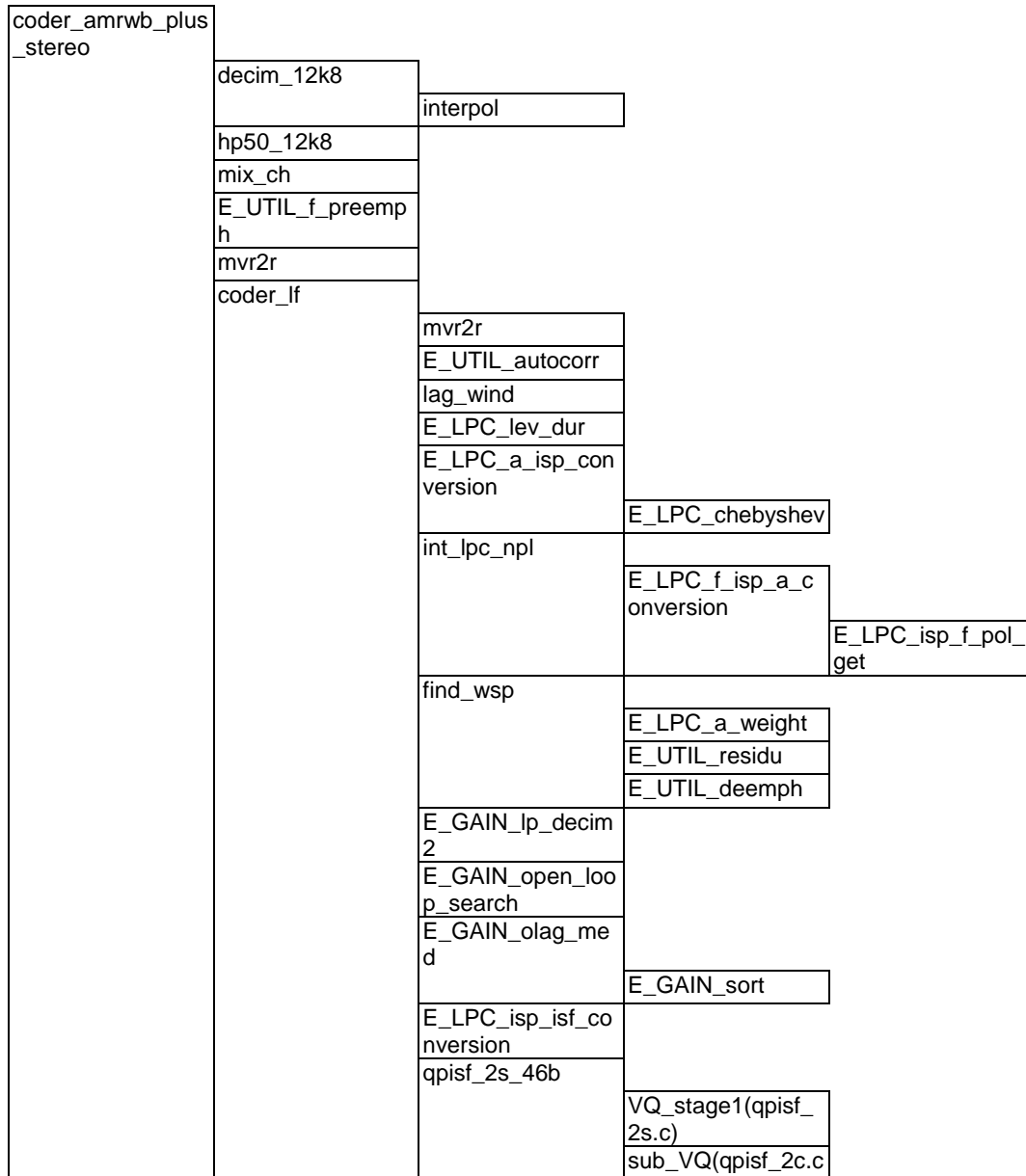
The encoder and decoder options will be explained by running the applications without input arguments. See the file *readme.txt* for more information on how to run the *encoder* and *decoder* programs.

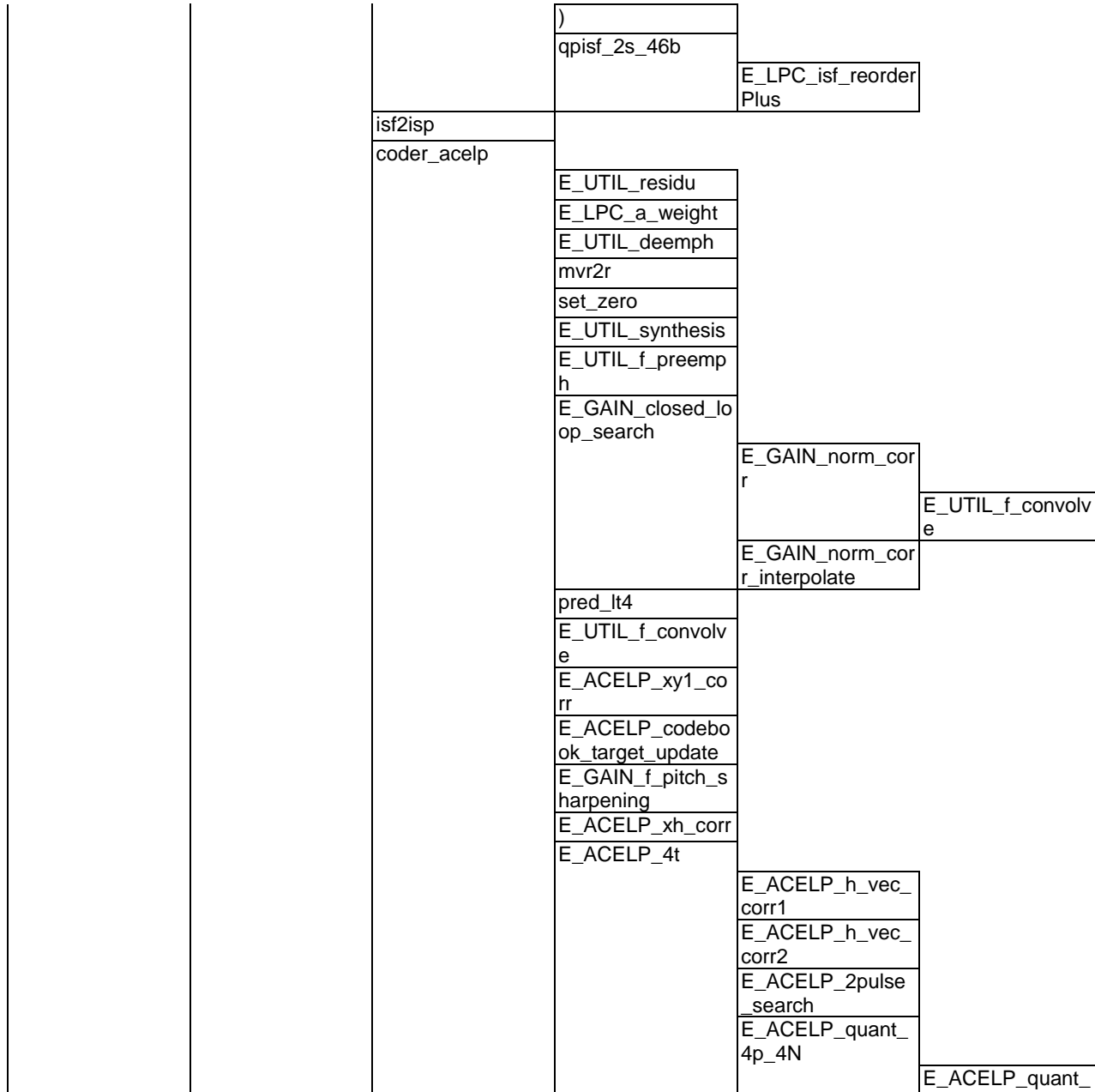
4.3 Code hierarchy

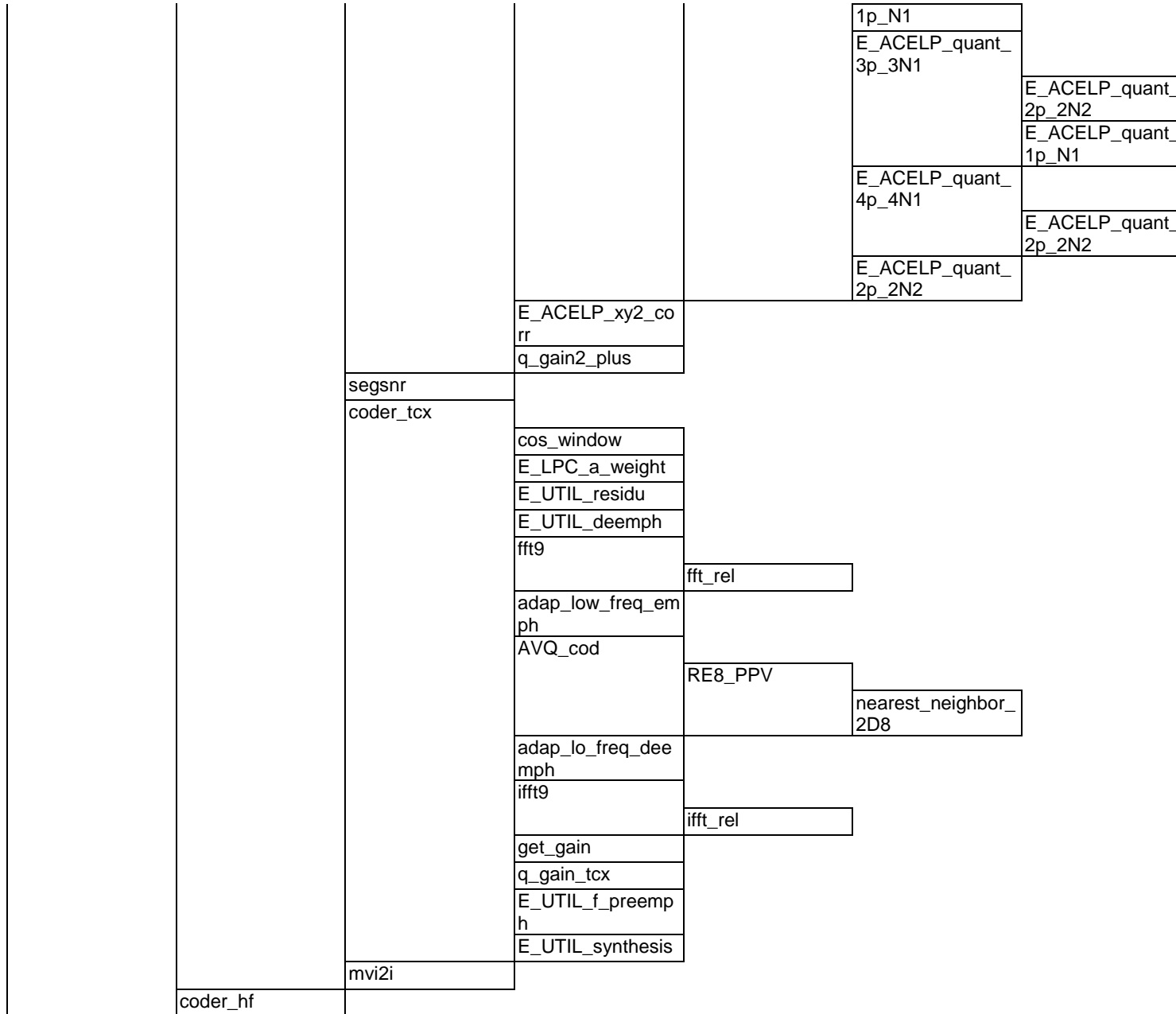
Tables 1 and 2 are call graphs that show the functions used in the audio codec.

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.

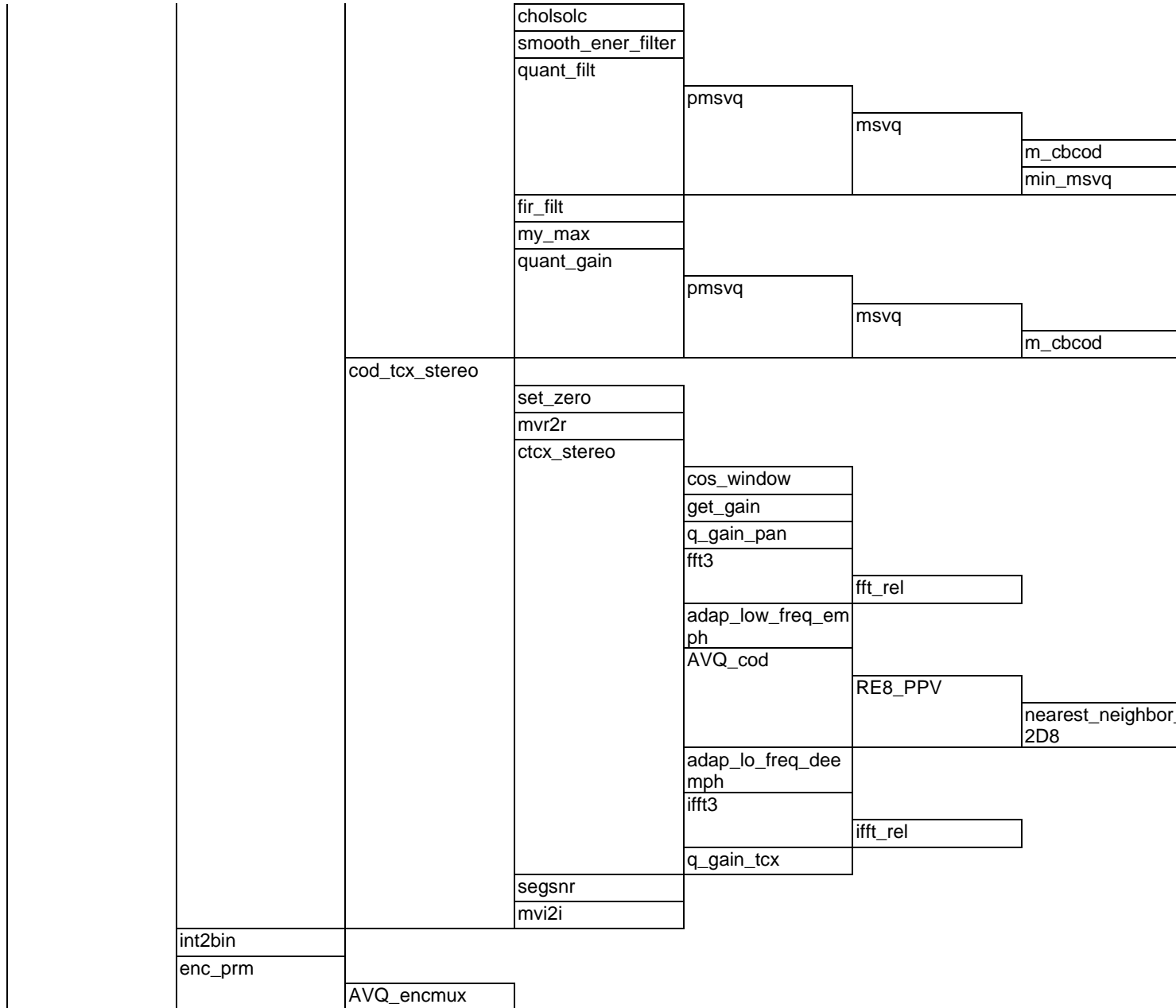
Table 1: Encoder call structure

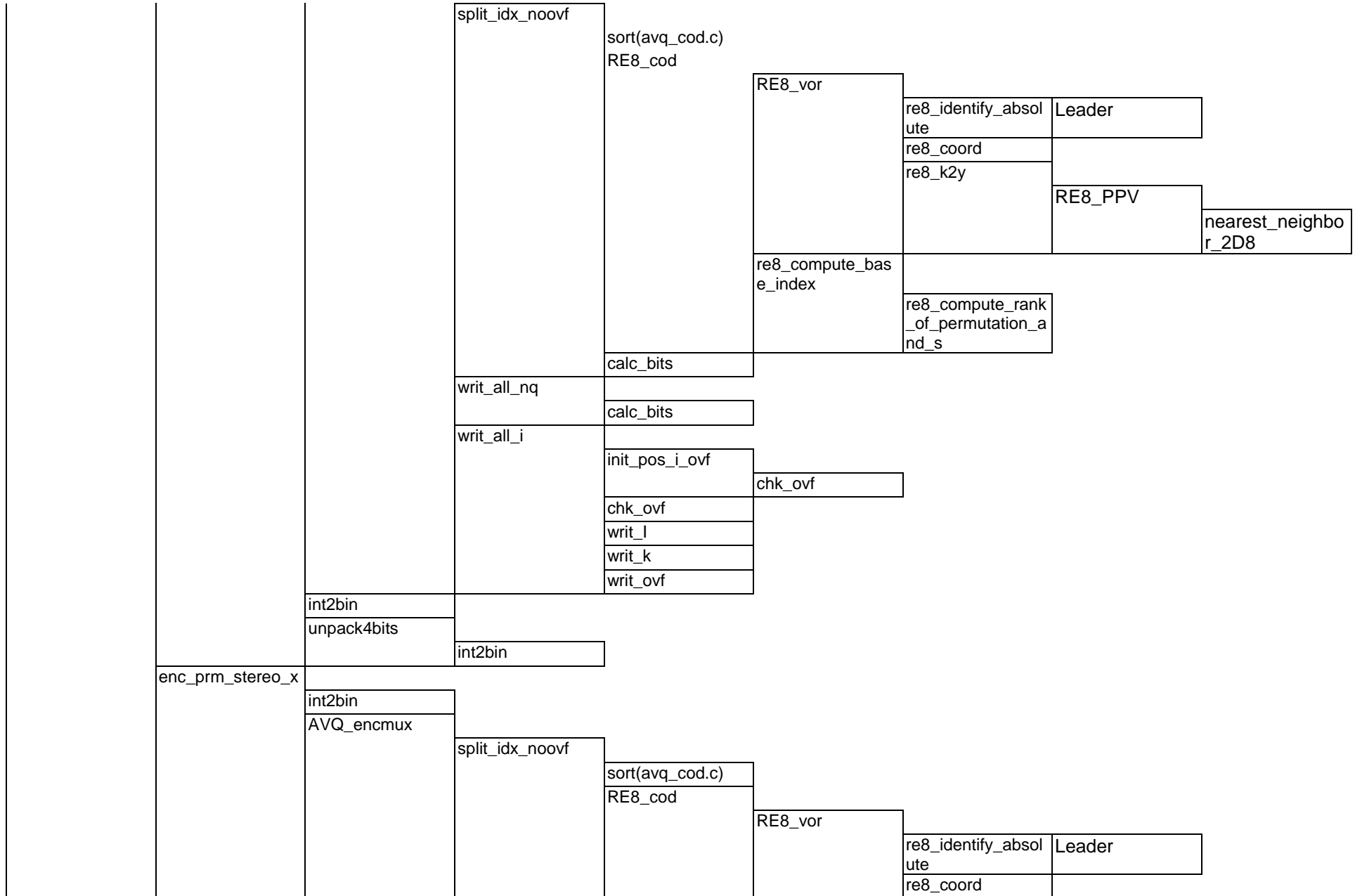






	E_UTIL_autocorr	
	lag_wind	
	E_LPC_lev_dur	
	E_LPC_a_isp_conversion	E_LPC_chebyshev
	int_lpc_npl	E_LPC_f_isp_a_conversion
		E_LPC_isp_f_pol_get
	mvr2r	
	E_LPC_isp_isf_conversion	
	q_isf_hf	sub_VQ(q_isf_hf.c)
		E_LPC_isf_reorderPlus
	isf2isp	
	match_gain_6k4	set_zero
		E_UTIL_residu
		E_UTIL_synthesisPlus
	int_gain	
	E_UTIL_residu	
	E_UTIL_synthesisPlus	
	E_LPC_a_weight	
	E_UTIL_residuPlus	
	q_gn_hf	
band_split_talignd_2k		interpol
coder_stereo_x		cod_hi_stereo
		mvr2r
		residu





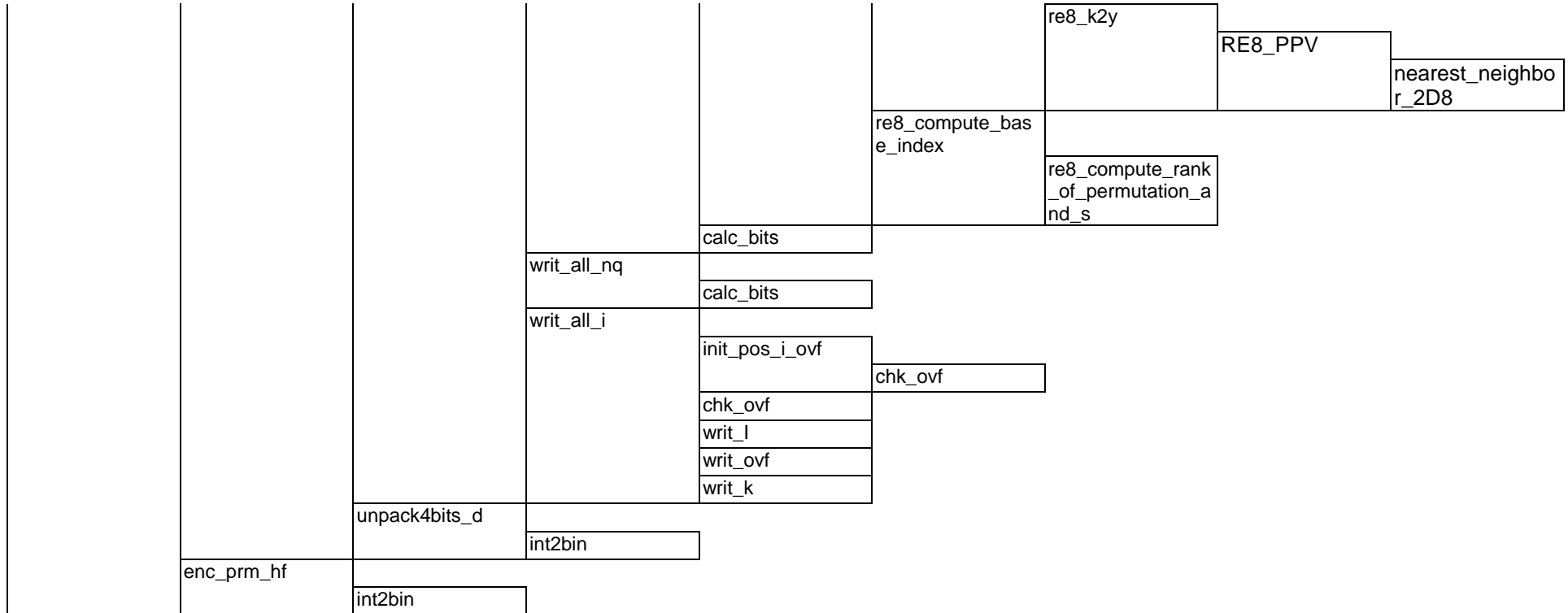
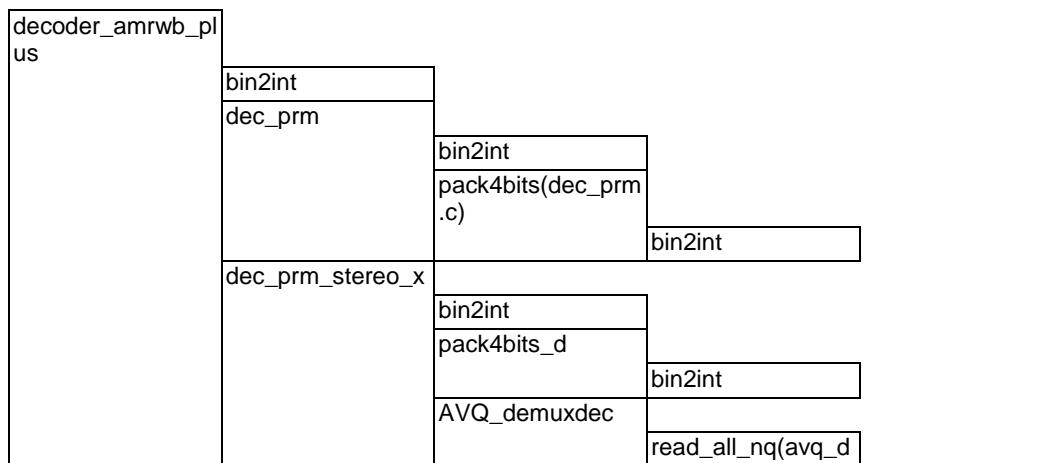
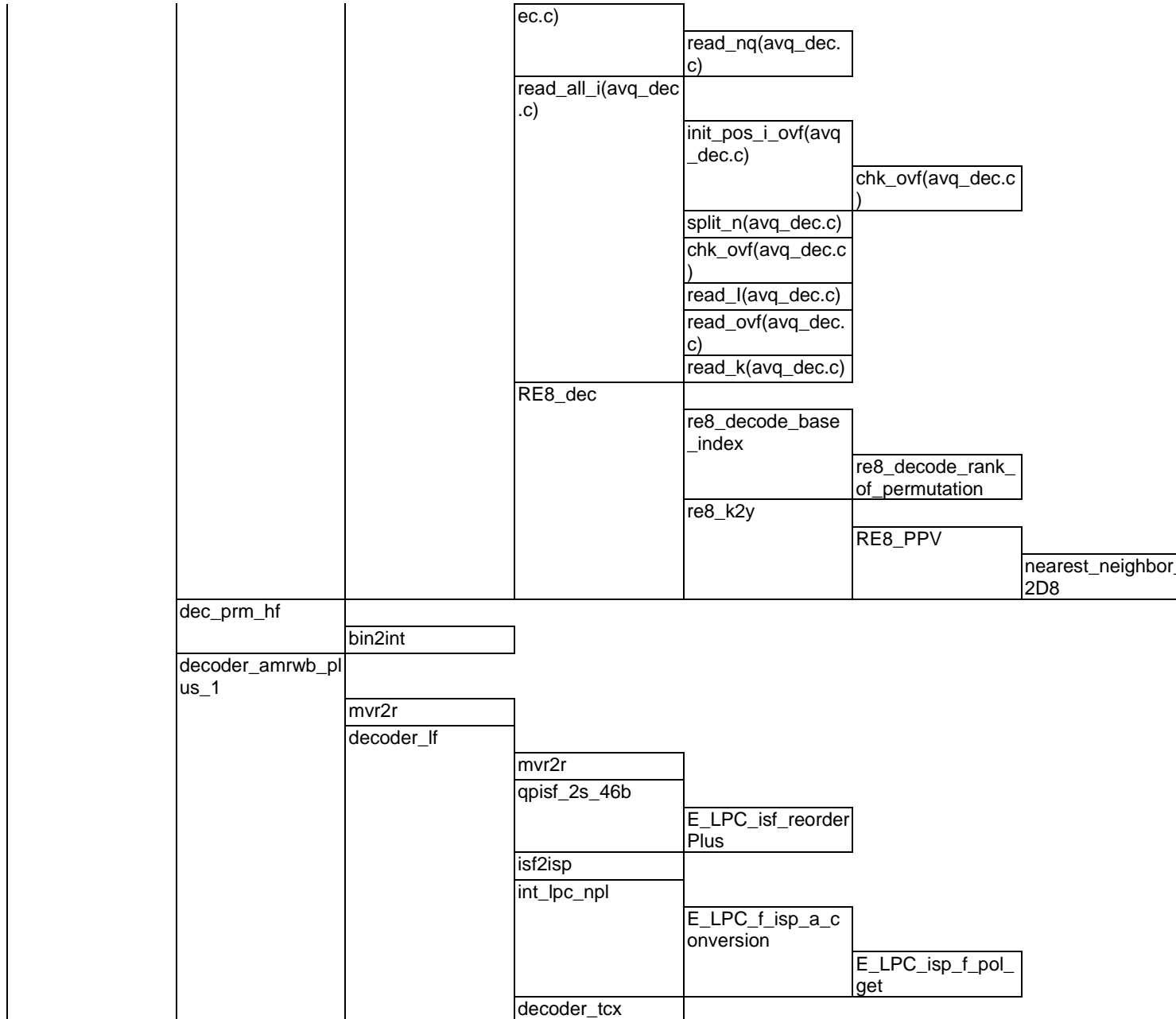
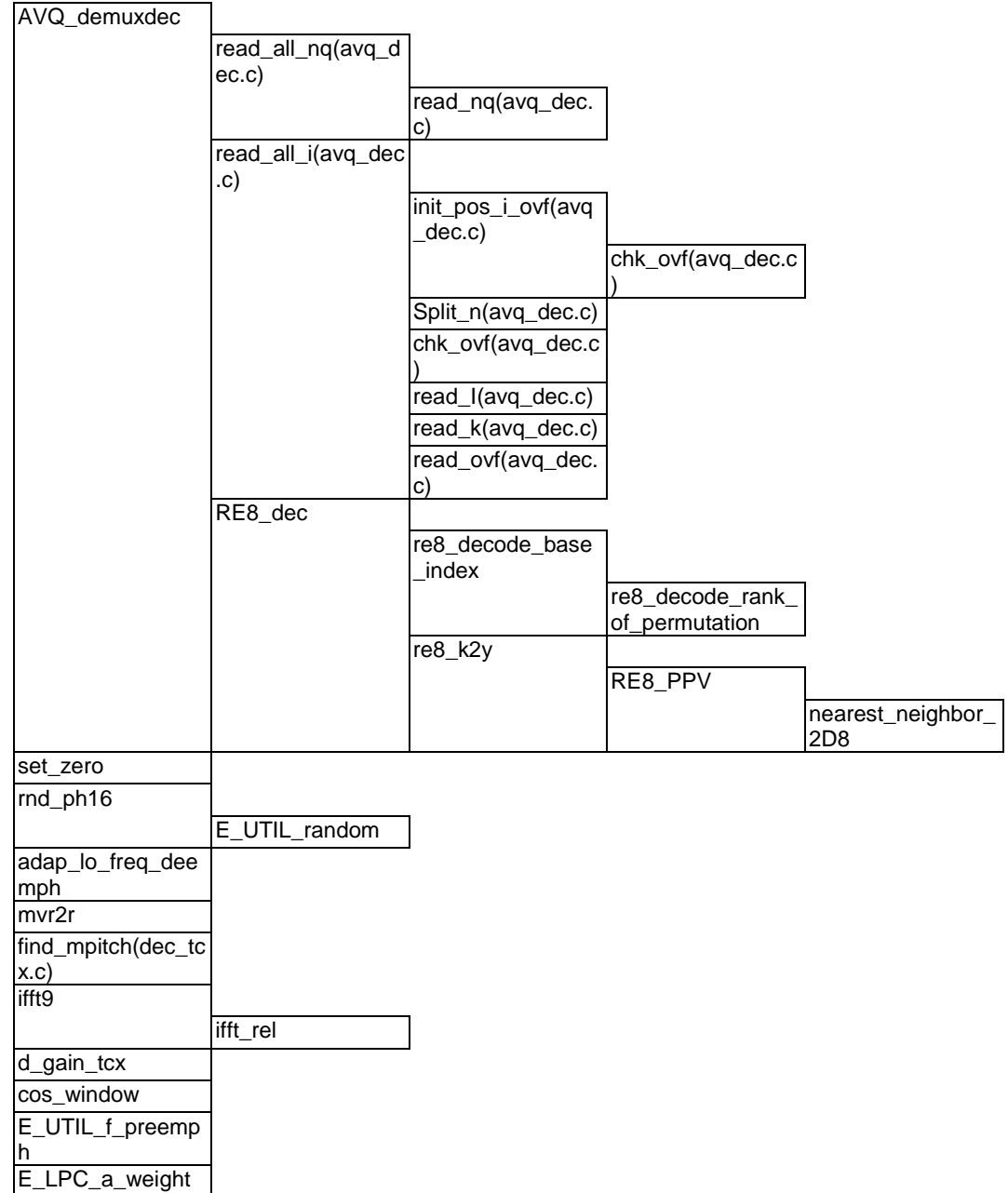
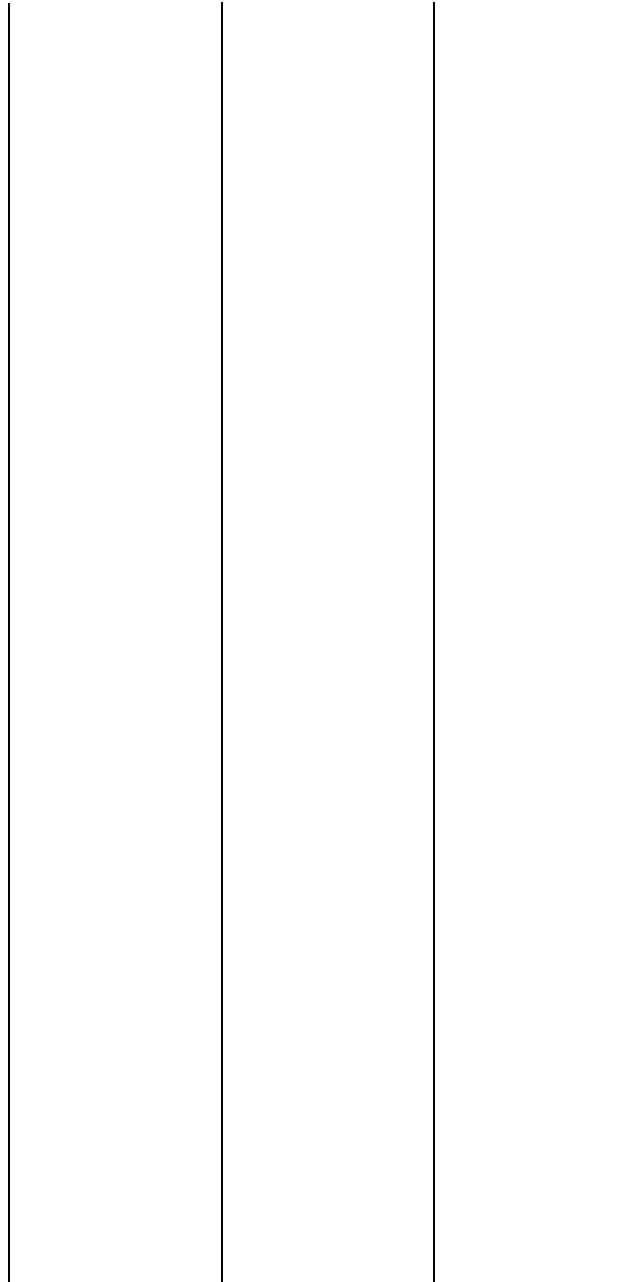
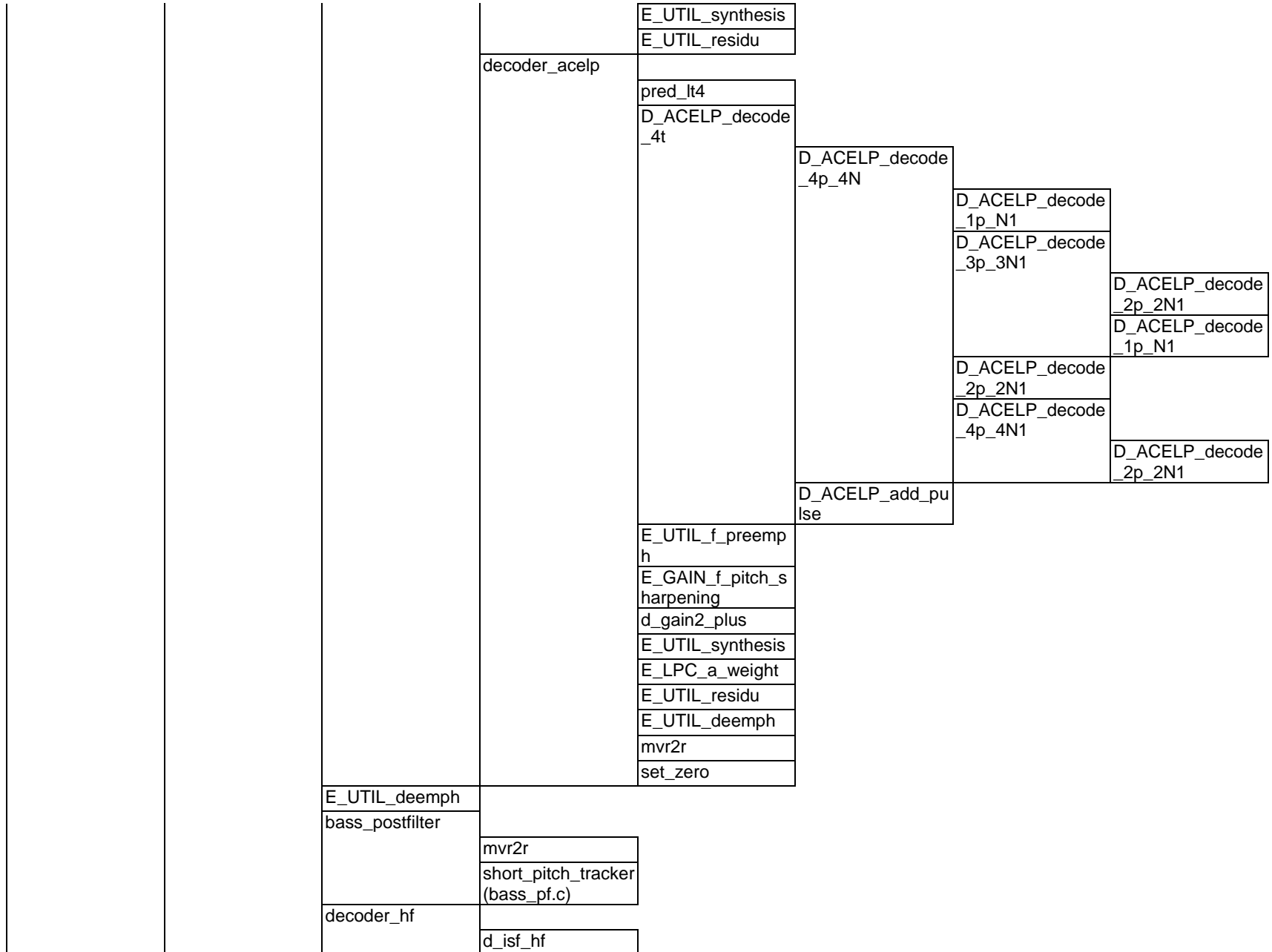


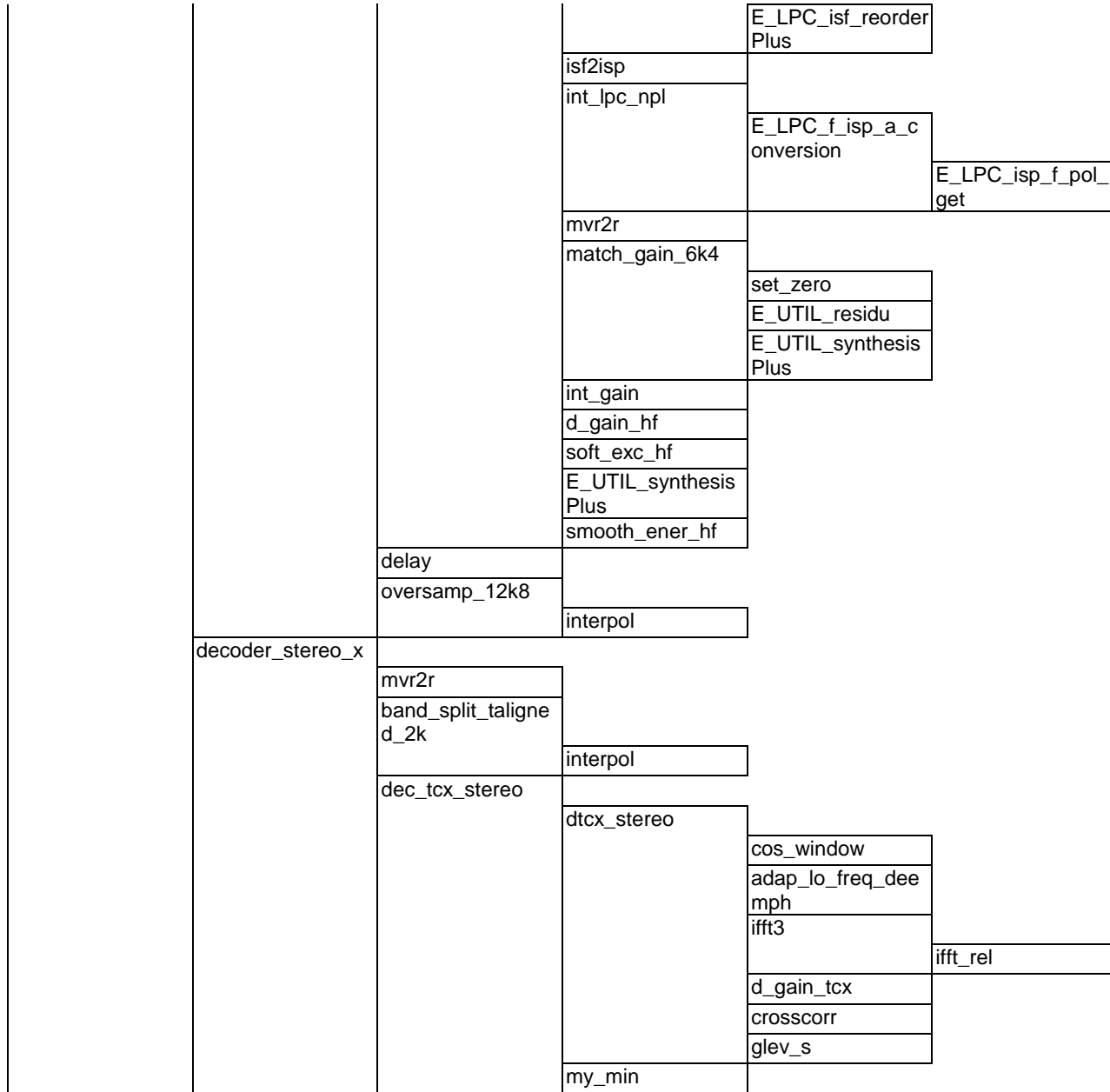
Table 2: Decoder call structure

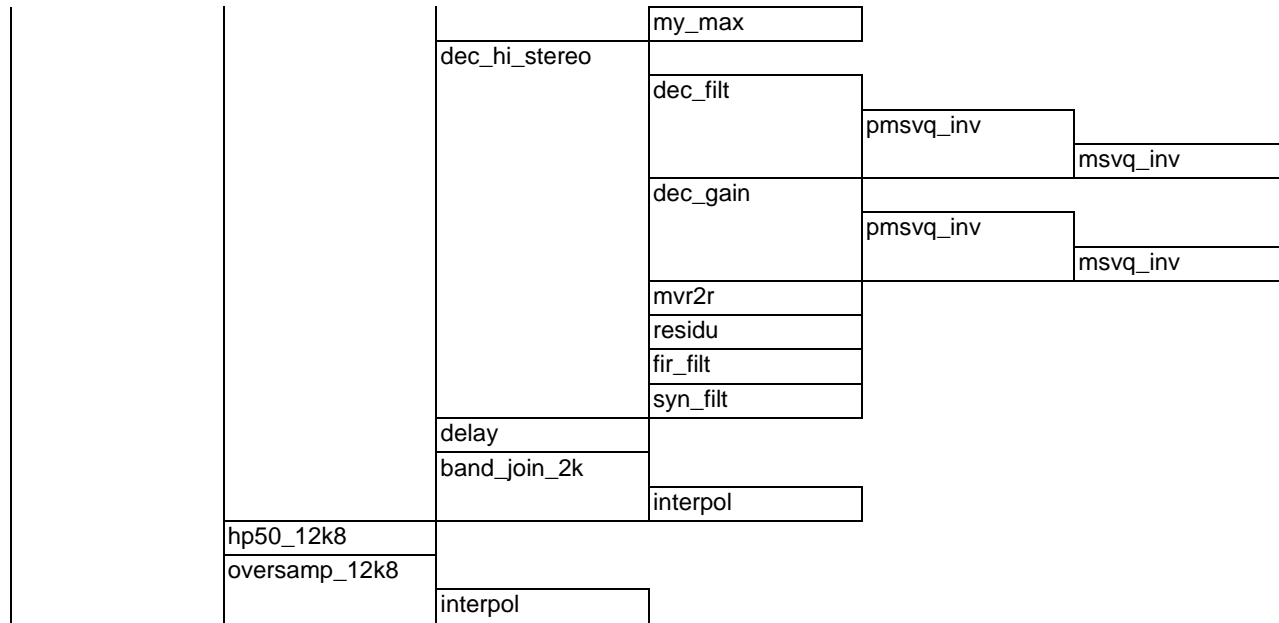












4.4 Variables, constants and tables

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 classifier
Float32	Gain_jf_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	Gain_hi_mscb_2a	4*2	
Float32	Gain_hi_mscb_5a	32*2	
	TBC		

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	
	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	TBC
	float	old_chan_2k	140	...
	float	old_chan_hi	448	...
	float	old_speech_2k	140	...
	float	old_speech_hi	448	...
	float	old_speech_pe	528	...
	float	old_wh	9	...
	float	old_wh_q	9	...
	float	old_gm_gain	2	...
	float	old_exc_mono	9	...
	float	filt_energy_threshold	1	...
	float	w_window	64	...
	PMSVQ*	*filt_hi_pmsvq	1	...
	PMSVQ*	*gain_hi_pmsvq	1	...
	int	mem_stereo_ovlp_size	1	...
	float	mem_stereo_ovlp	32	...
	NCLASSDATA	*stClass	1	...
	VadVars	*vadSt	1	...
	short	vad_hist	1	...
	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	TBC
	int	old_T0_med	1	...
	float	hp_old_wsp	699	...
	float	hp_ol_ltp_mem	7	...
	float	window	512	LP analysis window
	short	SwitchFlagPlusToWB	1	TBC
	float	mem_gain_code	4	...
	short	prev_mod	1	...

Table 6: Decoder static variables

struct name	type	variable	size	description
Decoder_StState				
	float	mem_oversamp	72	memory
	int	over_frac	1	
	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	
	float	mem_synth_hi	16	
	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	TBC
	float	mem_right_2k	20	...
	float	mem_left_hi	64	...
	float	mem_right_hi	64	...
	float	my_old_synth_2k	35	...
	float	my_old_synth_hi	128	...
	float	my_old_synth	148	...
	float	old_AqLF	85	...
	float	old_wh	9	...
	float	old_wh2	9	...
	float	old_exc_mono	9	...
	float	old_gain_left	4	...
	float	old_gain_right	4	...
	float	old_wh_q	9	...
	float	old_gm_gain	2	...
	float	w_window	64	...
	PMSVQ	*filt_hi_pmsvq	1	...
	PMSVQ	*gain_hi_pmsvq	1	...
	int	mem_stereo_ovlp_size	1	...
	float	mem_stereo_ovlp	32	...
	int	last_stereo_mode	1	...
	float	side_rms	1	...
	float	h	9	...
	float	mem_balance	1	...

	int	fer_hist	500	...
	int	fer_hist_ptr	1	...
	float	fer_mean	1	...
	float	old_xri	1148	...
	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	TBC
	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	TBC
	float	*dico1_isf_hf	1	...
	float	mem_gain_code	4	...
	float	mem_lpc_hf	9	...
	float	mem_gain_hf	1	...
	short	ramp_state	1	...

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)

[TBA]

Annex A (informative): Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
2004-06	SP-24	SP-040427	-	-	Presentation to TSG SA for information	-	1.0.0