ETSI TS 126 073 V12.0.0 (2014-09)

**Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
ANSI-C code for the Adaptive Multi Rate (AMR) speech codec
(3GPP TS 26.073 version 12.0.0 Release 12)**

Reference

RTS/TSGS-0426073vc00

Keywords

GSM,LTE,UMTS

Intellectual Property Rights

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

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

Contents

Foreword

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

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

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document contains an electronic copy of the ANSI-C code for the Adaptive Multi-Rate codec. The ANSI-C code is necessary for a bit exact implementation of the Adaptive Multi Rate speech transcoder (TS 26.090 [2]), Voice Activity Detection (TS 26.094 [6]), comfort noise (TS 26.092 [4]), source controlled rate operation (TS 26.093 [5]) and example solutions for substituting and muting of lost frames (TS 26.091 [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.

[1] 3GPP TS 26.074: "AMR Speech Codec; Test sequences".

[2] 3GPP TS 26.090: "AMR Speech Codec; Speech transcoding".

[3] 3GPP TS 26.091: "AMR Speech Codec; Substitution and muting of lost frames".

[4] 3GPP TS 26.092: "AMR Speech Codec; Comfort noise aspects".

[5] 3GPP TS 26.093: "AMR Speech Codec; Source controlled rate operation".

[6] 3GPP TS 26.094: "AMR Speech Codec; Voice Activity Detection".

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

3 Definitions and abbreviations

3.1 Definitions

Definition of terms used in the present document, can be found in TS 06.090 [2], TS 06.091 [3], TS 06.092 [4], TS 06.093 [5] and TS 06.094 [6].

3.2 Abbreviations

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

ANSI	American National Standards Institute
ETS	European Telecommunication Standard
GSM	Global System for Mobile communications
I/O	Input/Output
RAM	Random Access Memory
ROM	Read Only Memory

4 C code structure

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

The C code has been verified on the following systems:

- Sun Microsystems workstations and GNU gcc compiler;
- DEC Alpha workstations and GNU gcc compiler;
- IBM PC/AT compatible computers with Linux operating system and GNU gcc compiler.

ANSI-C 9899 was selected as the programming language because portability was desirable.

4.1 Contents of the C source code

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

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

The C code distribution also contains one speech coder installation verification data file, "spch_dos.inp". The reference encoder output file is named "spch_dos.cod", the reference decoder input file is named "spch_dos.dec" and the reference decoder output file is named "spch_dos.out". These four files are formatted such that they are correct for an IBM PC/AT compatible computer. The same files with reversed byte order of the 16 bit words are named "spch_unx.inp", "spch_unx.cod", "spch_unx.dec" and "spch_unx.out", respectively.

Final verification is to be performed using the GSM Adaptive Multi-Rate test sequences described in GSM 06.74 [2].

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

4.2 Program execution

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

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

The programs should be called like:

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

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

The encoder and decoder options will be explained by running the applications with option –h. See the file readme.txt for more information on how to run the *encoder* and *decoder* programs.

4.3 Coding style

The C code is written according to the following structuring conventions. Each function func() that needs static variables is considered a module. A module consists of:

- a 'state structure' (struct) combining the static variables of the module;
- three auxiliary functions func_init(), func_reset(), and func_exit();
- the processing function func() itself.

The initialization function func_init() allocates (from the heap) a new state structure, calls the func_reset() function, stores the pointer to the newly allocated structure in its first function parameter, and returns with a value of 0 if completed successful or a value of 1 otherwise.

The reset function func_reset() takes a pointer to the state structure and resets all members of the structure to a predefined value ('homing').

The exit function func_exit() performs any necessary cleanup and frees the state structure memory.

The processing function func() also takes a pointer to the state structure as well as all other necessary parameters and performs its task using (and possibly modifying) the values in the state structure.

If a module calls other modules, the higher level state structure contains a pointer to the lower level state structures, and the init, reset, and exit functions recursively call the corresponding lower level functions.

By this convention, the code becomes "instantiable" (more than one copy of a module can be used in the same program) and the static data hierarchy is clearly visible in the code.

4.4 Code hierarchy

Figures 1 to 4 are call graphs that show the functions used in the speech codec, including the functions of VAD, DTX, and comfort noise generation.

Each column represents a call level and each cell a function. The functions contain calls to the functions in rightwards neighbouring cells. The time order in the call graphs is from the top downwards as the processing of a frame advances. All standard C functions: printf(), fwrite(), etc. have been omitted. Also, no basic operations (add(), L_add(), mac(), etc.) or double precision extended operations (e.g. L_Extract()) appear in the graphs. The initialization of the static RAM (i.e. calling the _init functions) is also omitted.

The basic operations are not counted as extending the depth, therefore the deepest level in this software is level 7.

The encoder call graph is broken down into three separate call graphs, Table 1 to 3.

Table 1: Speech encoder call structure

Speech_Encode_Frame	Pre_Process				
	cod_amr	Copy			
		Vad1[1]	filter_bank	first_filter_stage	
				filter5	
				filter3	
				level_calculation	
			vad_decision	complex_estimate_adapt	
				complex_vad	
				noise_estimate_update	update_cntrl
				hangover_addition	
		Vad2[1]	block_norm		
			r_fft	c_fft	
			fn10Log10	Log2	Log2_norm
			Pow2		
		tx_dtx_handler			
		lpc	Autocorr		
			Lag_window		
			Levinson		
		lsp	Az_lsp	Chebps	
			Q_plsf_5	Lsp_lsf	
				Lsf_wt	
				Vq_subvec	
				Vq_subvec_s	
				Reorder_lsf	
				Lsf_lsp	
			Int_lpc_1and3_2	Lsp_az	Get_lsp_pol
			Int_lpc_1and3	Lsp_az	Get_lsp_pol
			Q_plsf_3	Lsp_lsf	
				Lsf_wt	
				Copy	
				Vq_subvec3	
				Vq_subvec4	
				Reorder_lsf	
				Lsf_lsp	
			Int_lpc_1to3_2	Lsp_az	Get_lsp_pol
			Int_lpc_1to3	Lsp_az	Get_lsp_pol
			Copy		
		dtx_buffer	Copy		
			Log2	Log2_norm	
		dtx_enc	Lsp_lsf		
			Reorder_lsf		
			Lsf_lsp		
		Set_zero			
		lsp_reset	Copy		
			Q_plsf_reset		
		cl_ltp_reset	Pitch_fr_reset		
		check_lsp			
		pre_big	Weight_Ai		
			Residu		
			Syn_filt		
		ol_ltp	Pitch_ol	vad_tone_detection_update[2]	
				Lag_max	vad_tone_detection[2]
					Inv_sqrt
				comp_corr[2]	
				hp_max[2]	
				vad_complex_detection_update[2]	
			Pitch_ol_wgh	comp_corr[2]	
				Lag_max[2]	vad_tone_detection_update[2]
					vad_tone_detection[2]
				gmed_n	
				hp_max[2]	
				vad_complex_detection_update[2]	
		vad_pitch_detection	LTP_flag_update[3]		
		subframePreProc	Weight_Ai		
			Syn_filt		
			Residu		
			Copy		
		cl_ltp	Pitch_fr	getRange	
				Norm_Corr	Convolve
					Inv_sqrt
				searchFrac	Interpol_3or6
				Enc_lag3	
				Enc_lag6	

(continued)

1 Option to call one or the other VAD option

2 Specific to VAD option 1

3 Specific to VAD option 2

Table 1 (concluded): Speech encoder call structure

			Pred_lt_3or6
			Convolve
			G_pitch
			check_gp_clipping
			q_gain_pitch
		cbsearch	see Table 2
		gainQuant	see Table 3
		update_gp_clipping	Copy
		subframePostProc	Syn_filt
		Pred_lt_3or6	
		Convolve	
	Prm2bits	Int2bin	

Table 2: cbsearch call structure

cbsearch	code_2i40_9bits	cor_h_x	
		set_sign	
		cor_h	Inv_sqrt
		search_2i40	
		build_code	
	code_2i40_11bits	cor_h_x	
		set_sign	
		cor_h	Inv_sqrt
		search_2i40	
		build_code	
	code_3i40_14bits	cor_h_x	
		set_sign	
		cor_h	Inv_sqrt
		search_3i40	
		build_code	
	code_4i40_17bits	cor_h_x	
		set_sign	
		cor_h	Inv_sqrt
		search_4i40	
		build_code	
	code_8i40_31bits	cor_h_x	
		set_sign12k2	Inv_sqrt
		cor_h	Inv_sqrt
		search_10and8i40	
		build_code	
		compress_code	compress10
	code_10i40_35bits	cor_h_x	
		set_sign12k2	Inv_sqrt
		cor_h	Inv_sqrt
		search_10and8i40	
		build_code	
		q_p	

Table 3: gainQuant call structure

gainQuant	gc_pred_copy	Copy		
	gc_pred	Log2	Log2_norm	
		Log2_norm		
	calc_filt_energies			
	calc_target_energy			
	MR475_update_unq_pred	gc_pred_update		
	MR475_gain_quant	MR475_quant_store_results	Log2	Log2_norm
			gc_pred_update	
		gc_pred	Log2	Log2_norm
			Log2_norm	
	G_code			
	q_gain_code	Pow2		
	MR795_gain_quant	q_gain_pitch		
		MR795_gain_code_quant3		
		calc_unfilt_energies	Log2	Log2_norm
		gain_adapt	gmed_n	
		MR795_gain_code_quant_mod	sqrt_l_exp	
	Qua_gain	Pow2		
	gc_pred_update			

Table 4: Speech decoder call structure

Speech_Decode_Frame	Bits2prm	Bin2int		
	Decoder_amr	rx_dtx_handler		
		Decoder_amr_reset	lsp_avg_reset	
			D_plsf_reset	
			ec_gain_pitch_reset	
			ec_gain_code_reset	
			gc_pred_reset	
			Bgn_scd_reset	Set_zero
			ph_disp_reset	
			dtx_dec_reset	Copy
				Set_zero
		dtx_dec	Copy	
			Lsf_lsp	
			Init_D_plsf_3	Copy
			D_plsf_3	Reorder_lsf
				Copy
				Lsf_lsp
			pseudonoise	
			Lsp_lsf	
			Reorder_lsf	
			Lsp_Az	Get_lsp_pol
			A_Refl	
			Log2	Log2_norm
			Build_CN_code	pseudonoise
			Syn_filt	
		Lsf_lsp		
		lsp_avg		
		Copy		
		D_plsf_3	Reorder_lsf	
			Copy	
			Lsf_lsp	
		Int_lpc_1to3	Lsp_Az	Get_lsp_pol
		D_plsf_5	Reorder_lsf	
			Copy	
			Lsf_lsp	
		Int_lpc_1and3	Lsp_Az	Get_lsp_pol
		Dec_lag3		
		Pred_lt_3or6		
		Dec_lag6		
		decode_2i40_9bits		
		decode_2i40_11bits		
		decode_3i40_14bits		
		decode_4i40_17bits		
		decode_8i40_31bits	decompress_code	decompress10
		ec_gain_pitch	gmed_n	
		d_gain_pitch		
		ec_gain_pitch_update		
		decode_10i40_35bits		
		Dec_gain	Log2	Log2_norm	
			gc_pred	Log2	Log2_norm
				Log2_norm	
			Pow2		
			gc_pred_update		
		ec_gain_code	gmed_n		
			gc_pred_average_limeted		
			gc_pred_update		
		ec_gain_code_update			
		d_gain_code	gc_pred	Log2	Log2_norm
				Log2_norm	
			Pow2		
			gc_pred_update		
		Int_lsf			
		Cb_gain_average			
		ph_disp_release			
		ph_disp_lock			
		ph_disp			
		sqrt_l_exp			
		Ex_ctrl	gmed_n		
		agc2	Inv_sqrt		
		Syn_filt			
		Bgn_scd	gmed_n		
		dtx_dec_activity_update	Copy		
			Log2	Log2_norm	
		lsp_avg			
	Post_Filter	Copy			
		Weight_Ai			
		Residu			
		Set_zero			
		Syn_filt			
		Preemphasis			
		agc	energy_old		
			energy_new	energy_old	
			Inv_sqrt		
	Post_Process				

4.5 Variables, constants and tables

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

- **Word16** 16 bit variable;
- **Word32** 32 bit variable.

Furthermore some **enum** types are used, all possible to represent with one byte, and a Boolean **Flag**.

4.5.1 Description of constants used in the C-code

This subclause contains a listing of all global constants defined in cnst.h.

Table 5: Global constants

Constant	Value	Description
L_TOTAL	320	total size of speech buffer.
L_WINDOW	240	window size in LP analysis
L_FRAME	160	frame size
L_FRAME_BY2	80	frame size divided by 2
L_SUBFR	40	subframe size
L_CODE	40	codevector length
NB_TRACK	5	number of tracks
STEP	5	codebook step size
NB_TRACK_MR102	4	number of tracks mode mr102
STEP_MR102	4	codebook step size mode mr102
M	10	order of LP filter
MP1	(M+1)	order of LP filter + 1
LSF_GAP	205	minimum distance between LSF after quantization; 50 Hz = 205
LSP_PRED_FAC_MR122	21299	MR122 LSP prediction factor (0.65 Q15)
AZ_SIZE	44	size of array of LP filters in 4 subframes (4*M+4)
PIT_MIN_MR122	18	minimum pitch lag (MR122 mode)
PIT_MIN	20	minimum pitch lag (all other modes)
PIT_MAX	143	maximum pitch lag
L_INTERPOL	(10+1)	length of filter for interpolation
L_INTER_SRCH	4	length of filter for CL LTP search interpolation
MU	26214	factor for tilt compensation filter 0,8
AGC_FAC	29491	factor for automatic gain control 0,9
L_NEXT	40	overhead in LP analysis
SHARPMAX	13017	maximum value of pitch sharpening
SHARPMIN	0	minimum value of pitch sharpening
MAX_PRM_SIZE	57	max. num. of params
MAX_SERIAL_SIZE	244	max. num. of serial bits
GP_CLIP	15565	pitch gain clipping = 0.95
N_FRAME	7	old pitch gains in average calculation
EHF_MASK	8	16 bit representation of all samples in the encoder homing frame (left justification)

4.5.2 Description of fixed tables used in the C-code

This section contains a listing of all fixed tables sorted by source file name and table name. All table data is declared as **Word16**.

Table 6: Fixed tables

File	Table name	Length	Description
c2_9pf.c	trackTable	4*5	track table for algebraic code book search (MR475, MR515)
cod_amr.c	gamma1	10	spectral expansion factors
cod_amr.c	gamma1_12k2	10	spectral expansion factors
cod_amr.c	gamma2	10	spectral expansion factors
dtx_dec.c	lsf_hist_mean_scale	10	initialization values for DTX lsf parameters
dtx_dec.c	dtx_log_en_adjust	9	level adjustments for ech mode
ec_gains.c	cdown	7	attenuation factors for codebook gain
ec_gains.c	pdown	7	attenuation factors for adaptive codebook gain
gc_pred.c	pred	4	algebraic code book gain MA predictor coefficients
gc_pred.c	pred_MR122	4	algebraic code book gain MA predictor coefficients (MR122)
pitch_fr.c	mode_dep_parm	72	parameters defining the adaptive codebook search per mode
post_pro.c	a	3	HP filter coefficients (denominator) in Post_Process
post_pro.c	b	3	HP filter coefficients (numerator) in Post_Process
pre_proc.c	a	3	HP filter coefficients (denominator) in Pre_Process
pre_proc.c	b	3	HP filter coefficients (numerator) in Pre_Process
pred_lt.c	inter_6	61	interpolation filter coefficients
pstfilt.c	gamma3_MR122	10	spectral expansion factors
pstfilt.c	gamma3	10	spectral expansion factors
pstfilt.c	gamma4_MR122	10	spectral expansion factors
pstfilt.c	gamma4	10	spectral expansion factors
bitno.tab	prmno	9	number of bits for each mode
bitno.tab	prmnofsf	8	number of parameters for LPC and first subframe for each mode (used for decoder homing procedure)
bitno.tab	bitno	9	pointers to the bitno_MR... tables
bitno.tab	bitno_MR475	17	number of bits per parameter to transmit (MR475)
bitno.tab	bitno_MR515	19	number of bits per parameter to transmit (MR515)
bitno.tab	bitno_MR59	19	number of bits per parameter to transmit (MR59)
bitno.tab	bitno_MR67	19	number of bits per parameter to transmit (MR67)
bitno.tab	bitno_MR74	19	number of bits per parameter to transmit (MR74)
bitno.tab	bitno_MR795	23	number of bits per parameter to transmit (MR795)
bitno.tab	bitno_MR102	39	number of bits per parameter to transmit (MR102)
bitno.tab	bitno_MR122	57	number of bits per parameter to transmit (MR122)
bitno.tab	bitno_MRDTX	5	number of bits per parameter to transmit (MRDTX)
c2_11pf.tab	startPos1	2	track start search position for first pulse
c2_11pf.tab	startPos2	4	track start search position for second pulse
c2_9pf.tab	startPos	16	track start search position
corrwght.tab	corrweight	251	weighting of the correlation function in open loop LTP search (MR102)
d_homing.tab	dhf	8	pointers to the dhf_MR… tables
d_homing.tab	dhf_MR475	17	parameter values for the decoder homing frame (MR475)
d_homing.tab	dhf_MR515	19	parameter values for the decoder homing frame (MR515)
d_homing.tab	dhf_MR59	19	parameter values for the decoder homing frame (MR59)
d_homing.tab	dhf_MR67	19	parameter values for the decoder homing frame (MR67)
d_homing.tab	dhf_MR74	19	parameter values for the decoder homing frame (MR74)
d_homing.tab	dhf_MR795	23	parameter values for the decoder homing frame (MR795)
d_homing.tab	dhf_MR102	39	parameter values for the decoder homing frame (MR102)
d_homing.tab	dhf_MR122	57	parameter values for the decoder homing frame (MR122)
gains.tab	qua_gain_pitch	16	adaptive codebook gain quantization table (MR122, MR795)
gains.tab	qua_gain_code	96	fixed codebook gain quantization table (MR122, MR795)
gray.tab	gray	8	gray coding table
gray.tab	dgray	8	gray decoding table
grid.tab	grid	61	grid points at wich Chebyshev polynomials are evaluated
inter_36.tab	inter_6	25	interpolation filter coefficients
inv_sqrt.tab	table	49	table used in inverse square root computation
lag_wind.tab	lag_h	10	high part of the lag window table
lag_wind.tab	lag_l	10	low part of the lag window table

(continued)

Table 6 (concluded): Fixed tables

File	Table name	Length	Description
log2.tab	table	33	table used inbase 2 logharithm computation
lsp.tab	lsp_init_data	10	initialization table for lsp history in DTX
lsp_lsf.tab	table	65	table to compute cos(x) in Lsf_lsp()
lsp_lsf.tab	slope	64	table to compute acos(x) in Lsp_lsf()
ph_disp.tab	ph_imp_low_MR795	40	phase dispersion impulse response (MR795)
ph_disp.tab	ph_imp_mid_MR795	40	phase dispersion impulse response (MR795)
ph_disp.tab	ph_imp_low	40	phase dispersion impulse response (MR475 - MR67)
ph_disp.tab	ph_imp_mid	40	phase dispersion impulse response (MR475 - MR67)
pow2.tab	table	33	table used in 2 to the power computation
q_plsf_3.tab	past_rq_init	80	initialization table for the MA predictor in DTX
q_plsf_3.tab	mean_lsf	10	LSF means (not in MR122)
q_plsf_3.tab	pred_fac	10	LSF prediction factors (not in MR122)
q_plsf_3.tab	dico1_lsf	3*256	1st LSF quantizer (not in MR122 and MR795)
q_plsf_3.tab	dico2_lsf	3*512	2nd LSF quantizer (not in MR122)
q_plsf_3.tab	dico3_lsf	4*512	3rd LSF quantizer (not in MR122, MR515 and MR475)
q_plsf_3.tab	mr515_3_lsf	4*128	3rd LSF quantizer (MR515 and MR475)
q_plsf_3.tab	mr795_1_lsf	3*512	1st LSF quantizer (MR795)
q_plsf_5.tab	mean_lsf	10	LSF means (MR122)
q_plsf_5.tab	dico1_lsf	4*128	1st LSF quantizer (MR122)
q_plsf_5.tab	dico2_lsf	4*256	2nd LSF quantizer (MR122)
q_plsf_5.tab	dico3_lsf	4*256	3rd LSF quantizer (MR122)
q_plsf_5.tab	dico4_lsf	4*256	4th LSF quantizer (MR122)
q_plsf_5.tab	dico5_lsf	4*64	5th LSF quantizer (MR122)
qgain475.tab	table_gain_MR475	4*256	gain quantization table (MR475)
qua_gain.tab	table_gain_highrates	128*4	gain quantization table (MR67, MR74 and MR102)
qua_gain.tab	table_gain_lowrates	64*4	gain quantization table (MR515 and MR59)
R_fft.c	phs_tbl	128	sine/cosine phase table
R_fft.c	ii_table	8	indexing table
sqrt_l	table	49	table to compute sqrt(x)
Vad1.c	ch_tbl	2*16	channel energy combination table
Vad1.c	ch_tbl_sh	16	channel energy scaling table
Vad1.c	vm_tbl	90	voice metric table
Vad1.c	hangover_table	20	used to determine hangover as a function of SNR
Vad1.c	burstcount_table	20	used to determine burst count threshold as a function of SNR
Vad1.c	vm_thresh_table	20	used to determine the voice metric threshold as a function of SNR
Vad1.c	energy state tables	2*6	constants as a function of scaling state
window.tab	window_200_40	240	LP analysis window (not in MR122)
window.tab	window_160_80	240	1st LP analysis window (MR122)
window.tab	window_232_8	240	2nd LP analysis window (MR122)

4.5.3 Static variables used in the C-code

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

Table 7: Speech encoder static variables

Struct name	Variable	Type[Length]	Description
Speech_Encode_ FrameState	cod_amr_state	cod_amrState	see below in this table
	pre_state	Pre_ProcessState	see below in this table
	dtx	Flag	Is set if DTX functionality is used
	complexityCounter	int	Used for wMOPS counting
Pre_ProcessState	y2_hi	Word16	filter state, upper word
	y2_lo	Word16	filter state, lower word
	y1_hi	Word16	filter state, upper word
	y1_lo	Word16	filter state, lower word
	x0	Word16	filter state
	x1	Word16	filter state
cod_amrState	old_speech	Word16[320]	speech buffer
	speech	Word16*	pointer to current frame in old_speech
	p_window	Word16*	pointer to LPC analysis window in old_speech
	p_window_12k2	Word16*	pointer to LPC analysis window with no lookahead in old_speech (MR122)
	new_speech	Word16*	pointer to the last 160 speech samples in old_speech
	old_wsp	Word16[303]	buffer holding spectral weighted speech
	wsp	Word16*	pointer to the current frame in old_wsp
	old_lags	Word16[5]	open loop LTP states
	ol_gain_flg	Word16[2]	enables open loop pitch lag weighting (MR102)
	old_exc	Word16[314]	excitation vector
	exc	Word16*	current excitation
	ai_zero	Word16[51]	history of weighted synth. filter followed by zero vector
	zero	Word16*	zero vector
	h1	Word16*	impulse response of weighted synthesis filter
	hvec	Word16[80]	zero vector followed by impulse response
	lpcSt	lpcState	see below in this table
	lspSt	lspState	see below in this table
	clLtpSt	clLtpState	see below in this table
	gainQuantSt	gainQuantState	see below in this table
	pitchOLWghtSt	pitchOLWghtState	see below in this table
	tonStabSt	tonStabState	see below in this table
	vadSt	vadState1	see below in this table
	vadSt	vadState2	see below in this table
	dtx	Flag	is set if DTX functionality is used
	dtx_encSt	dtx_encState	see below in this table
	mem_syn	Word16[10]	synthesis filter memory
	mem_w0	Word16[10]	weighting filter memory (applied to error signal)
	mem_w	Word16[10]	weighting filter memory (applied to input signal)
	mem_err	Word16[50]	filter memory for production of error vector
	error	Word16*	error signal (input minus synthesized speech)
	sharp	Word16	pitch sharpening gain
vadState1	bckr_est	Word16[9]	background noise estimate
	ave_level	Word16[9]	averaged input components for stationary estimation
	old_level	Word16[9]	input levels of the previous frame
	sub_level	Word16[9]	input levels calculated at the end of a frame (lookahead)
	a_data5	Word16[6]	memory for the filter bank
	a_data3	Word16[5]	memory for the filter bank
	burst_count	Word16	counts length of a speech burst
	hang_count	Word16	hangover counter
	stat_count	Word16	stationary counter
	vadreg	Word16	15 flags for intermediate VAD decisions
	pitch	Word16	15 flags for pitch detection
	tone	Word16	15 flags for tone detection
	complex_high	Word16	flags for complex detection
	complex_low	Word16	flags for complex detection
	oldlag_count	Word16	variables for pitch detection
	oldlag	Word16	variables for pitch detection
	complex_hang_count	Word16	complex hangover counter, used by VAD

Struct name	Variable	Type[Length]	Description
vadState2	complex_hang_timer	Word16	hangover initiator, used by CAD
	best_corr_hp	Word16	filtered value
	speech_vad_decision	Word16	final decision
	complex_warning	Word16	complex background warning
	sp_burst_count	Word16	counts length of a speech burst incl HO addition
	corr_hp_fast	Word16	filtered value
	pre_emp_mem	Word16	input pre-emphasis memory
	update_cnt	Word16	noise update counter
	hyster_cnt	Word16	hysteresis counter
	last_update_cnt	Word16	noise update counter value for last frame
	ch_enrg_long_db	Word16[16]	long term channel energy in dB
	Lframe_cnt	Word32	10 ms frame counter
	Lch_enrg	Word32[16]	channel energy estimate
	Lch_noise	Word32[16]	channel noise estimate
	last_normb_shift	Word16	block shift factor for last frame, used for pre_emp_mem
	tsnr	Word16	total estimated peak SNR in dB
	hangover	Word16	VAD hangover
	burstcount	Word16	number of consecutive voice active frames
	fupdate_flag	Word16	A flag to control a forced update of the noise estimate
	negSNRvar	Word16	SNR variability
	negSNRbias	Word16	sensitivity bias
	shift_state	Word16	indicates scaling state of channel energy estimate
	L_R0	Word32	LTP energy
	L_Rmax	Word32	LTP max correlation
	LTP_flag	Flag	set when open loop pitch prediction gain > threshold
dtx_encState	lsp_hist	Word16[80]	LSP history (8 frames)
	log_en_hist	Word16[8]	logarithmic frame energy history (8 frames)
	hist_ptr	Word16	pointer to the cyclic history vectors
	log_en_index	Word16	Index for logarithmic energy
	init_lsf_vq_index	Word16	initial index for lsf predictor
	lsp_index	Word16[3]	lsp indecies to the three code books
	dtxHangoverCount	Word16	is decreased in DTX hangover period
	decAnaElapsedCount	Word16	counter for elapsed speech frames in DTX
lpcState	LevinsonSt	LevinsonState	see below
LevinsonState	old_A	Word16[11]	last frames direct form coefficients
lspState	lsp_old	Word16[10]	old LSP vector
	lsp_old_q	Word16[10]	old quantized LSP vector
	qSt	Q_plsfState	see below in this table
Q_plsfState	past_rq	Word16[10]	past quantized LSF prediction error
clLtpState	pitchSt	Pitch_frState	see below in this table
tonStabState	count	Word16	count consecutive (potential) resonance frames
	gp	Word16[7]	pitch gain history
Pitch_frState	T0_prev_subframe	Word16	integer. pitch lag of previous subframe
gainQuantState	sf0_exp_gcode0	Word16	subframe 0/2 codebook gain exponent
	sf0_frac_gcode0	Word16	subframe 0/2 codebook gain fraction
	sf0_exp_target_en	Word16	subframe 0/2 target energy exponent
	sf0_frac_target_en	Word16	subframe 0/2 target energy fraction
	sf0_exp_coeff	Word16[5]	subframe 0/2 energy coefficient exponents
	sf0_frac_coeff	Word16[5]	subframe 0/2 energy coefficient fractions
	gain_idx_ptr	Word16*	pointer to gain index value in parameter frame
	gc_predSt	gc_predState	see below in this table
	gc_predUncSt	gc_predState	see below in this table
	adaptSt	GainAdaptState	see below in this table
gc_predState	past_qua_en	Word16[4]	MA predictor memory (20*log10(pred. error))
	past_qua_en_MR122	Word16[4]	MA predictor memory, 12.2 style (log2(pred. error))
GainAdaptState	onset	Word16	onset counter
	prev_alpha	Word16	previous adaptor output
	prev_gc	Word16	previous codebook gain
	ltpg_mem	Word16[5]	pitch gain history

Struct name	Variable	Type[Length]	Description
pitchOLWghtState	old_T0_med	Word16	weighted open loop pitch lag
	ada_w	Word16	weigthing level depeding on open loop pitch gain
	wght_flg	Word16	switches lag weighting on and off

Table 8: Speech decoder static variables

Struct name	Variable	Type[Length]	Description
Speech_Decode_FrameState	decoder_amrState	Decoder_amrState	see below in this table
	post_state	Post_FilterState	see below in this table
	postHP_state	Post_ProcessState	see below in this table
	ComplexityCounter	int	Used for wMOPS counting
Decoder_amrState	old_exc	Word16[194]	excitation vector
	exc	Word16*	current excitation
	lsp_old	Word16[10]	LSP vector of previous frame
	mem_syn	Word16[10]	synthesis filter memory
	sharp	Word16	pitch sharpening gain
	old_T0	Word16	pitch sharpening lag
	prev_bf	Word16	previous value of "bad frame" flag
	prev_pdf	Word16	previous value of "pot. dangerous frame" flag
	state	Word16	ECU state (0..6)
	excEnergyHist	Word16[9]	excitation energy history
	T0_lagBuff	Word16	received pitch lag for ECU
	inBackgroundNoise	Word16	background noise flag
	voicedHangover	Word16	hangover flag
	ltpGainHistory	Word16[9]	pitch gain history
	background_state	Bgn_scdState	see below in this table
	Cb_gain_averState	Cb_gain_averageState	see below in this table
	lsp_avg_st	lsp_avgState	see below in this table
	lsfState	D_plsfState	see below in this table
	ec_gain_p_st	ec_gain_pitchState	see below in this table
	ec_gain_c_st	ec_gain_codeState	see below in this table
	pred_state	gc_predState	see table 7
	nodataSeed	Word16	seed for CN generator
	ph_disp_st	ph_dispState	see below in this table
	dtxDecoderState	dtx_decState	see below in this table
dtx_decState	since_last_sid	Word16	number of frames since last SID frame
	true_sid_period_inv	Word16	inverse of true SID update rate
	log_en	Word16	logarithmic frame energy
	old_log_en	Word16	previous value of log_en
	L_pn_seed_rx	Word32	random number generator seed
	lsp	Word16[10]	LSP vector
	lsp_old	Word16[10]	previous LSP vector
	lsf_hist	Word16[80]	LSF vector history (8 frames)
	lsf_hist_ptr	Word16	index to beginning of LSF history
	lsf_hist_mean	Word16[80]	mean-removed LSF history (8 frames)
	log_pg_mean	Word16	mean-removed logarithmic prediction gain
	log_en_hist	Word16[8]	logarithmic frame energy history
	log_en_hist_ptr	Word16	index to beginning of log, frame energy history
	log_en_adjust	Word16	mode-dependent frame energy adjustment
	dtxHangoverCount	Word16	counts down in hangover period
	decAnaElapsedCount	Word16	counts elapsed speech frames after DTX
	sid_frame	Word16	flags SID frames
	valid_data	Word16	flags SID frames containing valid data
	dtxHangoverAdded	Word16	flags hangover period at end of speech
	dtxGlobalState	enum DTXStateType	DTX state flags
	data_updated	Word16	flags CNI updates
Bgn_scdState	frameEnergyHist	Word16[60]	history of synthesis frame energy
	bgHangover	Word16	number of frames since last speech frame

Struct name	Variable	Type[Length]	Description
Cb_gain_averageState	cbGainHistory hangVar	Word16[7] Word16	codebook gain history counts length of talkspurt in subframes
	hangCount	Word16	number of subframes since last talkspurt
lsp_avgState	lsp_meanSave	Word16[10]	averaged LSP vector
D_plsfState	past_r_q past_lsf_q	Word16[10] Word16[10]	past quantized LSF prediction vector past dequantized LSF vector
ec_gain_pitchState	pbuf past_gain_pit prev_gp	Word16[5] Word16 Word16	pitch gain history previous pitch gain (limited to 1.0) previous good pitch gain
ec_gain_codeState ph_dispState	gbuf past_gain_code prev_gc gainMem prevState prevCbGain lockFull onset	Word16[5] Word16 Word16 Word16[5] Word16 Word16 Word16 Word16	codebook gain history previous codebook gain previous good codebook gain pitch gain history previously used impulse response previous codebook gain force maximum phase dispersion onset counter
Post_FilterState	res2 mem_syn_pst synth_buf agc_state preemph_state	Word16[40] Word16[10] Word16[170] agcState preemphasisState	LP residual synthesis filter memory synthesis filter work area see below in this table see below in this table
agcState	past_gain	Word16	past agc gain
preemphasisState	mem_pre	Word16	filter state
Post_ProcessState	y2_hi y2_lo y1_hi y1_lo x0 x1	Word16 Word16 Word16 Word16 Word16 Word16	filter state, upper word filter state, lower word filter state, upper word filter state, lower word filter state filter state

5 Homing procedure

The principles of the homing procedures are described in [2]. This specification only includes a detailed description of the 8 decoder homing frames. For each AMR codec mode, the corresponding decoder homing frame has a fixed set of speech parameters shown in table 9a-9h. The bit allocation within these parameters is identical to the corresponding bit allocation of the source encoder output parameters given in [2].

In the following tables, the following naming convention is used for the individual parameters. Letters in *italics* indicate numbers.

LPC_*n*	index of *n*th LSF submatrix.
LTP-LAG *m*	adaptive codebook index for subframe *m.*
LTP-GAIN *m*	adaptive codebook gain index in subframe *m.*
FCB-GAIN *m*	fixed codebook gain index in subframe *m.*
GAIN_VQ *m*	codebook gain VQ index in subframe *m* (subframe *m* and *m+1* for MR475).
POS *m_n*	position index of *n*th pulse in subframe m.
POS *m_n_k*	position index of *n*th and *k*th pulse in subframe *m.*
POS *m_n_k_l_j*	position index of *n*th, *k*th, *l*th, and *j*th pulse in subframe *m.*
SIGN *m_n_k*	sign information for *n*th and *k*th pulse in subframe *m.*
SIGN *m_ n_k_l_j*	sign information for *n*th, *k*th, *l*th, and *j*th pulse in subframe *m.*
SIGN_*m_n_k*_POS_*m_n*	sign information for *n*th and *k*th pulse and position index for *n*th pulse in subframe *m.*

Table 9a: Parameter values for the decoder homing frame (MR475)

Parameter	Value (LSB=b0)
LPC 1	0x00F8
LPC 2	0x009D
LPC 3	0x001C
LTP-LAG 1	0x0066
POS 1_1_2	0x0000
SIGN_1_1_2	0x0003
GAIN-VQ 1	0x0028
LTP-LAG 2	0x000F
POS 2_1_2	0x0038
SIGN_2_1_2	0x0001
LTP-LAG 3	0x000F
POS 3_1_2	0x0031
SIGN_3_1_2	0x0002
GAIN-VQ 3	0x0008
LTP-LAG 4	0x000F
POS 4_1_2	0x0026
SIGN_4_1_2	0x0003

Table 9b: Parameter values for the decoder homing frame (MR515)

Parameter	Value (LSB=b0)
LPC 1	0x00F8
LPC 2	0x009D
LPC 3	0x001C
LTP-LAG 1	0x0066
POS 1_1_2	0x0000
SIGN_1_1_2	0x0003
GAIN-VQ 1	0x0037
LTP-LAG 2	0x000F
POS 2_1_2	0x0000
SIGN_2_1_2	0x0003
GAIN-VQ 2	0x0005
LTP-LAG 3	0x000F
POS 3_1_2	0x0037
SIGN_3_1_2	0x0003
GAIN-VQ 3	0x0037
LTP-LAG 4	0x000F
POS 4_1_2	0x0023
SIGN_4_1_2	0x0003
GAIN-VQ 4	0x001F

Table 9c: Parameter values for the decoder homing frame (MR59)

Parameter	Value (LSB=b0)
LPC 1	0x00F8
LPC 2	0x00E3
LPC 3	0x002F
LTP-LAG 1	0x00BD
POS 1_1_2	0x0000
SIGN_1_1_2	0x0003
GAIN-VQ 1	0x0037
LTP-LAG 2	0x000F
POS 2_1_2	0x0001
SIGN_2_1_2	0x0003
GAIN-VQ 2	0x000F
LTP-LAG 3	0x0060
POS 3_1_2	0x00F9
SIGN_3_1_2	0x0003
GAIN-VQ 3	0x0037
LTP-LAG 4	0x000F
POS 4_1_2	0x0000
SIGN_4_1_2	0x0003
GAIN-VQ 4	0x0037

Table 9d: Parameter values for the decoder homing frame (MR67)

Parameter	Value (LSB=b0)
LPC 1	0x00F8
LPC 2	0x00E3
LPC 3	0x002F
LTP-LAG 1	0x00BD
POS 1_1_2_3	0x0002
SIGN_1_1_2_3	0x0007
GAIN-VQ 1	0x0000
LTP-LAG 2	0x000F
POS 2_1_2_3	0x0098
SIGN_2_1_2_3	0x0007
GAIN-VQ 2	0x0061
LTP-LAG 3	0x0060
POS 3_1_2_3	0x05C5
SIGN_3_1_2_3	0x0007
GAIN-VQ 3	0x0000
LTP-LAG 4	0x000F
POS 4_1_2_3	0x0318
SIGN_4_1_2_3	0x0007
GAIN-VQ 4	0x0000

Table 9e: Parameter values for the decoder homing frame (MR74)

Parameter	Value (LSB=b0)
LPC 1	0x00F8
LPC 2	0x00E3
LPC 3	0x002F
LTP-LAG 1	0x00BD
POS 1_1_2_3_4	0x0006
SIGN_1_1_2_3_4	0x000F
GAIN-VQ 1	0x0000
LTP-LAG 2	0x001B
POS 2_1_2_3_4	0x0208
SIGN_2_1_2_3_4	0x000F
GAIN-VQ 2	0x0062
LTP-LAG 3	0x0060
POS 3_1_2_3_4	0x1BA6
SIGN_3_1_2_3_4	0x000F
GAIN-VQ 3	0x0000
LTP-LAG 4	0x001B
POS 4_1_2_3_4	0x0006
SIGN_4_1_2_3_4	0x000F
GAIN-VQ 4	0x0000

Table 9f: Parameter values for the decoder homing frame (MR795)

Parameter	Value (LSB=b0)
LPC 1	0x00C2
LPC 2	0x00E3
LPC 3	0x002F
LTP-LAG 1	0x00BD
POS_1_1_2_3_4	0x0006
SIGN_1_1_2_3_4	0x000F
LTP-GAIN 1	0x000A
FCB-GAIN 1	0x0000
LTP-LAG 2	0x0039
POS_2_1_2_3_4	0x1C08
SIGN_2_1_2_3_4	0x0007
LTP-GAIN 2	0x000A
FCB-GAIN 2	0x000B
LTP-LAG 3	0x0063
POS_3_1_2_3_4	0x11A6
SIGN_3_1_2_3_4	0x000F
LTP-GAIN 3	0x0001
FCB-GAIN 3	0x0000
LTP-LAG 4	0x0039
POS_4_1_2_3_4	0x09A0
SIGN_4_1_2_3_4	0x000F
LTP-GAIN 4	0x0002
FCB-GAIN 4	0x0001

Table 9g: Parameter values for the decoder homing frame (MR102)

Parameter	Value (LSB=b0)
LPC 1	0x00F8
LPC 2	0x00E3
LPC 3	0x002F
LTP-LAG 1	0x0045
SIGN_1_1_5	0x0000
SIGN_1_2_6	0x0000
SIGN_1_3_7	0x0000
SIGN_1_4_8	0x0000
POS_1_1_2_5	0x0000
POS_1_3_6_7	0x0000
POS_1_4_8	0x0000
GAIN-VQ_1	0x0000
LTP-LAG 2	0x001B
SIGN_2_1_5	0x0000
SIGN_2_2_6	0x0001
SIGN_2_3_7	0x0000
SIGN_2_4_8	0x0001
POS_2_1_2_5	0x0326
POS_2_3_6_7	0x00CE
POS_2_4_8	0x007E
GAIN-VQ_2	0x0051
LTP-LAG 3	0x0062
SIGN_3_1_5	0x0000
SIGN_3_2_6	0x0000
SIGN_3_3_7	0x0000
SIGN_3_4_8	0x0000
POS_3_1_2_5	0x015A
POS_3_3_6_7	0x0359
POS_3_4_8	0x0076
GAIN-VQ_3	0x0000
LTP-LAG 4	0x001B
SIGN_4_1_5	0x0000
SIGN_4_2_6	0x0000
SIGN_4_3_7	0x0000
SIGN_4_4_8	0x0000
POS_4_1_2_5	0x017C
POS_4_3_6_7	0x0215
POS_4_4_8	0x0038
GAIN-VQ_4	0x0030

Table 9h: Parameter values for the decoder homing frame (MR122)

Parameter	Value (LSB=b0)
LPC1	0x0004
LPC2	0x002A
LPC3	0x00DB
LPC4	0x0096
LPC5	0x002A
LTP-LAG 1	0x0156
LTP-GAIN 1	0x000B
SIGN_1_1_6_POS_1_1	0x0000
SIGN_1_2_7_POS_1_2	0x0000
SIGN_1_3_8_POS_1_3	0x0000
SIGN_1_4_9_POS_1_4	0x0000
SIGN_1_5_10_POS_1_5	0x0000
POS 1_6	0x0000
POS 1_7	0x0000
POS 1_8	0x0000
POS 1_9	0x0000
POS 1_10	0x0000
FCB-GAIN 1	0x0000
LTP-LAG 2	0x0036
LTP-GAIN 2	0x000B
SIGN_2_1_6_POS_2_1	0x0000
SIGN_2_2_7_POS_2_2	0x000F
SIGN_2_3_8_POS_2_3	0x000E
SIGN_2_4_9_POS_2_4	0x000C
SIGN_2_5_10_POS_2_5	0x000D
POS 2_6	0x0000
POS 2_7	0x0001
POS 2_8	0x0005
POS 2_9	0x0007
POS 2_10	0x0001
FCB-GAIN 2	0x0008
LTP-LAG 3	0x0024
LTP-GAIN 3	0x0000
SIGN_3_1_6_POS_3_1	0x0001
SIGN_3_2_7_POS_3_2	0x0000
SIGN_3_3_8_POS_3_3	0x0005
SIGN_3_4_9_POS_3_4	0x0006
SIGN_3_5_10_POS_3_5	0x0001
POS 3_6	0x0002
POS 3_7	0x0004
POS 3_8	0x0007
POS 3_9	0x0004
POS 3_10	0x0002
FCB-GAIN 3	0x0003
LTP-LAG 4	0x0036
LTP-GAIN 4	0x000B
SIGN_4_1_6_POS_4_1	0x0000
SIGN_4_2_7_POS_4_2	0x0002
SIGN_4_3_8_POS_4_3	0x0004
SIGN_4_4_9_POS_4_4	0x0000
SIGN_4_5_10_POS_4_5	0x0003
POS 4_6	0x0006
POS 4_7	0x0001
POS 4_8	0x0007
POS 4_9	0x0006
POS 4_10	0x0005
FCB-GAIN 4	0x0000

6 File formats

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

6.1 Speech file (encoder input / decoder output)

Speech files read by the encoder and written by the decoder consist of 16-bit words where each word contains a 13-bit, left aligned speech sample. The byte order depends on the host architecture (e.g. MSByte first on SUN workstations, LSByte first on PCs etc.). Both the encoder and the decoder program process complete frames (of 160 samples) only.

This means that the encoder will only process *n* frames if the length of the input file is *n*160 + k* words, while the files produced by the decoder will always have a length of *n*160* words.

6.2 Mode control file (encoder input)

The encoder program can optionally read in a mode control file which specifies the encoding mode for each frame of speech processed. The file is a text file containing one line per speech frame. Each line contains one of the mode names from the list {`MR475`, `MR515`, `MR59`, `MR67`, `MR74`, `MR795`, `MR102`, `MR122`}.

6.3 Parameter bitstream file (encoder output / decoder input)

The files produced by the speech encoder/expected by the speech decoder contain an arbitrary number of frames in the following format.

FRAME_TYPE	B1	B2	…	B244	MODE_INFO	*unused1*	…	*unused4*

Each box corresponds to one `Word16` value in the bitstream file, for a total of 250 words or 500 bytes per frame. The fields have the following meaning:

`FRAME_TYPE`	transmit frame type, which is one of
	`TX_SPEECH (0x0000)`
	`TX_SID_FIRST (0x0001)`
	`TX_SID_UPDATE (0x0002)`
	`TX_NO_DATA (0x0003)`

`B0…B244`	speech encoder parameter bits (i.e. the bitstream itself). Each `B`*x* either has the value `0x0000` or `0x0001`. Only mode `MR122` really uses all 244 bits; for the other modes, only the first *n* bits are used (35 ≤ n ≤ 204). The remaining bits are unused (written as `0x0000`)

`MODE_INFO`	encoding mode information, which is one of
	`MR475 (0x0000)`
	`MR515 (0x0001)`
	`MR59 (0x0002)`
	`MR67 (0x0003)`
	`MR74 (0x0004)`
	`MR795 (0x0005)`
	`MR102 (0x0006)`
	`MR122 (0x0007)`

unused1…4	unused, written as `0x0000`

As indicated in section 6.1 above, the byte order depends on the host architecture.

By using a preprocessor definition the encoder output and decoder input can optionally use format described in [7], sections 5.1 and 5.3.

Annex A (informative): Change History

SMG #	Tdoc SMG	Spec	CR	Cat	PH	Vers	New Version	Subject
SA6	SP-99560	26.073					3.0.0	Approved at TSG-SA#6
SA7	SP-000025	26.073	001	A	R99	3.0.0	3.1.0	Avoidance of pulse cancellation in FCB excitation
SA11	SP-010100	26.073	003	A	R99	3.1.0	3.2.0	Correction of potential bug in AMR decoder due to usage of standard C abs() function
SA11	SP-010100	26.073	005	A	R99	3.1.0	3.2.0	Correction of comfort noise parameter interpolation bug of AMR decoder
SA11	SP-010100	26.073	007	A	R99	3.1.0	3.2.0	Correction of mode state bug in AMR decoder
SA11	SP-010100	26.073	009	A	R99	3.1.0	3.2.0	Correction of TX_TYPE and RX_TYPE identifiers
SA11	SP-010100	26.073	011	A	R99	3.1.0	3.2.0	Correction of potential bug in AMR decoder due to the usage of standard C abs() function (VAD option_2)
SA11	SP-010100	26.073	004	A	Rel-4	3.1.0	4.0.0	Correction of potential bug in AMR decoder due to usage of standard C abs() function
SA11	SP-010100	26.073	006	A	Rel-4	3.1.0	4.0.0	Correction of comfort noise parameter interpolation bug of AMR decoder
SA11	SP-010100	26.073	008	A	Rel-4	3.1.0	4.0.0	Correction of mode state bug in AMR decoder
SA11	SP-010100	26.073	010	A	Rel-4	3.1.0	4.0.0	Correction of TX_TYPE and RX_TYPE identifiers
SA11	SP-010100	26.073	012	A	Rel-4	3.1.0	4.0.0	Correction of potential bug in AMR decoder due to the usage of standard C abs() function (VAD option_2)
SA14	SP-010696	26.073	014	A	Rel-4	4.0.0	4.1.0	Correction of RX-DTX handling of NO_DATA frames in AMR decoder
SA14	SP-010697	26.073	016	A	Rel-4	4.0.0	4.1.0	Correction in AMR decoder to avoid division by zero in RX-DTX Handling
SA16					Rel-5	4.1.0	5.0.0	Version for Release 5
SA19	SP-030085	26.073	017		Rel-5	5.0.0	5.1.0	MMS compatible input/output option
SA21	SP-030444	26.073	018		Rel-5	5.1.0	5.2.0	Correction of the MMS_IO flag Note. The following line (missing in the approved CR) was added for the ANSI-C code to compile correctly: +const char sp_enc_id[] = "@(#)$Id $" sp_enc_h;
SA23	SP-040197	26.073	019		Rel-5	5.2.0	5.3.0	Correction of AMR DTX functionality
SA26					Rel-6	5.3.0	6.0.0	Version for Release 6
SA36	SP-070321	26.073	0020	1	Rel-7	6.0.0	7.0.0	Bit order of Mode Indication in AMR comfort noise frames
SA42					Rel-8	7.0.0	8.0.0	Version for Release 8
SA46					Rel-9	8.0.0	9.0.0	Version for Release 9
SA51					Rel-10	9.0.0	10.0.0	Version for Release 10
SA57					Rel-11	10.0.0	11.0.0	Version for Release 11
SA65					Rel-12	11.0.0	12.0.0	Version for Release 12

History

Document history		
V12.0.0	September 2014	Publication