ETSI TS 126 104 V5.3.0 (2003-12)

Technical Specification

**Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
ANSI-C code for the floating-point
Adaptive Multi-Rate (AMR) speech codec
(3GPP TS 26.104 version 5.3.0 Release 5)**

GLOBAL SYSTEM FOR
MOBILE COMMUNICATIONS

Reference
RTS/TSGS-0426104v530

Keywords
GSM, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

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://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

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

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp .

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

This Technical Standard (TS) contains an electronic copy of the ANSI-C code for a floating-point implementation of the Adaptive Multi-Rate codec. This floating-point codec specification is mainly targeted to be used in multimedia applications such as the 3G-324M terminal specified in 3GPP TS 26.110, or in packet-based (e.g., H.323) applications. The bit-exact fixed-point ANSI-C code in 3GPP TS 26.073 remains the preferred implementation for all applications, but the floating-point codec may be used instead of the fixed-point codec when the implementation platform is better suited for a floating-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 this specification is the only standard conforming non-bit-exact implementation of the Adaptive Multi Rate speech transcoder (3GPP TS 26.090 [2]), Voice Activity Detection (3GPP TS 26.094 [6]), comfort noise generation (3GPP TS 26.092 [4]), and source controlled rate operation (3GPP TS 26.093 [5]). The floating-point code also contains example solutions for substituting and muting of lost frames (3GPP TS 26.091 [3]).

The fixed-point specification in 26.073 shall remain the only allowed implementation for the 3G mandatory speech service and the use of the floating-point codec is strictly limited to other services.

The floating-point encoder in this specification is a non-bit-exact implementation of the fixed-point encoder producing quality indistinguishable from that of the the fixed-point encoder. The decoder in this specification is functionally a bit-exact implementation of the fixed-point decoder, but the code has been optimized for speed and the standard fixed-point libraries are not used as such.

2 Normative 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 26.074: "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] 3GPP TS 26.073: "ANSI-C code for the Adaptive Multi Rate speech codec".

[8] 3GPP TS 26.101: "AMR Speech Codec Frame Structure".

[9] 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 3GPP TS 26.090 [2], 3GPP TS 26.091 [3], 3GPP TS 26.092 [4], 3GPP TS 26.093 [5], and 3GPP TS 26.094 [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 floating-point C code and provides an overview of the contents and organization of the C code attached to this document. The basic structure of the floating-point C code follows that of the bit-exact fixed-point code [7].

The C code has been verified on the following systems:

- IBM PC/AT compatible computers with Windows NT40 and Microsoft Visual C++ v.5.0 compiler;
- HP 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 files with suffix "c" contain the source code and the files with suffix "h" are the header files. The ROM data is contained in "rom" files with suffix "h".

The C code does not contain any speech coder installation verification data files. Verification for the bit-exact decoder is defined in specification 3GPP TS 26.073 [7].

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 and all the object files.

4.2 Program execution

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

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

The programs should be called like:

encoder [-dtx] mode speech_file bitstream_file

or

encoder [-dtx] -modefile=mode_file speech_file bitstream_file

decoder <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.

See the file readme.txt for more information on how to run the *encoder* and *decoder* programs.

4.3 Coding style

The C code has been written according to structuring conventions used in 3GPP TS 26.073 [7]. Encoder and decoder state structures are allocated and initialized with special initializing functions. There are no separate functions for each module, as opposed to the fixed-point implementation in 3GPP TS 26.073 [7].

4.4 Code hierarchy

The code hierarchy follows the one specified in 3GPP TS 26.073 [7].

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 neighboring cells. The time order in the call graphs is from the top downwards as the processing of a frame advances. All standard C functions, such as printf(), fwrite(), etc., have been omitted.

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

Table 1: Speech encoder call structure

Speech_Encode_Frame	Pre_Process			
	cod_amr	vad	filter_bank	first_filter_stage	
				filter5	
				filter3	
				level_calculation	
			vad_decision	complex_estimate_adapt	
				complex_vad	
				noise_estimate_update	update_cntrl
				hangover_addition	
		tx_dtx_handler			
		lpc	Autocorr		
			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	
				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
		dtx_buffer	Dotproduct40		
		dtx_enc	Lsp_lsf		
			Reorder_lsf		
			Lsf_lsp		
			Q_plsf_3	Lsp_lsf	
				Lsf_wt	
				Vq_subvec3	
				Vq_subvec4	
				Reorder_lsf	
				Lsf_lsp	
		check_lsp			
		pre_big	Weight_Ai		
			Residu		
			Syn_filt		
		ol_ltp	Pitch_ol	vad_tone_detection_update	
				Lag_max	vad_tone_detection
				comp_corr	
				hp_max	
			Pitch_ol_wgh	comp_corr	
				Lag_max_wght	vad_tone_detection_update
					vad_tone_detection
				gmed_n	
				hp_max[2]	
		vad_pitch_detection			
		subframePreProc	Weight_Ai		
			Syn_filt		
			Residu		
		cl_ltp	Pitch_fr	getRange	
				Norm_Corr	Dotproduct40
				searchFrac	Interpol_3or6
				Enc_lag3	
				Enc_lag6	
			Pred_lt_3or6		
			G_pitch	Dotproduct40	
			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			

Table 2: cbsearch call structure

cbsearch	code_2i40_9bits	cor_h_x	Dotproduct40
		set_sign	
		cor_h	Dotproduct40
		search_2i40_9bits	
		build_code_2i40_9bits	
	code_2i40_11bits	cor_h_x	Dotproduct40
		set_sign	
		cor_h	Dotproduct40
		search_2i40_11bits	
		build_code_2i40_11bits	
	code_3i40_14bits	cor_h_x	Dotproduct40
		set_sign	
		cor_h	Dotproduct40
		search_3i40	
		build_code_3i40_14bits	
	code_4i40_17bits	cor_h_x	Dotproduct40
		set_sign	
		cor_h	Dotproduct40
		search_4i40	
		build_code_4i40	
	code_8i40_31bits	cor_h_x	Dotproduct40
		set_sign12k2	Dotproduct40
		cor_h	Dotproduct40
		search_8i40	
		build_code_8i40_31bits	
		compress_code	compress10
	code_10i40_35bits	cor_h_x	Dotproduct40
		set_sign12k2	Dotproduct40
		cor_h	Dotproduct40
		search_10i40	
		build_code_10i40_35bits	
		q_p	

Table 3: gainQuant call structure

gainQuant	gc_pred	Dotproduct40	
	calc_filt_energies	Dotproduct40	
	Dotproduct40		
	MR475_update_unq_pred		
	MR475_gain_quant	gc_pred	Dotproduct40
	q_gain_code		
	MR795_gain_quant	q_gain_pitch	
		MR795_gain_code_quant3	
		calc_unfilt_energies	Dotproduct40
		gain_adapt	Gmed_n_f
		MR795_gain_code_quant_mod	
	Qua_gain		

Table 4: Speech decoder call structure

Speech_Decode_Frame	Decoder_amr	rx_dtx_handler			
		Decoder_amr_reset			
		dtx_dec	Copy		
			Lsf_lsp		
			D_plsf_3	Lsf_lsp	
			pseudonoise		
			Lsp_lsf		
			Reorder_lsf		
			Lsp_Az	Get_lsp_pol	
			A_Refl		
			Log2	Log2_norm	
			Pow2		
			Build_CN_code	pseudonoise	
			Syn_filt		
		Lsf_lsp			
		lsp_avg			
		Build_CN_param			
		D_plsf_3	Lsf_lsp		
		Int_lpc_1to3	Lsp_Az	Get_lsp_pol	
		D_plsf_5	Reorder_lsf		
			Lsf_lsp		
		Int_lpc_1and3	Lsp_Az	Get_lsp_pol	
		Dec_lag3			
		Pred_lt_3or6_40			
		Dec_lag6			
		decode_2i40_9bits			
		decode_2i40_11bits			
		decode_3i40_14bits			
		decode_4i40_17bits			
		decode_8i40_31bits	decompress_codewords	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_limited		
			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			
		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	Residu40			
		Syn_filt			
		agc	energy_new	energy_old	
			Inv_sqrt		
	Post_Process				

4.5 Variables, constants and tables

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

Word8 8 bit variable
UWord8 8 bit unsigned variable

Word16 16 bit variable
Word32 32 bit variable

Floating-point numbers use the IEEE (Institute of Electrical and Electronics Engineers) format:

Float32 8 bit exponent, 23 bit mantissa, 1 bit sign

Float64 11 bit exponent, 52 bit mantissa, 1 bit sign

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

Constants for the codec are defined in rom (h) files.

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.

Table 5: Speech encoder fixed tables

File	Table name	**Type[Length]**	Description
rom_enc.h	trackTable	Word8[4*5]	track table for algebraic code book search (MR475, MR515)
rom_enc.h	gamma1	Float32[10]	spectral expansion factors
rom_enc.h	gamma1_12k2	Float32[10]	spectral expansion factors
rom_enc.h	gamma2	Float32[10]	spectral expansion factors
rom_enc.h	b60	Float32]61]	interpolation filter coefficients
rom_enc.h	startPos1	Word16[2]	track start search position for first pulse
rom_enc.h	startPos2	Word16[4]	track start search position for second pulse
rom_enc.h	startPos	Word16[16]	track start search position
rom_enc.h	corrweight	Float32[251]	weighting of the correlation function in open loop LTP search (MR102)
rom_enc.h	qua_gain_pitch	Float32[16]	adaptive codebook gain quantization table (MR795)
rom_enc.h	qua_gain_pitch_MR122	Float32[16]	adaptive codebook gain quantization table (MR122)
rom_enc.h	qua_gain_code	Float32[64]	fixed codebook gain quantization table (MR122, MR795)
rom_enc.h	gray	Word8[8]	gray coding table
rom_enc.h	grid	Float32[61]	grid points at wich Chebyshev polynomials are evaluated
rom_enc.h	b24	Float32[25]	interpolation filter coefficients
rom_enc.h	lag_wind	Float32[10]	lag window table
rom_enc.h	lsp_init_data	Float32[10]	initialization table for lsp history in DTX
rom_enc.h	past_rq_init	Float32[80]	initialization table for the MA predictor in DTX
rom_enc.h	mean_lsf_3	Float32[10]	LSF means (not in MR122)
rom_enc.h	mean_lsf_5	Float32[10]	LSF means (MR122)
rom_enc.h	pred_fac	Float32[10]	LSF prediction factors (not in MR122)
rom_enc.h	dico1_lsf_3	Float32[3*256]	1st LSF quantizer (not in MR122 and MR795)
rom_enc.h	dico2_lsf_3	Float32[3*512]	2nd LSF quantizer (not in MR122)
rom_enc.h	dico3_lsf_3	Float32[4*512]	3rd LSF quantizer (not in MR122, MR515 and MR475)
rom_enc.h	mr515_3_lsf	Float32[4*128]	3rd LSF quantizer (MR515 and MR475)
rom_enc.h	mr795_1_lsf	Float32[3*512]	1st LSF quantizer (MR795)
rom_enc.h	dico1_lsf_5	Float32[4*128]	1st LSF quantizer (MR122)
rom_enc.h	dico2_lsf_5	Float32[4*256]	2nd LSF quantizer (MR122)
rom_enc.h	dico3_lsf_5	Float32[4*256]	3rd LSF quantizer (MR122)
rom_enc.h	dico4_lsf_5	Float32[4*256]	4th LSF quantizer (MR122)
rom_enc.h	dico5_lsf_5	Float32[4*64]	5th LSF quantizer (MR122)
rom_enc.h	table_gain_MR475	Float32[4*256]	gain quantization table (MR475)
rom_enc.h	table_gain_highrates	Float32[128*3]	gain quantization table (MR67, MR74 and MR102)
rom_enc.h	table_gain_lowrates	Float32[64*3]	gain quantization table (MR515 and MR59)
rom_enc.h	window_200_40	Float32[240]	LP analysis window (not in MR122)
rom_enc.h	window_160_80	Float32[240]	1st LP analysis window (MR122)
rom_enc.h	window_232_8	Float32[240]	2nd LP analysis window (MR122)
rom_enc.h	corrweight	Float32[251]	correlation weights
rom_enc.h	mode_dep_parm	Word8[8*9]	parameters defining the adaptive codebook search per mode

Table 6: Speech decoder fixed tables

File	Table name	**Type[Length]**	Description
rom_dec.h	dtx_log_en_adjust	Word16[9]	level adjustments for ech mode
rom_dec.h	cdown	Word32[7]	attenuation factors for codebook gain
rom_dec.h	pdown	Word32[7]	attenuation factors for adaptive codebook gain
rom_dec.h	pred	Word32[4]	algebraic code book gain MA predictor coefficients
rom_dec.h	pred_MR122	Word32[4]	algebraic code book gain MA predictor coefficients (MR122)
rom_dec.h	gamma3_MR122	Word32[10]	spectral expansion factors
rom_dec.h	gamma3	Word32[10]	spectral expansion factors
rom_dec.h	gamma4_MR122	Word32[10]	spectral expansion factors
rom_dec.h	gamma4	Word32[10]	spectral expansion factors
rom_dec.h	bitno_MR475	Word16[17]	number of bits per parameter to transmit (MR475)
rom_dec.h	bitno_MR515	Word16[19]	number of bits per parameter to transmit (MR515)
rom_dec.h	bitno_MR59	Word16[19]	number of bits per parameter to transmit (MR59)
rom_dec.h	bitno_MR67	Word16[19]	number of bits per parameter to transmit (MR67)
rom_dec.h	bitno_MR74	Word16[19]	number of bits per parameter to transmit (MR74)
rom_dec.h	bitno_MR795	Word16[23]	number of bits per parameter to transmit (MR795)
rom_dec.h	bitno_MR102	Word16[39]	number of bits per parameter to transmit (MR102)
rom_dec.h	bitno_MR122	Word16[57]	number of bits per parameter to transmit (MR122)
rom_dec.h	bitno_MRDTX	Word16[5]	number of bits per parameter to transmit (MRDTX)
rom_dec.h	qua_gain_pitch	Word32[16]	adaptive codebook gain quantization table (MR122, MR795)
rom_dec.h	qua_gain_code	Word32[96]	fixed codebook gain quantization table (MR122, MR795)
rom_dec.h	gray	Word8[8]	gray coding table
rom_dec.h	dgray	Word8[8]	gray decoding table
rom_dec.h	sqrt_table	Word32[49]	table to compute sqrt(x)
rom_dec.h	inv_sqrt_table	Word32[49]	table used in inverse square root computation
rom_dec.h	log2_table	Word32[33]	table used inbase 2 logharithm computation
rom_dec.h	pow2_table	Word32[33]	table used in 2 to the power computation
rom_dec.h	cos_table	Word32[65]	table to compute cos(x) in Lsf_lsp()
rom_dec.h	acos_slope	Word32[64]	table to compute acos(x) in Lsp_lsf()
rom_dec.h	ph_imp_low_MR795	Word32[40]	phase dispersion impulse response (MR795)
rom_dec.h	ph_imp_mid_MR795	Word32[40]	phase dispersion impulse response (MR795)
rom_dec.h	ph_imp_low	Word32[40]	phase dispersion impulse response (MR475 - MR67)
rom_dec.h	ph_imp_mid	Word32[40]	phase dispersion impulse response (MR475 - MR67)
rom_dec.h	past_rq_init	Word32[80]	initialization table for the MA predictor in DTX
rom_dec.h	mean_lsf_3	Word32[10]	LSF means (not in MR122)
rom_dec.h	mean_lsf_5	Word32[10]	LSF means (MR122)
rom_dec.h	pred_fac	Word32[10]	LSF prediction factors (not in MR122)
rom_dec.h	dico1_lsf_3	Word32[3*256]]	1st LSF quantizer (not in MR122 and MR795)
rom_dec.h	dico2_lsf_3	Word32[3*512]	2nd LSF quantizer (not in MR122)
rom_dec.h	dico3_lsf_3	Word32[4*512]	3rd LSF quantizer (not in MR122, MR515 and MR475)
rom_dec.h	mr515_3_lsf	Word32[4*128]	3rd LSF quantizer (MR515 and MR475)
rom_dec.h	mr795_1_lsf	Word32[3*512]	1st LSF quantizer (MR795)
rom_dec.h	dico1_lsf_5	Word32[4*128]	1st LSF quantizer (MR122)
rom_dec.h	dico2_lsf_5	Word32[4*256]	2nd LSF quantizer (MR122)
rom_dec.h	dico3_lsf_5	Word32[4*256]	3rd LSF quantizer (MR122)
rom_dec.h	dico4_lsf_5	Word32[4*256]	4th LSF quantizer (MR122)
rom_dec.h	dico5_lsf_5	Word32[4*64]	5th LSF quantizer (MR122)
rom_dec.h	table_gain_MR475	Word32[4*256]	gain quantization table (MR475)
rom_dec.h	table_gain_highrates	Word32[128*4]	gain quantization table (MR67, MR74 and MR102)
rom_dec.h	table_gain_lowrates	Word32[64*4]	gain quantization table (MR515 and MR59)
rom_dec.h	inter_6	Word32[61]	interpolation filter coefficients
rom_dec.h	window_200_40	Word32[240]	LP analysis window (not in MR122)
rom_dec.h	table_speech_bad	UWord8[9]	comparison optimisation table in DTX
rom_dec.h	table_SID	Uword8[9]	comparison optimisation table in DTX
rom_dec.h	table_DTX	Uword8[9]	comparison optimisation table in DTX
rom_dec.h	table_mute	Uword8[9]	comparison optimisation table in DTX

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	Word32	Is set if DTX functionality is used
Pre_ProcessState	y2	Float32	filter state
	y1	Word16 Float32	filter state
	x0	Float32	filter state
	x1	Float32	filter state
cod_amrState	old_speech	Float32 [320]	speech buffer
	speech	Float32*	pointer to current frame in old_speech
	p_window	Float32*	pointer to LPC analysis window in old_speech
	p_window_12k2	Float32*	pointer to LPC analysis window with no lookahead in old_speech (MR122)
	new_speech	Float32*	pointer to the last 160 speech samples in old_speech
	old_wsp	Float32 [303]	buffer holding spectral weighted speech
	wsp	Float32*	pointer to the current frame in old_wsp
	old_lags	Word32[5]	open loop LTP states
	ol_gain_flg	Float32 [2]	enables open loop pitch lag weighting (MR102)
	old_exc	Float32 [314]	excitation vector
	exc	Float32*	current excitation
	ai_zero	Float32 [51]	history of weighted synth. filter followed by zero vector
	zero	Float32*	zero vector
	h1	Float32*	impulse response of weighted synthesis filter
	hvec	Float32 [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	vadState	see below in this table
	vadSt2	vadState2	see below in this table
	dtx	Word32	is set if DTX functionality is used
	dtx_encSt	dtx_encState	see below in this table
	mem_syn	Float32 [10]	synthesis filter memory
	mem_w0	Float32 [10]	weighting filter memory (applied to error signal)
	mem_w	Float32 [10]	weighting filter memory (applied to input signal)
	mem_err	Float32 [50]	filter memory for production of error vector
	error	Float32*	error signal (input minus synthesized speech)
	sharp	Float32	pitch sharpening gain
vadState	bckr_est	Float32 [9]	background noise estimate
	ave_level	Float32 [9]	averaged input components for stationary estimation
	old_level	Float32 [9]	input levels of the previous frame
	sub_level	Float32 [9]	input levels calculated at the end of a frame (lookahead)
	a_data5	Float32 [6]	memory for the filter bank
	a_data3	Float32 [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	Word32	15 flags for intermediate VAD decisions
	pitch	Word32	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	Word32	variables for pitch detection
	oldlag	Word32	variables for pitch detection
	complex_hang_count	Word16	complex hangover counter, used by VAD
	complex_hang_timer	Word16	hangover initiator, used by CAD

Struct name	Variable	Type[Length]	Description
	best_corr_hp	Float32	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
dtx_encState	lsp_hist	Float32[80]	LSP history (8 frames)
	log_en_hist	Float32 [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	Word32	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	Float32[11]	last frames direct form coefficients
lspState	lsp_old	Float32 [10]	old LSP vector
	lsp_old_q	Float32 [10]	old quantized LSP vector
	qSt	Q_plsfState	see below in this table
Q_plsfState	past_rq	Float32[10]	past quantized LSF prediction error
clLtpState	pitchSt	Pitch_frState	see below in this table
tonStabState	count	Word16	count consecutive (potential) resonance frames
	gp	Float32[7]	pitch gain history
Pitch_frState	T0_prev_subframe	Word32	integer. pitch lag of previous subframe
gainQuantState	sf0_ gcode0	Float32	subframe 0/2 codebook gain
	sf0_ target_en	Float32	subframe 0/2 target energy
	sf0_ coeff	Float32 [5]	subframe 0/2 energy coefficient
	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	Float32[4]	MA predictor memory (20*log10(pred. error))
GainAdaptState	onset	Word16	onset counter
	prev_alpha	Float32	previous adaptor output
	prev_gc	Float32	previous codebook gain
	ltpg_mem	Float32 [5]	pitch gain history
pitchOLWghtState	old_T0_med	Word32	weighted open loop pitch lag
	ada_w	Float32	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
Decoder_amrState	old_exc	Word32[194]	excitation vector
	exc	Word32*	current excitation
	lsp_old	Word32[10]	LSP vector of previous frame
	mem_syn	Word32[10]	synthesis filter memory
	sharp	Word32	pitch sharpening gain
	old_T0	Word32	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	Word32[9]	excitation energy history
	T0_lagBuff	Word32	received pitch lag for ECU
	inBackgroundNoise	Word32	background noise flag
	voicedHangover	Word32	hangover flag
	ltpGainHistory	Word32[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	Word32	logarithmic frame energy
	old_log_en	Word32	previous value of log_en
	pn_seed_rx	Word32	random number generator seed
	lsp	Word32[10]	LSP vector
	lsp_old	Word32[10]	previous LSP vector
	lsf_hist	Word32[80]	LSF vector history (8 frames)
	lsf_hist_ptr	Word16	index to beginning of LSF history
	lsf_hist_mean	Word32[80]	mean-removed LSF history (8 frames)
	log_pg_mean	Word16	mean-removed logarithmic prediction gain
	log_en_hist	Word32[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	Word32[60]	history of synthesis frame energy
	bgHangover	Word16	number of frames since last speech frame
Cb_gain_averageState	cbGainHistory	Word32[7]	codebook gain history
	hangVar	Word16	counts length of talkspurt in subframes
	hangCount	Word16	number of subframes since last talkspurt
lsp_avgState	lsp_meanSave	Word32[10]	averaged LSP vector
D_plsfState	past_r_q	Word32[10]	past quantized LSF prediction vector
	past_lsf_q	Word32[10]	past dequantized LSF vector
ec_gain_pitchState	pbuf	Word32[5]	pitch gain history
	past_gain_pit	Word32	previous pitch gain (limited to 1.0)
	prev_gp	Word32	previous good pitch gain
ec_gain_codeState	gbuf	Word32[5]	codebook gain history
	past_gain_code	Word32	previous codebook gain
	prev_gc	Word32	previous good codebook gain
ph_dispState	gainMem	Word32[5]	pitch gain history
	prevState	Word32	previously used impulse response
	prevCbGain	Word32	previous codebook gain
	lockFull	Word16	force maximum phase dispersion
	onset	Word16	onset counter
Post_FilterState	res2	Word32[40]	LP residual
	mem_syn_pst	Word32[10]	synthesis filter memory
	synth_buf	Word16[170]	synthesis filter work area
	agc_state	agcState	see below in this table
	preemph_state	preemphasisState	see below in this table

Struct name	Variable	Type[Length]	Description
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	Word32 Word32 Word32 Word32 Word32 Word32	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 3GPP TS 06.090 [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 3GPP TS 06.090 [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 [2] 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 format described in RFC 3267 [9], sections 5.1 and 5.3.

By using preprocessor definition encoder/decoder can optionally use AMR Interface Format 2. The format is described in TS 26.101 [8] Annex A.

By using another preprocessor definition encoder/decoder can optionally use format compatible with the existing AMR fixed-point C-code. Frame format is following.

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.

Annex A (informative): Change History

TSG SA#	Tdoc	CR	Rev	Cat	PH	Vers	New Vers	Subject
10	SP-000577	002		A	Rel-4	3.0.0	4.0.0	AMR Core Frame bit ordering (AMR speech Codec; Floating point C-Code
12	SP-010306	004	1	A	Rel-4	4.0.0	4.1.0	Limiting predicted codebook gain computing in encoder
12	SP-010306	006	1	A	Rel-4	4.0.0	4.1.0	Correction of decoder operation in error concealment of lost frames
12	SP-010306	008	1	A	Rel-4	4.0.0	4.1.0	Correction of mode state bug in AMR decoder
12	SP-010306	012	1	A	Rel-4	4.0.0	4.1.0	Correction of decoder Reset
12	SP-010306	014	1	A	Rel-4	4.0.0	4.1.0	Correction of comfort noise parameter interpolation bug of AMR decoder
12	SP-010306	016	1	A	Rel-4	4.0.0	4.1.0	Correction of the TX_TYPE and RX_TYPE identifiers
	MCC				Rel-4	4.1.0	4.1.1	Correction of bugs in code
13	SP-010452	010	1	A	Rel-4	4.1.1	4.2.0	Correction to make encoder and decoder memories independent
13	SP-010452	018		A	Rel-4	4.1.1	4.2.0	Correction of decoder operation in error concealment of lost frames
15	SP-020079	019		A	Rel-4	4.2.0	4.3.0	Maintaining bit-exactness with TS 26.073
16							5.0.0	Version for Release 5
19	SP-030088	21	1	F	Rel-5	5.0.0	5.1.0	MMS compatible i/o format option
19	SP-030088	24		A	Rel-5	5.0.0	5.1.0	Correction to floating-point implementation of sp_dec.c
20	SP-030214	26		A	Rel-5	5.1.0	5.2.0	Correction on codec mode handling during DTX
22	SP-030681	29	1	F	Rel-5	5.2.0	5.3.0	Correction on the implementation of the interface of decoder.c

History

Document history		
V5.0.0	June 2002	Publication
V5.1.0	March 2003	Publication
V5.2.0	June 2003	Publication
V5.3.0	December 2003	Publication