ETSI TS 126 073 V17.0.0 (2022-05)

TECHNICAL SPECIFICATION

**Digital cellular telecommunications system (Phase 2+) (GSM);
Universal Mobile Telecommunications System (UMTS);
LTE;
5G;
ANSI-C code for the Adaptive Multi Rate (AMR) speech codec
(3GPP TS 26.073 version 17.0.0 Release 17)**

Reference

RTS/TSGS-0426073vh00

Keywords

5G,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 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

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 prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

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 https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

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

If you find a security vulnerability in the present document, please report it through our Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

Copyright Notification

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are 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 Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, 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.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

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. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 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

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			