



SLOVENSKI STANDARD
SIST EN 301 712 V7.4.1:2003

01-december-2003

8 [[]HJb]`WV] b]`hY`_ca i b]_UW`g_]`g]ghYa `fZUnU&ZL`E`Df]U[cX`Tj]j Y \]fcb]`
f5 AFZ[c] cf `E`?cX`5 BG=7 `nU5 AF [c] cfb]`_cXY`_f! GA`\$*`+` žfUn`]WU+`(`%ž
]nXUU%- , Ł

Digital cellular telecommunications system (Phase 2+) (GSM); Adaptive Multi Rate (AMR) speech; ANSI-C code for the AMR speech codec (GSM 06.73 version 7.4.1 Release 1998)

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[SIST EN 301 712 V7.4.1:2003](https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-5a33317ea5dd/sist-en-301-712-v7-4-1-2003)

[https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-](https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-5a33317ea5dd/sist-en-301-712-v7-4-1-2003)

[5a33317ea5dd/sist-en-301-712-v7-4-1-2003](https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-5a33317ea5dd/sist-en-301-712-v7-4-1-2003)

Ta slovenski standard je istoveten z: EN 301 712 Version 7.4.1

ICS:

33.070.50	Globalni sistem za mobilno telekomunikacijo (GSM)	Global System for Mobile Communication (GSM)
-----------	---	--

SIST EN 301 712 V7.4.1:2003

en

iTeh STANDARD PREVIEW
(standards.iteh.ai)

SIST EN 301 712 V7.4.1:2003

<https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-5a33317ea5dd/sist-en-301-712-v7-4-1-2003>

ETSI EN 301 712 V7.4.1 (2000-09)

European Standard (Telecommunications series)

**Digital cellular telecommunications system (Phase 2+);
Adaptive Multi Rate (AMR) speech;
ANSI-C code for the AMR speech codec
(GSM 06.73 version 7.4.1 Release 1998)**

iTeh STANDARD PREVIEW
(standards.iteh.ai)

GSM®

GLOBAL SYSTEM FOR
MOBILE COMMUNICATIONS

[SIST EN 301 712 V7.4.1:2003](https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-5a33317ea5dd/sist-en-301-712-v7-4-1-2003)

<https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-5a33317ea5dd/sist-en-301-712-v7-4-1-2003>



Reference

REN/TSGS-040673Q7R3

Keywords

Digital cellular telecommunications system,
Global System for Mobile communications (GSM)
AMR, ANSI C Code**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

iTeh STANDARD PREVIEW
(standards.iteh.ai)

SIST EN 301 712 V7.4.1:2003

<https://standards.iteh.ai/catalog/standards/sist/a81b499c-32fc-4949-8205-5a33317ea5dd/sist-en-301-712-v7-4-1-2003>

Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the 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://www.etsi.org/tb/status/>

If you find errors in the present document, send your comment to:
editor@etsi.fr

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.

© European Telecommunications Standards Institute 2000.
All rights reserved.

Contents

Intellectual Property Rights	4
Foreword.....	4
1 Scope	5
2 References	5
3 Definitions and abbreviations.....	6
3.1 Definitions	6
3.2 Abbreviations	6
4 C code structure.....	6
4.1 Contents of the C source code	6
4.2 Program execution.....	7
4.3 Coding style.....	7
4.4 Code hierarchy	7
4.5 Variables, constants and tables.....	11
4.5.1 Description of constants used in the C-code.....	11
4.5.2 Description of fixed tables used in the C-code	11
4.5.3 Static variables used in the C-code	13
5 Homing procedure.....	17
6 File formats	23
6.1 Speech file (encoder input / decoder output).....	23
6.2 Mode control file (encoder input).....	23
6.3 Parameter bitstream file (encoder output / decoder input)	23
Annex A (informative): Change Request History.....	24
History	25

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 (<http://www.etsi.org/ipr>).

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 European Standard (Telecommunications series) has been produced by ETSI Technical Committee Special Mobile Group (SMG).

The present document provides the bit exact definition of the Adaptive Multi Rate (AMR) speech traffic codec for the digital cellular telecommunications system.

The present document contains an electronic copy of the ANSI-C code for the GSM Adaptive Multi-Rate codec, given in the associated file "en_301712v070401o0.zip". The ANSI-C code is necessary for a bit exact implementation of the Adaptive Multi Rate speech transcoder (GSM 06.90 [3]), Voice Activity Detection (GSM 06.94 [7]), comfort noise (GSM 06.92 [5]), Discontinuous Transmission (GSM 06.93 [6]) and example solutions for substituting and muting of lost frames (GSM 06.91 [4]). The associated file "en_301712v070401o0.zip" contains a "readme.txt" file, which explains the procedure for installation and usage of the ANSI-C code files.

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

Version 7.x.y

where:

- 7 indicates Release 1998 of GSM Phase 2+.
- x the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- y the third digit is incremented when editorial only changes have been incorporated in the specification.

National transposition dates

Date of adoption of the present document:	25 August 2000
Date of latest announcement of the present document (doa):	30 November 2000
Date of latest publication of new National Standard or endorsement of the present document (dop/e):	31 May 2001
Date of withdrawal of any conflicting National Standard (dow):	31 May 2001

1 Scope

The present document contains an electronic copy of the ANSI-C code for the GSM Adaptive Multi-Rate codec. The ANSI-C code is necessary for a bit exact implementation of the Adaptive Multi Rate speech transcoder (GSM 06.90 [3]), Voice Activity Detection (GSM 06.94 [7]), comfort noise (GSM 06.92 [5]), Discontinuous Transmission (GSM 06.93 [6]) and example solutions for substituting and muting of lost frames (GSM 06.91 [4]).

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.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.
- For this Release 1998 document, references to GSM documents are for Release 1998 versions (version 7.x.y).

- [1] GSM 01.04: "Digital cellular telecommunications system (Phase 2+); Abbreviations and acronyms".
- [2] GSM 06.74: "Digital cellular telecommunications system (Phase 2+); Test sequences for the GSM Adaptive Multi-Rate (AMR) speech codec".
- [3] GSM 06.90: "Digital cellular telecommunications system (Phase 2+); Adaptive Multi-Rate (AMR) speech transcoding".
- [4] GSM 06.91: "Digital cellular telecommunications system (Phase 2+); Substitution and muting of lost frame for Adaptive Multi-Rate (AMR) speech traffic channels".
- [5] GSM 06.92: "Digital cellular telecommunications system (Phase 2+); Comfort noise aspects for Adaptive Multi-Rate (AMR) speech traffic channels".
- [6] GSM 06.93: "Digital cellular telecommunications system (Phase 2+); Discontinuous transmission (DTX) for Adaptive Multi-Rate (AMR) speech traffic channels".
- [7] GSM 06.94: "Digital cellular telecommunications system (Phase 2+); Voice Activity Detector (VAD) for Adaptive Multi-Rate (AMR) speech traffic channels".

3 Definitions and abbreviations

3.1 Definitions

Definition of terms used in the present document, can be found in GSM 06.90 [3], GSM 06.91 [4], GSM 06.92 [5], GSM 06.93 [6] and GSM 06.94 [7].

3.2 Abbreviations

For the purposes 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

For abbreviations not given in this subclause see GSM 01.04 [1].

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 the present 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 initialisation 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, tables 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
				hangover_addition
		Vad2 ¹	block_norm	
			r_fft	c_fft
			fn10Log10	Log2
			Pow2	Log2_norm
		tx_dtx_handler		
		lpc	Autocorr	
			Lag_window	
			Levinson	
		lsp	Az_lsp	Chebpps
			Q_plsf_5	Lsp_lsf
				Lsf_wt
				Vq_subvec
				Vq_subvec_s
				Reorder_lsf
				Lsf_lsp
			Int_lpc_1and3_2	Lsp_az
			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
			Int_lpc_1to3	Lsp_az
				Get_lsp_pol
		dtx_buffer	Copy	
			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	Weight_Ai	
		pre_big	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			
code_10i40_35bits	compress_code	compress10	
	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	
			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				