Application Report
SPRA624 - February 2000

GSM Mixed-Rate Voice Coding on the TMS320C62x™ DSP

Peter Dent

Wireless Communications

ABSTRACT

This document describes how to link together the GSM full-rate (FR), half-rate (HR), enhanced full-rate (EFR) and adaptive multi-rate (AMR) voice coding implementations on the TMS320C62x™ digital signal processor (DSP).

Contents

List of Tables

List of Examples

1 Introduction

This document describes how to link together the GSM full-rate (FR), half-rate (HR), enhanced full-rate (EFR) and adaptive multi-rate (AMR) voice-coding implementations on the TMS320C62x™ digital signal processor (DSP).

This document does not describe the GSM FR, HR, EFR, or AMR voice-coding algorithms. For more detailed information on these algorithms, see the respective application reports. [1][2][3][4] This document contains additional information for mixed voice coder (vocoder) implementation.

2 Files

The host.c, host.h, and typedef.h files are common to all implementations of mixed-rate voice coding.

Each vocoder must be separately and partially linked to produce relinkable object files. The link command file that is required depends on whether the old GSM vocoder format or the new eXpressDSP™ vocoder format is used in the final application. The partial link command file to use is shown in Table 1.

Table 1. Link Command Files

Algorithm	Old GSM Format	XDSP Format
Full-rate	partial_link_fr.cmd	gsmfr_ti.cmd
Half-rate	partial_link_hr.cmd	gsmhr_ti.cmd
Enhanced-full-rate	partial_link_efr.cmd	gsmefr_ti.cmd
Adaptive-multi-rate	partial_link_amr.cmd	gsmamr_ti.cmd

These algorithms produce the object files necessary to complete the link and remove local variables that may have common names but are not related.

3 Old Mixed-Rate Vocoder Files

The functions of old mixed-rate vocoder files are explained here.

3.1 Gsm.h

This is a C header file that defines the subroutines that are still external when the vocoders are linked with the partial link command files.

3.2 Gsm.c

This C file reads the Cont_Dual.gsm command file and controls the application of test patterns to the vocoders. This program is used with the *load6x* command to test the vocoders on the simulator. Data input/output is controlled via the channel input/output (CIO). Since the data sections of all four vocoders are overlaid to reduce memory requirements, only memory for the largest (AMR) vocoder must be assigned. (FR_Assign_GSM, HR_Assign_GSM, and EFR_Assign_GSM are not called).

4 New Mixed-Channel Functions

New mixed-channel vocoder functions are explained here.

4.1 Set_Channel (Address);

This function points the data page pointer (DP) register to the local (.bss) section.

4.2 Return_Channel ( );

This function restores the DP register to the default (.bss) section.

4.3 Handle_Ints ( );

This NULL function restores the DP register and handles interrupts in the middle of voice coding code. Although not actually used, this function may be inserted in the C-code to allow more interrupts, if needed.

4.4 Linker Notes (dual_rate.cmd)

Because only one vocoder runs on a particular channel, there is no need to allocate data memory for all rates. The small memory models (FR, HR, and EFR) can overlay the largest memory model (AMR) with the UNION linker directive, as shown in Example 1.

Example 1. UNION Linker Directive

```
UNION:run = BMEM
  {
    .bss:part1:
    {
      fr_pass\Partial_Link_FR.obj (.bss)
    }
    .bss:part2:
    {
      hr–optimized–pass\Partial_Link_HR.obj (.bss)
    }
    .bss:part3:
    {
      efr_pass\Partial_Link_EFR.obj (.bss)
    }
    .bss:part4:
    {
      amr_pass\Partial_Link_AMR.obj (.bss)
    }
  }
```

5 GSM eXpressDSP Mixed-Rate Functions

This section contains detailed specifics of the GSM vocoder algorithm within the eXpressDSP environment. Further details on eXpressDSP functions can be found in the TI *eXpressDSP Algorithm Standard API Reference*[5].

5.1 Standard eXpressDSP Functions

The functions shown in Table 2 are common to FR (full-rate), HR (half-rate), EFR (enhanced full-rate), and AMR (adaptive multi-rate) algorithms.

Table 2. Standard eXpressDSP Functions

Function	Implementation	Comments
*implementationId	Mandatory – Implemented	See the TI *eXpressDSP Algorithm Standard API Reference.*
algActivate	Optional – Not implemented	NULL
algAlloc	Mandatory – Implemented	The parentFxns field is ignored. 2 memTabs
algControl	Optional – Implemented	Performs channel reset and mode selection. (See Section 5.1.1, *algControl.*)
algDeactivate	Optional – Not implemented	NULL
algFree	Mandatory – Implemented	See the TI *eXpressDSP Algorithm Standard API Reference.* 2 memTabs
algInit	Mandatory – Implemented	See the TI *eXpressDSP Algorithm Standard API Reference.* Resets coder (DTX disabled) + decoder
algMoved	Optional – Implemented	See the TI *eXpressDSP Algorithm Standard API Reference.* Changes algorithm history pointer(s)
algNumAlloc	Optional – Implemented	Returns 2

5.1.1 algControl

This function implements algorithm control. Status is unused, and the command word is divided into bit fields, as shown in Table 3.

Table 3. Algorithm Control Functions

Bit Field	Value	Function	FR	HR	EFR	AMR
1 … 0	00 01 10 11	No action No action Reset encoder, DTX disabled Reset encoder, DTX enabled	Yes	Yes	Yes	Yes
2	0 1	No action Reset decoder	Yes	Yes	Yes	Yes
31 … 3	0	Unused	N/A	N/A	N/A	N/A

5.2 Extended eXpressDSP Functions

Extended eXpressDSP functions, shown in Table 4, are common to FR, HR, EFR, and AMR algorithms.

Table 4. Extended eXpressDSP Functions

Function	Description
GSM_SpeechEncoder1	Code frame parameters for vocoder
GSM_SpeechEncoder2	Code subframe 1 parameters for vocoder
GSM_SpeechEncoder3	Code subframe 2 parameters for vocoder
GSM_SpeechEncoder4	Code subframe 3 parameters for vocoder
GSM_SpeechEncoder5	Code subframe 4 parameters for vocoder
GSM_SpeechDecoder1	Decode frame and subframe 1 parameters for vocoder
GSM_SpeechDecoder2	Decode subframe 2 parameters for vocoder
GSM_SpeechDecoder3	Decode subframe 3 parameters for vocoder
GSM_SpeechDecoder4	Decode subframe 4 parameters for vocoder

In all cases, the format of these functions is the same, for example:

```
Void  (*GSM_SpeechVocoder)(IGSM_Handle handle, Short pswIn[], Short pswOut[]);
```

Where:

GSM_SpeechVocoder is the partial encoder or partial decoder.

Handle is the same as that used in the standard XDSP functions.

pswin is the pointer to the first data word of input data for the frame.

pswOut(coders) is the pointer to the first data word of output data for the frame.

pswOut(decoders) is the pointer to the first data word of output data for the subframe.

More details of these functions can be found in the algorithm-specific application notes. However, note that for the coders, the pointers always point to the beginning of the *frame* data; they do not point to the local *subframe* data within the frame. For the decoders, the output is to the subframe.

5.3 Algorithm Files

For each algorithm, ** represents FR, HR, EFR, or AMR, respectively. These files are needed to generate the relinkable modules. Table 5 gives the algorithm-specific files and Table 6 gives the algorithm-common files.

Table 5. Algorithm-Specific Files

Algorithm	Description
igsm.h	Specifies standard extensions to ialg.h for the 5 parts of the coder and the 4 parts of the decoder. It is common to all algorithms and the control program, and is used to link the algorithm to the control program.
gsm**_ti.h	Extension of common GSM functions for type-safe application to the specific algorithm
gsm**_ti_priv.h	Declaration of internal functions for linking implementation to indirection table
gsm**_ti_ialg.c	Internal implementation of both standard and extended eXpressDSP functions
gsm**_ti_ialgvt.c	Indirection tables for both standard and extended eXpressDSP functions
gsm**_ti.cmd	Link control file for making a relinkable object file
gsm**_ti.mak	Code Composer Studio™ make file for compiling and creating a relinkable object file

Table 6. Algorithm-Common Files

Algorithm	Description
igsm.h	Specifies standard extensions to ialg.h for the 5 parts of the coder and the 4 parts of the decoder. It is common to all algorithms and the control program, and is used to link the algorithm to the control program.
igsm.c	Implements the common IGSM_PARAMS structure, which is used unmodified by all four algorithms
gsm.h	Provides an external inline GSM vocoder-independent implementation of standard eXpressDSP functions (create, delete, init, exit, and control), as well as the 9 subprocessing functions

Example 2. Algorithm Code

```
static inline GSM_Handle GSM_create(const IGSM_Fxns *fxns, const GSM_Params *prms)
static inline Void GSM_delete(GSM_Handle handle)
static inline Void GSM_init(Void)
static inline Void GSM_exit(Void)
static inline Int GSM_control(GSM_Handle gsm,IALG_Cmd cmd, IALG_Status *status)
static inline Void GSM_SpeechEncoder1(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechEncoder2(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechEncoder3(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechEncoder4(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechEncoder5(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechDecoder1(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechDecoder2(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechDecoder3(GSM_Handle gsm, Short in[], Short out[])
static inline Void GSM_SpeechDecoder4(GSM_Handle gsm, Short in[], Short out[])
```

Code Composer Studio is a trademark of Texas Instruments.

5.3.1 Reassigning Algorithms

If an algorithm is created and deleted using the algorithm with the largest data structure, it may be reassigned to an algorithm with a smaller structure and reset with GSM_control to simplify memory management, as shown in Table 7.

Table 7. Reassigning Algorithms

Create/Deleted With	FR	HR	EFR	AMR
FR	Yes	No	No	No
HR	Yes	Yes	No	No
EFR	Yes	Yes	Yes	No
AMR	Yes	Yes	Yes	Yes

For example, a FR algorithm can use the HR memory allocation, but a HR algorithm cannot use the FR memory allocation.

The indirection to the algorithm extension and control functions provides inline common *create* and *delete* functions for all four algorithms.

5.3.2 Example Files

Algorithm example files and memory management are discussed here.

- gsm_XDSP_evm_test3.c

  This is an example of an implementation file that implements a multichannel mixed-rate vocoder for FR, HR, and EFR running on a c6201 EVM board that processes frames received via the host interface port (HPI) of the c6201. It also includes some examples of basic memory management functions for create and delete that would normally be part of an operating system. The full system can be generated with the Code Composer™ make file, gsm_xdsp3.mak.

- gsm_XDSP_evm_test2.c

  This is an example of an implementation file that implements a multichannel mixed-rate vocoder for FR and AMR running on a c6201 EVM board that processes frames received via the HPI port of the c6201. It also includes some examples of basic memory management functions for create and delete that would normally be part of an operating system. The full system can be generated with the Code Composer make file, gsm_xdsp2.mak.

- PC_Host SW

  This directory contains a Borland C++ program that applies the European Telecommunications Standards Institute (ETSI) test vectors to the voice coders via the HPI on the EVM.

Code Composer is a trademark of Texas Instruments.

6 Host Files

These host files are common to both mixed-rate implementations and they run on the host PC.

6.1 Gsm_hpi.c

This C file receives control information through the HPI on a C6x DSP EVM. This information is in turn provided by a host program running on the PC (HPI_PC_HOST) that reads the command file Cont_Dual.gsm. Because the data sections of all four vocoders are overlaid to reduce memory requirements, only memory for the largest (EFR) vocoder must be assigned. (FR_Assign_GSM and HR_Assign_GSM are not called).

6.2 Cont_Dual.gsm

This file tells the CIO how to handle all of the test pattern files. For each test sequence, there are 6 lines.

```
PCM data input file

GSM data output File

GSM data input file

PCM data output file

dtx| nodtx

HALF | FULL | EFR | MR122 | MR102 | MR795 | MR74 | MR67 | MR59 | MR515 | MR475 | AMR

Mode (exists only with AMR voice coding)
```

The first 4 lines are the filenames of the input and output files. The fifth line indicates whether that code sequence is to be run with voice activity and discontinuous transmission enabled or disabled (dtx or nodtx, respectively). The 6th line indicates the GSM data rate, and this code can be any of the selections listed in line 6 above.

With AMR voice coding, there is an additional 7th line containing the filename that lists the mode to be used on a frame-by-frame basis.

This structure is repeated for the number of test sequences that must be run; after all test sequences have run, the C program exits.

References

1. TI *GSM Half-Rate Voice Coding on the TMS320C62xx DSP* (SPRA582)
2. TI *GSM Full-Rate Voice Coding on the TMS320C62xx DSP* (SPRA583)
3. TI *GSM Enhanced Full-Rate Voice Coding on the TMS320C62xx DSP* (SPRA565)
4. TI *GSM Adaptive Multi-Rate Voice Coding on the TMS320C62xx DSP* (SPRA625)
5. TI *eXpressDSP Algorithm Standard API Reference* (SPRU360)