Embedded SDK (Software Development Kit)

G.723.1A Speech Codec Library

SDK138/D
Rev. 1, 07/19/2002

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Contents

About This Document

Audience ix
Organization ix
Suggested Reading ix
Conventions x
Definitions, Acronyms, and Abbreviations x
References xi

Chapter 1
Introduction

1.1 Quick Start 1-1
1.2 Overview of G.723.1A Dual Rate Speech Coder 1-1
1.2.1 Background 1-2
1.2.2 Features and Performance 1-3

Chapter 2
Directory Structure

2.1 Required Core Directories 2-1
2.2 Optional (Domain-Specific) Directories 2-2
2.3 Demo Directory Structure 2-3

Chapter 3
G.723.1A Codec Library Interfaces

3.1 G.723.1A Codec Services 3-1
3.2 Interface 3-1
3.3 Specifications 3-8
3.3.1 Init_Coder 3-9
3.3.2 Init_Vad 3-11
3.3.3 Init_Cod_Cng 3-13
3.3.4 Coder 3-15
3.3.5 Init_Decod 3-18
3.3.6 Init_Dec_Cng 3-20
3.3.7 Decod 3-22

Chapter 4
Building the G.723.1A Codec Library

4.1 Building the G.723.1A Codec Library 4-1

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 5
Linking Applications with the G.723.1A Codec Library

5.1 G.723.1A Codec Library 5-1

Chapter 6
G.723.1A Applications

6.1 Test and Demo Applications. 6-1

Chapter 7
License

7.1 Limited Use License Agreement 7-1

List of Tables

Table 3-1 Init_Coder Arguments 3-9
Table 3-2 Init_Vad Arguments 3-11
Table 3-3 Init_Cod_Cng Arguments 3-13
Table 3-4 Coder Arguments 3-15
Table 3-5 Init_Decod Arguments 3-18
Table 3-6 Init_Dec_Cng Arguments 3-20
Table 3-7 Decod Arguments 3-22

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Figures

Figure 2-1 Core Directories 2-1
Figure 2-2 DSP56858 Directories 2-2
Figure 2-3 G723.1A Dual Rate Speech Codec Directory Structure. 2-3
Figure 2-4 SDK Application Directory Structure. 2-4
Figure 2-5 g723 Directory Structure 2-4

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Examples

Code Example 3-1 C Header File g723.h 3-1
Code Example 3-2 Use of the Init_Coder Interface 3-10
Code Example 3-3 Use of the Init_Vad Interface 3-12
Code Example 3-4 Use of the Init_Cod_Cng Interface 3-14
Code Example 3-5 Use of Coder Interface 3-16
Code Example 3-6 Use of the Init_Decod Interface 3-19
Code Example 3-7 Use of the Init_Dec_Cng Interface 3-21
Code Example 3-8 Use of Decod Interface 3-22
Code Example 5-1 linker.cmd File 5-2

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

About This Document

This manual describes the G.723.1A Dual Rate Speech Coder with Annex A Silence Compression Scheme, Codec algorithm for use with Motorola's Embedded Software Development Kit, (SDK).

Audience

This document targets software developers implementing G.723.1A Codec within software applications.

Organization

This manual is arranged in the following sections:

- **Chapter 1, Introduction**--provides a brief overview of this document
- **Chapter 2, Directory Structure**--provides a description of the required core directories
- **Chapter 3, G.723.1A Codec Library Interfaces**--describes all of the G.723.1A Library functions
- **Chapter 4, Building the G.723.1A Codec Library**--tells how to execute the system library project build
- **Chapter 5, Linking Applications with the G.723.1A Codec Library**--describes the organization of the G.723.1A Library
- **Chapter 6, G.723.1A Applications**--describes the use of the G.723.1A Library through test/demo applications
- **Chapter 7, License**--provides the license required to use this product

Suggested Reading

We recommend that you have a copy of the following references:

- *DSP56800E Family Manual*, DSP56800ERM/D
- *DSP5685x User's Manual*, DSP5685xUM/AD
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Code examples	`//Process command for line flash`
Italic	Directory names, project names, calls, functions, statements, procedures, routines, arguments, file names, applications, variables, directives, code snippets in text	...and contains these core directories: *applications* contains applications software... ...CodeWarrior project, *3des.mcp* is... ...the *pConfig* argument.... ...defined in the C header file, *aec.h*....
Bold	Reference sources, paths, emphasis	...refer to the **Targeting DSP56F80x Platform** manual.... ...see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
Blue Text	Linkable on-line	...refer to Chapter 7, License....
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}
ALL CAPITAL LETTERS	# defines/ defined constants	# define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]
Quotation marks, "..."	Returned messages	...the message, "Test Passed" is displayed.... ...if unsuccessful for any reason, it will return "NULL"...

Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document. As this template develops, this list will be generated from the document. As we develop more group resources, these acronyms will be easily defined from a common acronym dictionary. Please note that while the acronyms are in solid caps, terms in the definition should be initial capped ONLY IF they are trademarked names or proper nouns.

ACELP	Algebraic-Code-Excited Linear Prediction
CNG	Comfort Noise Generator
DSP	Digital Signal Processor or Digital Signal Processing
I/O	Input/Output
IDE	Integrated Development Environment
LPC	Linear Prediction Coder
LSB	Least Significant Byte
MIPS	Million Instructions Per Second
MP-MLQ	Multipluse Maximum Likelihood Quantization
MSB	Most Significant Byte
OnCE™	On-Chip Emulation
OMR	Operating Mode Register
PC	Program Counter
PSVQ	Predictive Split Vector Quantizer
SDK	Software Development Kit
SP	Stack Pointer
SPI	Serial Peripheral Interface
SR	Status Register
SRC	Source
VAD	Voice Activity Detector

References

The following sources were referenced to produce this book:

[1] *DSP56800E Reference Guide*, DSP56800ERM/D
[2] *DSP5685x User's Manual*, DSP5685xUM/AD
[3] *Embedded SDK Programmer's Guide*, SDK101/D
[4] *ITU-T Recommendation G.723.1*
[5] *ITU-T Recommendation G.723.1 Annex A*

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 1
Introduction

Welcome to Motorola's Family of Digital Signal Processors, (DSPs). This document describes the G.723.1A Dual Rate Speech Coder Library, which is a part of Motorola's comprehensive Software Development Kit, (SDK), for its DSPs. In this document, you will find all the information required to use and maintain the G.723.1A Dual Rate Speech Coder Library interface and algorithms.

Motorola provides these algorithms to you for use on the Motorola Digital Signal Processors to expedite your application development and reduce the time it takes to bring your own products to market.

Motorola's G.723.1A Dual Rate Speech Coder Library is licensed for your use on Motorola processors. Please refer to the standard Software License Agreement in **Chapter 7** for license terms and conditions; please consult with your Motorola representative for premium product licensing.

1.1 Quick Start

Motorola's Embedded SDK is targeted to a large variety of hardware platforms. To take full advantage of a particular hardware platform, use **Quick Start** from the appropriate **Targeting Motorola DSP568xx Platform** documentation.

For example, the **Targeting Motorola DSP56852 Platform** manual provides more specific information and examples about this hardware architecture. If you are developing an application for a DSP56852EVM board or any other DSP56852 development system, refer to the **Targeting Motorola DSP56852 Platform** manual for **Quick Start** or other DSP56852-specific information.

1.2 Overview of G.723.1A Dual Rate Speech Coder

The ITU-T Recommendation G.723.1 Dual Rate Speech Coder provides a coded representation used for compressing speech in either of two very low bit rates. The higher 6.3 Kbits/s rate produces a greater quality reproduction, while the lower 5.3 Kbits/s rate provides system designers additional flexibility. Both rates are mandatory, and it is possible to switch between the rates on any 30ms frame boundary. Annex A of this Recommendation (G.723.1A) provides an optional silence compression system used to reduce the transmitted bit rate during silent intervals of speech.

This coder was optimized to represent speech with a high quality of the two rates using a limited amount of complexity. Music and other audio signals are not represented as faithfully as speech, but can be compressed and decompressed using this coder.

1.2.1 Background

The ITU-T G.723.1 Recommendation describes an algorithm for coding speech in frames using a linear predictive analysis-by-synthesis technique. The excitation signal for the higher 6.3kBps rate is Multipulse Maximum Likelihood Quantization (MP-MLQ) and for the low rate coder is Algebraic-Code-Excited Linear-Prediction (ACELP).

This coder encodes speech or other audio signals in 30ms frames. In addition, there is a look-ahead of 7.5ms, resulting in a total algorithmic delay of 37.5ms. All additional delays in the implementation and operation of this coder are due to processing delays of the implementation, transmission delays in the communication link, and buffering delays of the multiplexing protocol.

This coder is designed to operate with a digital signal obtained by first performing telephone bandwidth filtering (Recommendation G.712) of the analog input, sampling at 8000Hz, then converting to 16-bit linear PCM for the input to the encoder. The output of the decoder should be converted back to analog by similar means. Other input/output characteristics, such as those specified by Recommendation G.711 for 64kBps PCM data, should be converted to 16-bit linear PCM before encoding or from 16-bit linear PCM to the appropriate format after decoding. The bitstream from the encoder to the decoder is defined within the ITU-T G.723.1 Recommendation.

The coder is based on the principles of linear prediction analysis-by-synthesis coding and attempts to minimize a perceptually weighted error signal. The encoder operates on blocks (frames) of 240 samples each (30ms). Each block is first high-pass filtered to remove the DC component, then divided into four subframes of 60 samples each. For every subframe, a 10th order Linear Prediction Coder (LPC) filter is computed using the unprocessed input signal. The LPC filter for the last subframe is quantized using a Predictive Split Vector Quantizer (PSVQ). The unquantized LPC coefficients are used to construct the short-term perceptual weighting filter, which filters the entire frame and obtains the perceptually weighted speech signal.

For every two subframes (120 samples), the open loop pitch period, L_{OL}, is computed using the weighted speech signal. This pitch estimation is performed on blocks of 120 samples. The pitch period is searched in the range from 18 to 142 samples.

From this point, the speech is processed on a 60 samples per subframe basis.

Using the estimated pitch period computed previously, a harmonic noise shaping filter is constructed. The combination of the LPC synthesis filter, the formant perceptual weighting filter, and the harmonic noise shaping filter is used to create an impulse response. The impulse response is then used for further computations.

Using the pitch period estimation, the L_{OL}, and the impulse response, a closed loop pitch predictor is computed. A fifth order pitch predictor is used. The pitch period is computed as a small differential value around the open loop pitch estimate. The contribution of the pitch predictor is then subtracted from the initial target vector. Both the pitch period and the differential value are transmitted to the decoder.

Finally, the non-periodic component of the excitation is approximated. As explained previously, MP-MLQ excitation is used for the high bit rate and ACELP is used for the low bit rate .

The decoder operation is also performed on a frame-by-frame basis. First, the quantized LPC indices are decoded, then the decoder constructs the LPC synthesis filter. For every subframe, both the adaptive codebook excitation and fixed codebook excitation are decoded and input to the synthesis filter. The adaptive post filter consists of a formant and a forward-backward pitch post filter. The excitation signal is

input to the pitch post filter, which in turn is input to the synthesis filter. The output of the synthesis filter is input to the pitch post filter, which in turn is input to the synthesis filter, whose output is input to the formant post filter. A gain scaling unit maintains the energy at the input level of the formant post filter.

The ITU-T G.732.1 Annex A is the silence compression system for the G.723.1 vocoder. Silence compression techniques are used to reduce the transmitted bit rate during silent intervals of speech. Systems allowing discontinuous transmission are based on a Voice Activity Detection (VAD) algorithm and a Comfort Noise Generator (CNG) algorithm that allows the insertion of an artificial noise during silence periods. This feature is necessary to avoid the noise modulation introduced when the transmission is switched off. If the background acoustic noise that was present during active periods abruptly disappears, this very unpleasant noise modulation may even reduce the intelligibility of the speech.

1.2.2 Features and Performance

The G.723.1A Dual Rate Speech Coder library is multichannel and re-entrant.

For details on Memory and MIPS for a particular DSP, refer to the **Libraries** chapter of the appropriate Targeting manual.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 2
Directory Structure

2.1 Required Core Directories

Figure 2-1 details required platform directories:

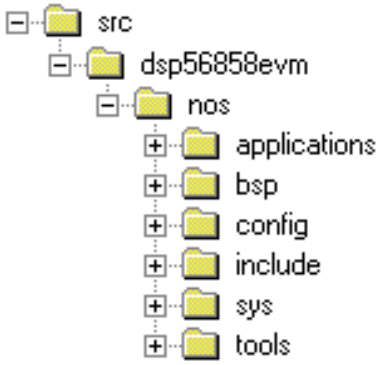

Figure 2-1. Core Directories

As shown in Figure 2-1, DSP56858EVM has no operating system (nos) support. This platform contains these core directories:

- ***applications*** contains applications software that can be exercised on this platform
- ***bsp*** contains board support package specific for this platform
- ***config*** contains default hardware/software configurations for this platform
- ***include*** contains SDK header files which define the Application Programming Interface
- ***sys*** contains required system components
- ***tools*** contains utilities used by system components

There are also optional directories that include domain-specific libraries.

2.2 Optional (Domain-Specific) Directories

Figure 2-2 demonstrates how the G.723.1A Dual Rate Speech Codec is encapsulated in the domain-specific directory *telephony*.

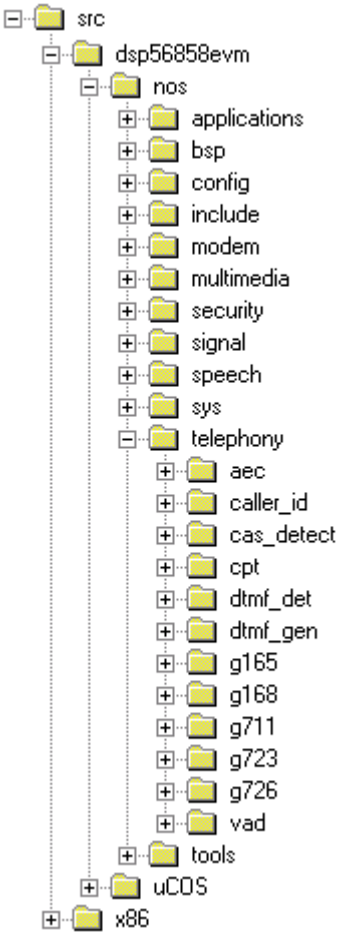

Figure 2-2. DSP56858 Directories

The *g723.1A* Dual Rate Speech Codec directory includes G723.1A Dual Rate Speech Codec-specific algorithms. **Figure 2-3** shows the *g723.1A* directory structure.

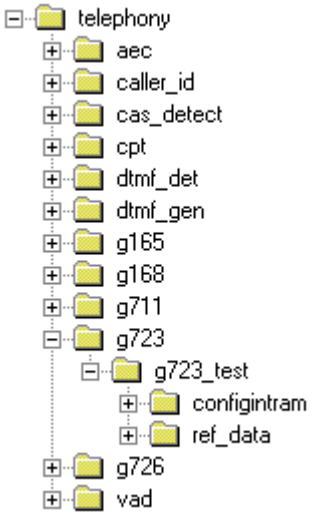

Figure 2-3. *G723.1A* Dual Rate Speech Codec Directory Structure

The *g723* directory the G.723.1A library as well as the following sub-directories:

- ***g723_test*** includes *c_source* files and configuration necessary for testing G723.1A library modules
  - ***configintram*** contains configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to G726 Encoder/Decoder testing
  - ***ref_data*** contains files containing reference data used to compare the output of the test

2.3 Demo Directory Structure

Figure 2-4 demonstrates how the G.723.1A Dual Rate Speech Codec demo (*g723*) is encapsulated in the *telephony* directory under *applications.*

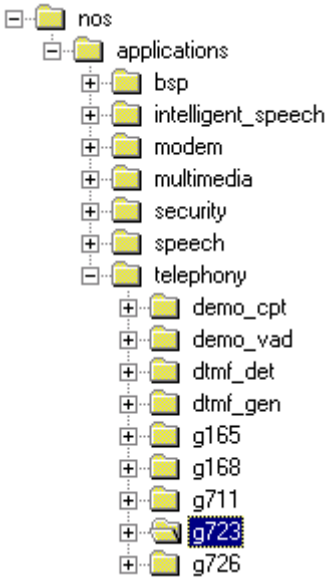

Figure 2-4. SDK Application Directory Structure

The *g723* directory includes G.723.1A Dual Rate Speech Codec demo-specific algorithms. **Figure 2-5** shows the *g723* directory structure.

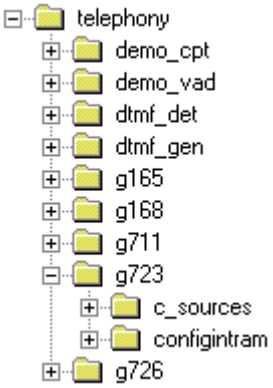

Figure 2-5. *g723* Directory Structure

- ***g723*** includes c sources and configuration necessary for testing G.723.1A Dual Rate Speech Codec demo modules
  - — ***c_sources*** contains an example demo code for the G.723.1A Dual Rate Speech Codec
  - — ***configintram*** contains configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to the G.723.1A Dual Rate Speech Codec demo

Chapter 3
G.723.1A Codec Library Interfaces

3.1 G.723.1A Codec Services

The principal application of the G.723.1A speech codec is compression of speech in either of two very low bit rates for telephony applications.

3.2 Interface

The C interface for the G.723.1A Dual Rate Speech Codec library service is defined in the C header files *g723.h*, shown in **Code Example 3-1**.

SDK defines the G.723.1A Dual Rate Speech Codec library:

G.723.1A Dual Rate Speech Codec Library (*g723.lib*)

A description of the interfaces and services provided follow.

Code Example 3-1. C Header File *g723.h*

```
/*-------------------------------------------------------------------------*/
/*                                                                         */
/* The Channel1 structure needs to be long word aligned, the following     */
/*  declaration can be used to guarantee this.                            */
/*                                                                         */
/* long Channel1[GLOBAL_MEM_Size/2];                                       */
/*                                                                         */
/*                                                                         */
/*                                                                         */
/* The input and output speech arrays should be declared as follows:       */
/*                                                                         */
/* int EncodeSpeech[Frame];                                                */
/* int DecodeSpeech[Frame];                                                */
/*                                                                         */
/*                                                                         */
```

```
/* The input and output channel arrays should be declared as follows:          */
/*                                                                             */
/* int EncodeChannel[EncodedFrame];                                            */
/* int DecodeChannel[EncodedFrame];                                            */
/*                                                                             */
/*                                                                             */
/* UseHp - Indicates whether to use high-pass filtering                        */
/* UseVx - indicates whether to use Comp_Vad (Voice Activity Detection)        */
/* WrkRate - sets the coder bit rate (Rate63 or Rate53)                        */
/*                                                                             */
/*-----------------------------------------------------------------------------*/
#ifndef __G723_H
#define __G723_H

#include "port.h"


void Init_Coder(Word32 *Channel1);

void Init_Vad(Word32 *Channel1);

void Init_Cod_Cng(Word32 *Channel1);

Word16  Coder(Word32 *Channel1, Word16 *EncodeSpeech, Word16 *EncodeChannel,
              Word16 UseHp,Word16 UseVx, Word16 WrkRate);

void Init_Decod(Word32 *Channel1);

void Init_Dec_Cng(Word32 *Channel1);

Word16  Decod(Word32 *Channel1, Word16 *DecodeSpeech, Word16 *DecodeChannel,
              Word16 Crc, Word16 UsePf);


#define       False               0
#define       True                1


/* Definition of the working mode */
#define       Both                0
#define       Cod                       1
#define       Dec                       2

/* Coder rate */
#define       Rate63              0
#define       Rate53              1

/* Coder global constants */
#define       Frame               240
#define       LpcFrame      180
#define       SubFrames     4
#define       SubFrLen      (Frame/SubFrames)
#define       EncodedFrame 12
```

```
/* LPC constants */
#define      LpcOrder     10
#define      RidgeFact    10
#define      CosineTableSize512
#define      PreCoef             0xc000

#define      LspPrd0             12288
#define      LspPrd1             23552

#define      LspQntBands  3
#define      LspCbSize    256
#define      LspCbBits    8

/* LTP constants */
#define      PitchMin     18
#define      PitchMax     (PitchMin+127)
#define      PwConst             0x2800
#define      PwRange             3
#define      ClPitchOrd   5
#define      Pstep               1
#define      NbFilt085    85
#define      NbFilt170    170

/* MP-MLQ constants */
#define      Sgrid               2
#define      MaxPulseNum  6
#define      MlqSteps     2

/* acelp constants */
#define      SubFrLen2    (SubFrLen+4)
#define      DIM_RR              416
#define      NB_POS              8
#define      STEP                8
#define      MSIZE               64
#define      threshold    16384
#define      max_time     120

/* Gain constant */
#define      NumOfGainLev 24

/* FER constant */
#define      ErrMaxNum    3

/* CNG constants */
#define      NbAvAcf             3
#define      NbAvGain     3
#define      ThreshGain   3
#define      FracThresh   7000
#define      NbPulsBlk    11

#define      InvNbPulsBlk 2979
#define      NbFilt              50
#define      LpcOrderP1   (LpcOrder+1)
#define      SizAcf              ((NbAvAcf+1)*LpcOrderP1)
#define      SubFrLenD    (2*SubFrLen)
#define      Gexc_Max     5000
```

```
/* Taming constants */
#define      NbFilt085_min51
#define      NbFilt170_min93
#define      SizErr              5
#define      Err0                0x00000004
#define      ThreshErr    0x40000000
#define      DEC                 (30-7)


/* Structure offsets - Later to be added to RAM include file */
#define Word16Size 1
#define Word32Size 2

/* Base address used to index first element in each structure */
#define      BaseAddr     0


/* Defines for CODSTATDEF structure elements */
/* High pass variables */
#define      CODSTATDEF_HpfPdlBaseAddr
#define      CODSTATDEF_HpfZdlCODSTATDEF_HpfPdl+Word32Size

/* Sine wave detector */
#define      CODSTATDEF_SinDetCODSTATDEF_HpfZdl+Word16Size

/* Taming procedure errors */
#define      CODSTATDEF_ErrCODSTATDEF_SinDet+Word16Size

/* Lsp previous vector */
#define      CODSTATDEF_PrevLspCODSTATDEF_Err+(Word32Size*SizErr)

/* All pitch operation buffers */
#define      CODSTATDEF_PrevWgtCODSTATDEF_PrevLsp+(Word16Size*LpcOrder)
#define      CODSTATDEF_PrevErrCODSTATDEF_PrevWgt+(Word16Size*PitchMax)
#define      CODSTATDEF_PrevExcCODSTATDEF_PrevErr+(Word16Size*PitchMax)

/* Required memory for the delay */
#define      CODSTATDEF_PrevDatCODSTATDEF_PrevExc+(Word16Size*PitchMax)

/* Used delay lines */
#define      CODSTATDEF_WghtFirDlCODSTATDEF_PrevDat+(Word16Size*(LpcFrame-SubFrLen))
#define      CODSTATDEF_WghtIirDlCODSTATDEF_WghtFirDl+(Word16Size*LpcOrder)
#define      CODSTATDEF_RingFirDlCODSTATDEF_WghtIirDl+(Word16Size*LpcOrder)
#define      CODSTATDEF_RingIirDlCODSTATDEF_RingFirDl+(Word16Size*LpcOrder)

#define      CODSTATDEF_SizeCODSTATDEF_RingIirDl+(Word16Size*LpcOrder)


/* Defines for DECSTATDEF structure elements */
/* High pass variables */
#define      DECSTATDEF_EcountBaseAddr
#define      DECSTATDEF_InterGainDECSTATDEF_Ecount+Word16Size
#define      DECSTATDEF_InterIndxDECSTATDEF_InterGain+Word16Size
#define      DECSTATDEF_RseedDECSTATDEF_InterIndx+Word16Size
#define      DECSTATDEF_ParkDECSTATDEF_Rseed+Word16Size
#define      DECSTATDEF_GainDECSTATDEF_Park+Word16Size
```

```
/* Lsp previous vector */
#define      DECSTATDEF_PrevLspDECSTATDEF_Gain+Word16Size

/* All pitch operation buffers */
#define      DECSTATDEF_PrevExcDECSTATDEF_PrevLsp+(Word16Size*LpcOrder)

/* Used delay lines */
#define      DECSTATDEF_SyntIirDlDECSTATDEF_PrevExc+(Word16Size*PitchMax)
#define      DECSTATDEF_PostFirDlDECSTATDEF_SyntIirDl+(Word16Size*LpcOrder)
#define      DECSTATDEF_PostIirDlDECSTATDEF_PostFirDl+(Word16Size*LpcOrder)
#define      DECSTATDEF_SizeDECSTATDEF_PostIirDl+(Word16Size*LpcOrder)


/* Defines for SFSDEF structure elements */
/* subframe coded parameters */
#define      SFSDEF_AcLg  BaseAddr
#define      SFSDEF_AcGn  SFSDEF_AcLg+Word16Size
#define      SFSDEF_Mamp  SFSDEF_AcGn+Word16Size
#define      SFSDEF_Grid  SFSDEF_Mamp+Word16Size
#define      SFSDEF_Tran  SFSDEF_Grid+Word16Size
#define      SFSDEF_Pamp  SFSDEF_Tran+Word16Size
#define      SFSDEF_Ppos  SFSDEF_Pamp+Word16Size
#define      SFSDEF_Size  SFSDEF_Ppos+Word32Size


/* Defines for LINEDEF structure elements */
/* frame coded parameters */
#define      LINEDEF_LspIdBaseAddr
#define      LINEDEF_Olp  LINEDEF_LspId+Word32Size
#define      LINEDEF_Sfs  LINEDEF_Olp+(Word16Size*(SubFrames/2))
#define      LINEDEF_Crc  LINEDEF_Sfs+(SFSDEF_Size*SubFrames)
#define      LINEDEF_Size LINEDEF_Crc+Word16Size


/* Defines for PWDEF structure elements */
/* harmonic noise shaping filter parameters */
#define      PWDEF_Indx   BaseAddr
#define      PWDEF_Gain   PWDEF_Indx+Word16Size
#define      PWDEF_Size   PWDEF_Gain+Word16Size


/* Defines for PFDEF structure elements */
/* pitch postfilter parameters */
#define      PFDEF_Indx   BaseAddr
#define      PFDEF_Gain   PFDEF_Indx+Word16Size
#define      PFDEF_ScGn   PFDEF_Gain+Word16Size
#define      PFDEF_Size   PFDEF_ScGn+Word16Size


/* Defines for BESTDEF structure elements */
/* best excitation vector parameters for the high rate */
#define      BESTDEF_MaxErrBaseAddr
#define      BESTDEF_GridIdBESTDEF_MaxErr+Word32Size
#define      BESTDEF_MampIdBESTDEF_GridId+Word16Size
#define      BESTDEF_UseTrnBESTDEF_MampId+Word16Size
```

```
#define      BESTDEF_Ploc BESTDEF_UseTrn+Word16Size
#define      BESTDEF_Pamp BESTDEF_Ploc+(Word16Size*MaxPulseNum)
#define      BESTDEF_Size BESTDEF_Pamp+(Word16Size*MaxPulseNum)


/* Defines for VADSTATDEF structure elements */
/* best excitation vector parameters for the high rate */
#define      VADSTATDEF_HcntBaseAddr
#define      VADSTATDEF_VcntVADSTATDEF_Hcnt+Word16Size
#define      VADSTATDEF_PenrVADSTATDEF_Vcnt+Word16Size
#define      VADSTATDEF_NlevVADSTATDEF_Penr+Word32Size
#define      VADSTATDEF_AenVADSTATDEF_Nlev+Word32Size
#define      VADSTATDEF_PolpVADSTATDEF_Aen+Word16Size
#define      VADSTATDEF_NLpcVADSTATDEF_Polp+(Word16Size*4)
#define      VADSTATDEF_SizeVADSTATDEF_NLpc+(Word16Size*LpcOrder)


/* CNG features */

/* Defines for CODCNGDEF structure elements */
/* Coder part */
#define      CODCNGDEF_CurGainBaseAddr
#define      CODCNGDEF_PastFtypCODCNGDEF_CurGain+Word16Size
#define      CODCNGDEF_AcfCODCNGDEF_PastFtyp+Word16Size
#define      CODCNGDEF_ShAcfCODCNGDEF_Acf+(Word16Size*SizAcf)
#define      CODCNGDEF_LspSidCODCNGDEF_ShAcf+(Word16Size*(NbAvAcf+1))
#define      CODCNGDEF_SidLpcCODCNGDEF_LspSid+(Word16Size*LpcOrder)
#define      CODCNGDEF_RC CODCNGDEF_SidLpc+(Word16Size*LpcOrder)
#define      CODCNGDEF_ShRCCODCNGDEF_RC+(Word16Size*LpcOrderP1)
#define      CODCNGDEF_EnerCODCNGDEF_ShRC+Word16Size
#define      CODCNGDEF_NbEnerCODCNGDEF_Ener+(Word16Size*NbAvGain)
#define      CODCNGDEF_IRefCODCNGDEF_NbEner+Word16Size
#define      CODCNGDEF_SidGainCODCNGDEF_IRef+Word16Size
#define      CODCNGDEF_RandSeedCODCNGDEF_SidGain+Word16Size
#define      CODCNGDEF_SizeCODCNGDEF_RandSeed+Word16Size


/* Defines for DECCNGDEF structure elements */
/* Decoder part */
#define      DECCNGDEF_CurGainBaseAddr
#define      DECCNGDEF_PastFtypDECCNGDEF_CurGain+Word16Size
#define      DECCNGDEF_LspSidDECCNGDEF_PastFtyp+Word16Size
#define      DECCNGDEF_SidGainDECCNGDEF_LspSid+(Word16Size*LpcOrder)
#define      DECCNGDEF_RandSeedDECCNGDEF_SidGain+Word16Size
#define      DECCNGDEF_SizeDECCNGDEF_RandSeed+Word16Size

/* Defines for offsets into global memory */
#define      CodStat             BaseAddr
#define      CodCng              CodStat+CODSTATDEF_Size
#define      DecCng              CodCng+CODCNGDEF_Size
#define      VadStat             DecCng+DECCNGDEF_Size
#define      DecStat             VadStat+VADSTATDEF_Size
#define      UseHp               DecStat+DECSTATDEF_Size
#define      UsePf               UseHp+Word16Size
#define      UseVx               UsePf+Word16Size
```

```
#define     WrkMode             UseVx+Word16Size
#define     WrkRate             WrkMode+Word16Size
#define     extra               WrkRate+Word16Size
#define     GLOBAL_MEM_Size(extra+Word16Size)

#endif
```

3.3 Specifications

The following pages describe the G.723.1A Dual Rate Speech Codec library functions.

Function arguments for each routine are described as *in*, *out*, or *inout*. An *in* argument means that the parameter value is an input only to the function. An *out* argument means that the parameter value is an output only from the function. An *inout* argument means that a parameter value is an input to the function, but the same parameter is also an output from the function.

Typically, *inout* parameters are input pointer variables in which the caller passes the address of a preallocated data structure to a function. The function stores its results within that data structure. The actual value of the *inout* pointer parameter is not changed.

3.3.1 *Init_Coder*

Call(s):

```
void Init_Coder(Word32 *Channel);
```

Required Header: "g723.h"

Arguments:

Table 3-1. *Init_Coder* Arguments

Channel	inout	A pointer to a data structure containing channel information for the G.723.1A algorithm

Description: The *Init_Coder* function initializes the G.723.1A Encoder algorithm. During initialization, all resources will be set to their initial values in preparation for G.723.1A Encoder operation. The *Init_Coder* function should be called only once before the first call to the *Coder* function.

The parameter *Channel* points to a data structure of type *Word32*; its fields initialize G.723.1A operation in the following manner:

Use_Hp - Sets high-pass filtering

`TRUE` - Sets high-pass filtering
`FALSE` - No high-pass filtering is set

Use_Pf - Sets post filtering

`TRUE` - Sets post filtering
`FALSE` - No post filtering is set

Use_Vx - Sets VAD/CNG

`TRUE` - Sets VAD/CNG
`FALSE` - No VAD/CNG is set

WrkMode - Selects encoding, decoding or both

`Both` - Sets encoding and decoding
`Cod` - Sets encoding only
`Dec` - Sets decoding only

WrkRate - Selects the speech codec's rate

`Rate63` - 6.3kBps
`Rate53` - 5.3kBps

extra - extra time

Returns: None

Special Considerations: None

Code Example 3-2. Use of the *Init_Coder* Interface

```
#include “g723.h”

/*============================================================================
                                    LOCAL VARIABLES
============================================================================*/
#define NUM_FRAMES 860

/* Declare the ChannelMem structure*/
Word32 Channel1[GLOBAL_MEM_Size/2];

/*============================================================================
                                    APPLICATION
============================================================================*/
int test_g7231a(void)
{

            /* Set two's complement rounding and enable saturation */
            dspfuncInitialize();


            /* Initialize the G.723 vocoder encode and decode functions as well as */
            /* voice activity detection and confort noise generation */
            Init_Coder(Channel1);
}
```

3.3.2 *Init_Vad*

Call(s):

```
void Init_Vad(Word32 *Channel);
```

Required Header: "g723.h"

Arguments:

Table 3-2. *Init_Vad* Arguments

Channel	inout	A pointer to a data structure containing channel information for the G.723.1A algorithm

Description: The *Init_Vad* function initializes the G.723.1A VAD static variables. During initialization, all resources will be set to their initial values in preparation for G.723.1A operation. The *Init_Vad* function should be called only once before the first call to the *Coder* function.

The parameter *Channel* points to a data structure of type *Word32*; its fields initialize G.723.1A operation in the following manner:

Use_Hp - Sets high-pass filtering

`TRUE` - Sets high-pass filtering
`FALSE` - No high-pass filtering is set

Use_Pf - Sets post filtering

`TRUE` - Sets post filtering
`FALSE` - No post filtering is set

Use_Vx - Sets VAD/CNG

`TRUE` - Sets VAD/CNG
`FALSE` - No VAD/CNG is set

WrkMode - Selects encoding, decoding or both

`Both` - Sets encoding and decoding
`Cod` - Sets encoding only
`Dec` - Sets decoding only

WrkRate - Selects the speech codec's rate

`Rate63` - 6.3kBps
`Rate53` - 5.3kBps

extra - extra time

Returns: None

Special Considerations: None

Code Example 3-3. Use of the *Init_Vad* Interface

```
#include “g723.h”

/*============================================================================
                                    LOCAL VARIABLES
============================================================================*/
#define NUM_FRAMES 860

/* Declare the ChannelMem structure*/
Word32 Channel1[GLOBAL_MEM_Size/2];

/*============================================================================
                                    APPLICATION
============================================================================*/
int test_g7231a(void)
{

            /* Set two's complement rounding and enable saturation */
            dspfuncInitialize();


            /* Initialize the G.723 vocoder encode and decode functions as well as */
            /* voice activity detection and confort noise generation */
            Init_Coder(Channel1);
            Init_Vad(Channel1);
}
```

3.3.3 *Init_Cod_Cng*

Call(s):

```
void Init_Cod_Cng(Word32 *Channel);
```

Required Header: "g723.h"

Arguments:

Table 3-3. *Init_Cod_Cng* Arguments

Channel	inout	A pointer to a data structure containing channel information for the G.723.1A algorithm

Description: The *Init_Cod_Cng* function initializes the G.723.1A *Cod_Cng* static variables. During initialization, all resources will be set to their initial values in preparation for G.723.1A operation. The *Init_Cod_Cng* function should be called only once before the first call to the *Coder* function.

The parameter *Channel* points to a data structure of type *Word32*; its fields initialize G.723.1A operation in the following manner:

Use_Hp - Sets high-pass filtering

`TRUE` - Sets high-pass filtering
`FALSE` - No high-pass filtering is set

Use_Pf - Sets post filtering

`TRUE` - Sets post filtering
`FALSE` - No post filtering is set

Use_Vx - Sets VAD/CNG

`TRUE` - Sets VAD/CNG
`FALSE` - No VAD/CNG is set

WrkMode - Selects encoding, decoding or both

`Both` - Sets encoding and decoding
`Cod` - Sets encoding only
`Dec` - Sets decoding only

WrkRate - To select the speech codec's rate

`Rate63` - 6.3kBps
`Rate53` - 5.3kBps

extra - extra time

Returns: None

Special Considerations: None

Code Example 3-4. Use of the *Init_Cod_Cng* Interface

```
#include “g723.h”

/*============================================================================
                                    LOCAL VARIABLES
============================================================================*/
#define NUM_FRAMES 860

/* Declare the ChannelMem structure*/
Word32 Channel1[GLOBAL_MEM_Size/2];

/*============================================================================
                                    APPLICATION
============================================================================*/
int test_g7231a(void)
{

            /* Set two's complement rounding and enable saturation */
            dspfuncInitialize();


            /* Initialize the G.723 vocoder encode and decode functions as well as */
            /* voice activity detection and confort noise generation */
            Init_Coder(Channel1);
            Init_Vad(Channel1);
            Init_Cod_Cng(Channel1);
}
```

3.3.4 *Coder*

Call(s):

```
Word16 Coder (      Word32 *Channel,
                    Word16  *EncodeSpeech,
                    Word16  *EncodeChannel,
                    Word16  UseHp,
                    Word16  UseVx,
                    Word16  WrkRate);
```

Required Header: "g723.h"

Arguments :

Table 3-4. *Coder* Arguments

Channel	in	A pointer to a data structure containing channel information for the G.723.1A algorithm
EncodeSpeech	in	Pointer to frame of speech data
EncodeChannel	inout	Pointer to an encoded frame of speech data
UseHp	in	High-pass filter flag
UseVx	in	VAD/CNG flag
WrkRate	in	Specifies 5.3 or 6.3kBps

Description: The *Coder* function processes input samples and encodes them to either 5.3 or 6.3 kbps output rate.

Returns: *Coder* will return "PASS" after successful encoding.

Special Considerations: None

Code Example 3-5. Use of *Coder* Interface

```
#include “g723.h”

/*============================================================================
                          LOCAL FUNCTIONS PROTOTYPES
============================================================================*/
Word16 ReadSpeech(char *buffer, int fin);
Word16 ReadChannel(char *buffer, int fin);

/*============================================================================
                          LOCAL VARIABLES
============================================================================*/
#define NUM_FRAMES 860

/* Declare the ChannelMem structure*/
Word32 Channel1[GLOBAL_MEM_Size/2];

/* Declare the G.723 speech buffers */
Word16       EncodeChannel[EncodedFrame];

/* Declare encoded channel buffers */
Word16       EncodeSpeech[Frame];
Word16       OutputSpeech[Frame];

/* Declare an ascii I/O buffer */
char         ioBuffer[2*EncodedFrame];


/*============================================================================
                          APPLICATION MAIN
============================================================================*/
int main(void)
{
             int f_tstfile; /* File of interleaved input speech, channel data */
                            /* and output speech */

             char TestFile[]  = {"\\\\PC\\Embedded
SDK\\src\\dsp56858evm\\nos\\telephony\\g723\\g723_test\\ref_data\\tstboth.bin"};

             Word16  i, CoderRate, BitErrors;
             Word16  frames;

             CoderRate = Rate63;
             BitErrors = 0;


             f_tstfile = open(TestFile, O_RDONLY);
             if (f_tstfile == NULL) assert("Error opening the test file.");


             /* Set two's complement rounding and enable saturation */
             dspfuncInitialize();


             /* Initialize the G.723 vocoder encode and decode functions as well as */
```

```
                /* voice activity detection and confort noise generation */
                Init_Coder(Channel1);
                Init_Vad(Channel1);
                Init_Cod_Cng(Channel1);

                for (frames=0; frames<NUM_FRAMES; frames++)
                {
                      if (!ReadSpeech((char *)EncodeSpeech, f_tstfile))
                            break;

                      for (i=0; i<EncodedFrame; i++) {
                            EncodeChannel[i] = 0;
                      }

                      Coder(Channel1,EncodeSpeech,EncodeChannel,True,True,CoderRate);

                      if (!ReadChannel((char *)DecodeChannel, f_tstfile))
                            break;

                      for (i=0; i<EncodedFrame; i++) {
                            if (DecodeChannel[i] != EncodeChannel[i]) {
                                BitErrors++;
                            }
                      }
                }
                close(f_tstfile);
}
```

3.3.5 *Init_Decod*

Call(s):

```
void Init_Decod(Word32 *Channel);
```

Required Header: "g723.h"

Arguments:

Table 3-5. *Init_Decod* Arguments

Channel	inout	A pointer to a data structure containing channel information for the G.723.1A algorithm

Description: The *Init_Decod* function initializes the G.723.1A Decoder algorithm. During initialization, all resources will be set to their initial values in preparation for G.723.1A Decoder operation. The *Init_Decod* function should be called only once before the first call to the *Decod* function

The parameter *Channel* points to a data structure of type *Word32*; its fields initialize G.723.1A operation in the following manner:

Use_Hp - Sets high-pass filtering

`TRUE` - Sets high-pass filtering
`FALSE` - No high-pass filtering is set

Use_Pf - Sets post filtering

`TRUE` - Sets post filtering
`FALSE` - No post filtering is set

Use_Vx - Sets VAD/CNG

`TRUE` - Sets VAD/CNG
`FALSE` - No VAD/CNG is set

WrkMode - Selects encoding, decoding or both

`Both` - Sets encoding and decoding
`Cod` - Sets encoding only
`Dec` - Sets decoding only

WrkRate - Selects the speech codec's rate

`Rate63` - 6.3kBps
`Rate53` - 5.3kBps

extra - extra time

Returns: None

Special Considerations: None

Code Example 3-6. Use of the *Init_Decod* Interface

```
#include “g723.h”

/*============================================================================
                                    LOCAL VARIABLES
============================================================================*/
#define NUM_FRAMES 860

/* Declare the ChannelMem structure*/
Word32 Channel1[GLOBAL_MEM_Size/2];

/*============================================================================
                                    APPLICATION
============================================================================*/
int test_g7231a(void)
{

            /* Set two's complement rounding and enable saturation */
            dspfuncInitialize();


            /* Initialize the G.723 vocoder encode and decode functions as well as */
            /* voice activity detection and confort noise generation */
            Init_Decod(Channel1);
}
```

3.3.6 *Init_Dec_Cng*

Call(s):

```
Init_Dec_Cng(Channel);
```

Required Header: "g723.h"

Arguments:

Table 3-6. *Init_Dec_Cng* Arguments

Channel	inout	A pointer to a data structure containing channel information for the G.723.1A algorithm

Description: The *Init_Dec_Cng* function initializes the G.723.1A *Dec_Cng* static variables. During initialization, all resources will be set to their initial values in preparation for G.723.1A operation. The *Init_Dec_Cng* function should be called only once before the first call to the *Decod* function.

The parameter *Channel* points to a data structure of type *Word32*; its fields initialize G.723.1A operation in the following manner:

Use_Hp - Sets high-pass filtering

`TRUE` - Sets high-pass filtering
`FALSE` - No high-pass filtering is set

Use_Pf - Sets post filtering

`TRUE` - Sets post filtering
`FALSE` - No post filtering is set

Use_Vx - Sets VAD/CNG

`TRUE` - Sets VAD/CNG
`FALSE` - No VAD/CNG is set

WrkMode - Selects encoding, decoding or both

`Both` - Sets encoding and decoding
`Cod` - Sets encoding only
`Dec` - Sets decoding only

WrkRate - To select the speech codec's rate

`Rate63` - 6.3kBps
`Rate53` - 5.3kBps

extra - extra time

Returns: None

Special Considerations: None

Code Example 3-7. Use of the *Init_Dec_Cng* Interface

```
#include “g723.h”

/*============================================================================
                                    LOCAL VARIABLES
============================================================================*/
#define NUM_FRAMES 860

/* Declare the ChannelMem structure*/
Word32 Channel1[GLOBAL_MEM_Size/2];

/*============================================================================
                                    APPLICATION
============================================================================*/
int test_g7231a(void)
{

            /* Set two's complement rounding and enable saturation */
            dspfuncInitialize();


            /* Initialize the G.723 vocoder encode and decode functions as well as */
            /* voice activity detection and confort noise generation */
            Init_Coder(Channel1);
            Init_Dec_Cng(Channel1);
}
```

3.3.7 *Decod*

Call(s):

```
Word16 Decod (      Word32 *Channel,
                    Word16  *DecodeSpeech,
                    Word16  *DecodeChannel,
                    Word16  Crc,
                    Word16  UsePf);
```

Required Header: “g723.h”

Arguments :

Table 3-7. *Decod* Arguments

Channel	in	A pointer to a data structure containing channel information for the G.723.1A algorithm
DecodeSpeech	inout	Pointer to empty speech buffer
EncodeChannel	in	Pointer to an encoded frame of speech data
Crc	in	Frame erasure indicator
UsePf	in	Post filter flag

Description: The *Decod* function processes the 5.3 or 6.3kBps encoded input and decodes it to speech samples.

Returns: *Decod* will return “PASS” after successful encoding.

Special Considerations: None

Code Example 3-8. Use of *Decod* Interface

```
#include “g723.h”

/*============================================================================
                    LOCAL FUNCTIONS PROTOTYPES
============================================================================*/
Word16 ReadSpeech(char *buffer, int fin);
Word16 ReadChannel(char *buffer, int fin);

/*============================================================================
                    LOCAL VARIABLES
============================================================================*/
#define NUM_FRAMES 860

/* Declare the ChannelMem structure*/
Word32 Channel1[GLOBAL_MEM_Size/2];

/* Declare the G.723 speech buffers */
Word16       EncodeChannel[EncodedFrame];
Word16       DecodeChannel[EncodedFrame];
```

```
/* Declare encoded channel buffers */
Word16       EncodeSpeech[Frame];
Word16       DecodeSpeech[Frame];
Word16       OutputSpeech[Frame];

/* Declare an ascii I/O buffer */
char         ioBuffer[2*EncodedFrame];



/*============================================================================
                    APPLICATION MAIN
============================================================================*/
int main(void)
{
             int f_tstfile; /* File of interleaved input speech, channel data */
                            /* and output speech */

             char TestFile[]  = {"\\\\PC\\Embedded
SDK\\src\\dsp56858evm\\nos\\telephony\\g723\\g723_test\\ref_data\\tstboth.bin"};

             Word16  i, CoderRate, BitErrors;
             Word16  frames;

    CoderRate = Rate63;
    BitErrors = 0;

             f_tstfile = open(TestFile, O_RDONLY);
             if (f_tstfile == NULL) assert("Error opening the test file.");

             /* Set two's complement rounding and enable saturation */
             dspfuncInitialize();


             /* Initialize the G.723 vocoder encode and decode functions as well as */
             /* voice activity detection and confort noise generation */
             Init_Coder(Channel1);
             Init_Vad(Channel1);
             Init_Cod_Cng(Channel1);

             Init_Decod(Channel1);
             Init_Dec_Cng(Channel1);

             for (frames=0; frames<NUM_FRAMES; frames++)
             {

                    if (!ReadSpeech((char *)EncodeSpeech, f_tstfile))
                           break;

                    for (i=0; i<EncodedFrame; i++) {
                           EncodeChannel[i] = 0;
                           DecodeChannel[i] = 0;
                    }

                    Coder(Channel1,EncodeSpeech,EncodeChannel,True,True,CoderRate);
```

```
                    if (!ReadChannel((char *)DecodeChannel, f_tstfile))
                        break;

                    for (i=0; i<EncodedFrame; i++) {
                        if (DecodeChannel[i] != EncodeChannel[i]) {
                            BitErrors++;
                        }
                    }

                    /*     Decod(Channel1,Speechdata,Channeldata,Crc,UsePf); */
                    Decod(Channel1,DecodeSpeech,DecodeChannel,0,True);

                    if (!ReadSpeech((char *)OutputSpeech, f_tstfile))
                        break;

                    for (i=0; i<Frame; i++) {
                        if (OutputSpeech[i] != DecodeSpeech[i]) {
                            BitErrors++;
                        }
                    }

                }
                close(f_tstfile);
}
```

Chapter 4
Building the G.723.1A Codec Library

4.1 Building the G.723.1A Codec Library

The G.723.1A Codec library combines all of the components described in the previous sections into one library: *g723.lib*. This library's code is not provided with the SDK; therefore, it cannot be built from the SDK. The G.723.1A Codec library, *g723.lib*, is provided in the ....*\nos\telephony\g723* directory of the SDK directory structure.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 5
Linking Applications with the G.723.1A Codec Library

5.1 G.723.1A Codec Library

The G.723.1A library consists of Applications Program Interfaces, (APIs), which provide interface between the user application and the G.723.1A Codec modules. To invoke G.723.1A Codec, APIs must be called in this order:

For Encoder:

- Init_Coder (.....) /* Called once before Coder */
- Init_Vad (.....) /* Called once before Coder */
- Init_Cod_Cng (.....) /* Called once before Coder */
- Coder(.....); /* Can be called by the number of samples */

For Decoder:

- Init_Decod (.....) /* Called once before Decod */
- Init_Dec_Cng (.....) /* Called once before Decod */
- Decod(.....); /* Can be called by the number of samples */

A sample linker command file, *linker.cmd*, used in the test application for encoder and decoder, is shown in **Code Example 5-1**.

Code Example 5-1. *linker.cmd* File

```
#*****************************************************************************
#
# Linker.cmd file for DSP56852 Internal RAM
#    using both internal and external program
#    and data memory.
#
#*****************************************************************************

MEMORY {

       .pInterruptVector   (RWX) : ORIGIN = 0x000000, LENGTH = 0x00008C
       .pIntRAM            (RWX) : ORIGIN = 0x00008C, LENGTH = 0x009f74
#      .pExtRAM            (RWX) : ORIGIN = 0x001800, LENGTH = 0x1EE800
# LS010703 - Changed to use Viper extended internal RAM temporarily
       .pExtRAM            (RWX) : ORIGIN = 0x00a000, LENGTH = 0x00B800

       .pIntROM            (RX)  : ORIGIN = 0x1F0000, LENGTH = 0x000400

       .xIntRAM            (RW)  : ORIGIN = 0x000000, LENGTH = 0x000800
       .xIntRAM_DynamicMem (RW)  : ORIGIN = 0x000800, LENGTH = 0x000800

       .xStack             (RW)  : ORIGIN = 0x001000, LENGTH = 0x001000
       .xExtRAM_DynamicMem (RW)  : ORIGIN = 0x007800, LENGTH = 0x000800
#      .xExtRAM            (RW)  : ORIGIN = 0x002800, LENGTH = 0x1FD400
# LS010703 - Changed to use Viper extended internal RAM temporarily
       .xExtRAM            (RW)  : ORIGIN = 0x002000, LENGTH = 0x005000

       .xPeripherals       (RW)  : ORIGIN = 0x1FFC00, LENGTH = 0x000400
       .xExtRAM2           (RW)  : ORIGIN = 0x200000, LENGTH = 0xDFFF00

       .xCoreRegisters     (RW)  : ORIGIN = 0xFFFF00, LENGTH = 0x000100

}

#*****************************************************************************

FORCE_ACTIVE {FconfigInterruptVector}

#*****************************************************************************

SECTIONS {

   #*************************************************************************

    .ApplicationInterruptVector :
      {
            vector.c (.text)

      } > .pInterruptVector

   #*************************************************************************

      .ApplicationCode :
      {
```

```
    # Place all code into Program RAM

          * (.text)
          * (rtlib.text)
          * (fp_engine.text)
          * (user.text)

          PmemData.c (.const.data)
          PmemData.c (.data)

          # SDK data to be placed into Program RAM

          F_Pdata_start_addr_in_ROM = 0;
          F_Pdata_start_addr_in_RAM = .;
          pramdata.c (.const.data)
          pramdata.c (.data)
          F_Pdata_ROMtoRAM_length = 0;
    F_Pbss_start_addr = .;
          _P_BSS_ADDR = .;
          pramdata.c (.bss)
          PmemData.c (.bss)

          F_Pbss_length = . - _P_BSS_ADDR;

    } > .pIntRAM

#*****************************************************************************

    .ApplicationData :
    {
          # Define variables for C initialization code

          F_Xdata_start_addr_in_ROM = .;
          F_StackAddr               = ADDR(.xStack);
          F_StackEndAddr            = ADDR(.xStack) + SIZEOF(.xStack) - 1;
          F_Xdata_start_addr_in_RAM = .;

          # Define variables for SDK mem library

          # Data (X) Memory Layout

                _EX_BIT     = 0;

             # Internal Memory Partitions (for mem.h partitions)

                _NUM_IM_PARTITIONS = 1;  # IM_ADDR_1 (no IM_ADDR_2 )

             # External Memory Partition (for mem.h partitions)

                _NUM_EM_PARTITIONS = 1;   # EM_ADDR_1

             FmemEXbit = .;
                  WRITEH(_EX_BIT);
             FmemNumIMpartitions = .;
                  WRITEH(_NUM_IM_PARTITIONS);
```

```
                FmemNumEMpartitions = .;
                     WRITEH(_NUM_EM_PARTITIONS);
                FmemIMpartitionList = .;
                     WRITEH(ADDR(.xIntRAM_DynamicMem)*1);
                     WRITEH(SIZEOF(.xIntRAM_DynamicMem)*1);
                FmemEMpartitionList = .;
                     WRITEH(ADDR(.xExtRAM_DynamicMem)*1);
                     WRITEH(SIZEOF(.xExtRAM_DynamicMem)*1);

            # Add rest of the data into External RAM

               * (.const.data)
               * (.data)
               * (fp_state.data)
               * (rtlib.data)

            F_Xdata_ROMtoRAM_length = 0;

            F_Xbss_start_addr = .;
            _X_BSS_ADDR = .;

               * (rtlib.bss.lo)
               * (.bss)

            F_Xbss_length = . - _X_BSS_ADDR;  # Copy DATA

        } > .xExtRAM

    #*************************************************************************

        FArchIO             = 0x0000;
        FArchCore           = ADDR(.xCoreRegisters);
        FArchInterrupts     = ADDR(.pInterruptVector);

}
```

Chapter 6
G.723.1A Applications

6.1 Test and Demo Applications

To verify the G.723.1A algorithm, test and demo applications have been developed. Refer to the **Targeting Motorola DSP568xx Platform** Manual for the DSP you are using to see if the test and demo applications are available for your target.

Chapter 7
License

7.1 Limited Use License Agreement

LIMITED USE LICENSE AGREEMENT

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THIS SOFTWARE. BY USING OR COPYING THE SOFTWARE, YOU AGREE TO THE TERMS OF THIS AGREEMENT.

The software in either source code form ("Source") or object code form ("Object") (cumulatively hereinafter "Software") is provided under a license agreement ("Agreement") as described herein. Any use of the Software including copying, modifying, or installing the Software so that it is usable by or accessible by a central processing unit constitutes acceptance of the terms of the Agreement by the person or persons making such use or, if employed, the employer thereof ("Licensee") and if employed, the person(s) making such use hereby warrants that they have the authority of their employer to enter this license agreement,. If Licensee does not agree with and accept the terms of this Agreement, Licensee must return or destroy any media containing the Software or materials related thereto, and destroy all copies of the Software.

The Software is licensed to Licensee by Motorola Incorporated ("Motorola") for use under the terms of this Agreement. Motorola retains ownership of the Software. Motorola grants only the rights specifically granted in this Agreement and grants no other rights. Title to the Software, all copies thereof and all rights therein, including all rights in any intellectual property including patents, copyrights, and trade secrets applicable thereto, shall remain vested in Motorola.

For the Source, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to use, copy, and make derivatives of the Source solely in a development system environment in order to produce object code solely for operating on a Motorola semiconductor device having a central processing unit ("Derivative Object").

For the Object and Derivative Object, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to copy, use, and distribute the Object and the Derivative Object solely for operating on a Motorola semiconductor device having a central processing unit.

Licensee agrees to: (a) not use, modify, or copy the Software except as expressly provided herein, (b) not distribute, disclose, transfer, sell, assign, rent, lease, or otherwise make available the Software, any derivatives thereof, or this license to a third party except as expressly provided herein, (c) not remove obliterate, or otherwise defeat any copyright, trademark, patent or proprietary notices, related to the Software (d) not in any form export, re-export, resell, ship or divert or cause to be exported, re-exported, resold, shipped, or diverted, directly or indirectly, the Software or a direct product thereof to any country which the United States government or any agency thereof at the time of export or re-export requires an export license or other government approval without first obtaining such license or approval.

THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS AND WITHOUT WARRANTY OF ANY KIND INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MOTOROLA BE LIABLE FOR

ANY LIABILITY OR DAMAGES OF ANY KIND INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT OR INCIDENTAL OR CONSEQUENTIAL OR PUNITIVE DAMAGES OR LOST PROFITS OR LOSS OF USE ARISING FROM USE OF THE SOFTWARE OR THE PRODUCT REGARDLESS OF THE FORM OF ACTION OR THEORY OF LIABILITY (INCLUDING WITHOUT LIMITATION, ACTION IN CONTRACT, NEGLIGENCE, OR PRODUCT LIABILITY) EVEN IF MOTOROLA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THIS DISCLAIMER OF WARRANTY EXTENDS TO LICENSEE OR USERS OF PRODUCTS AND IS IN LIEU OF ALL WARRANTIES WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE.

Motorola does not represent or warrant that the Software is free of infringement of any third party patents, copyrights, trade secrets, or other intellectual property rights or that Motorola has the right to grant the licenses contained herein. Motorola does not represent or warrant that the Software is free of defect, or that it meets any particular requirements or need of the Licensee, or that it conforms to any documentation, or that it meets any standards.

Motorola shall not be responsible to maintain the Software, provide upgrades to the Software, or provide any field service of the Software. Motorola reserves the right to make changes to the Software without further notice to Licensee.

The Software is not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Software could create a situation where personal injury or death may occur. Should Licensee purchase or use the Software for any such unintended or unauthorized application, Licensee shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the Software.

The term of this Agreement is for as long as Licensee uses the Software for its intended purpose and is not in default of any provisions of this Agreement. Motorola may terminate this Agreement if Licensee is in default of any of the terms and conditions of this Agreement.

This Agreement shall be governed by and construed in accordance with the laws of the State of Arizona and can only be modified in a writing signed by both parties. Licensee agrees to jurisdiction and venue in the State of Arizona.

By using, modifying, installing, compiling, or copying the Software, Licensee acknowledges that this Agreement has been read and understood and agrees to be bound by its terms and conditions. Licensee agrees that this Agreement is the complete and exclusive statement of the agreement between Licensee and Motorola and supersedes any earlier proposal or prior arrangement, whether oral or written, and any other communications relative to the subject matter of this Agreement.

Index

A

ACELP xi

C

CNG xi
Coder 3-15
Conventions x

D

Decoder 5-1
Demo Directory Structure 2-3
DSP xi
DSP56800E Reference Guide xi
DSP5685x User's Manual xi

E

Embedded SDK Programmer's Guide xi
Encoder 5-1

G

G.723,1A Dual Rate Speech Coder Library 1-1
G.723.1A Applications 6-1
G.723.1A Codec Library 5-1

I

I/O xi
IDE xi
Init_Cod_Cng 3-13
Init_Coder 3-9
Init_Dec_Cng 3-20
Init_Decod 3-18
Init_Vad 3-11
Interface 3-1
ITU-T Recommendation G.723.1 xi
ITU-T Recommendation G.723.1 Annex A xi

L

Linking Applications 5-1
LPC xi
LSB xi

M

MIPS xi
MP-MLQ xi
MSB xi

O

OMR xi
OnCE xi
Optional Directories 2-2

P

PC xi
PSVQ xi

R

Required Core Directories 2-1

S

SDK xi
Software License Agreement 1-1
SP xi
SPI xi
SR xi
SRC xi

V

VAD xi

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors/