Embedded SDK (Software Development Kit)

G.726 Speech Codec Library

SDK118/D
Rev. 2, 07/19/2002

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

Contents

List of Tables

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

List of Figures

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

List of Examples

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

About This Document

This manual describes the G.726 Adaptive Differential Pulse Code Modulation, (ADPCM), Codec algorithm for use with Motorola's Embedded Software Development Kit, (SDK).

Audience

This document targets software developers implementing G.726 ADPCM 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.726 ADPCM Codec Library Interfaces**--describes all of the G.726 Library functions
- **Chapter 4, Building the G.726 Codec Library**--tells how to execute the system library project build
- **Chapter 5, Linking Applications with the G.726 Codec Library**--describes the organization of the G.726 Library
- **Chapter 6, G.726 Applications**--describes the use of the G.726 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:

- *DSP56800 Family Manual*, DSP56800FM/AD
- *DSP56824 User's Manual*, DSP56824UM/AD
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Commands, command parameters, code examples, expressions, datatypes, and directives	...`*Foundational include files`... ...a data structure of type `vad_tConfigure`...
Italic	Calls, functions, statements, procedures, routines, arguments, file names and applications	...the *pConfig* argument... ...defined in the C header file, *g726_Enc.h*... ...makes a call to the *Callback* procedure...
Bold	Reference sources, paths, emphasis	...refer to the **Targeting DSP56824 Platform** manual.... ... see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
Bold/Italic	Directory name, project name	...and contains these core directories: ***applications*** contains applications software.... ...CodeWarrior project, ***3des.mcp***, is.....
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
ALL CAPITAL LETTERS	Variables, directives, defined constants, files libraries	INCLUDE_DSPFUNC #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.

ADPCM Adaptive Differential Pulse Code Modulation

DSP Digital Signal Processor or Digital Signal Processing

I/O Input/Output

IDE Integrated Development Environment

LSB	Least Significant Byte
MIPS	Million Instructions Per Second
MSB	Most Significant Byte
OnCE™	On-Chip Emulation
OMR	Operating Mode Register
PC	Program Counter
SDK	Software Development Kit
SP	Stack Pointer
SPI	Serial Peripheral Interface
SR	Status Register
SRC	Source

References

The following sources were referenced to produce this book:

[1] *DSP56800 Family Manual, DSP56800FM/AD*
[2] *DSP56824 User's Manual, DSP56824UM/AD*
[3] *Embedded SDK Programmer's Guide*
[4] *ITU-T Recommendation G.726*

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.726 ADPCM Codec 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.726 ADPCM Codec 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.726 ADPCM Codec 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 DSP56824 Platform** manual provides more specific information and examples about this hardware architecture. If you are developing an application for a DSP56824EVM board or any other DSP56824 development system, refer to the **Targeting Motorola DSP56824 Platform** manual for **Quick Start** or other DSP56824-specific information.

1.2 Overview of G.726 ADPCM Codec

The ITU-T Recommendation G.726 Adaptive Differential Pulse Code Modulation Speech Codec converts a 64Kbits/s A-Law or Mu-Law pulse code modulation (PCM) channel to and from 40, 32, 24, 16Kbits/s PCM stream with an ADPCM transcoding technique.

1.2.1 Background

The G.726 ADPCM Codec converts a 64Kbits/s A-Law or Mu-Law PCM channel to and from a 40, 32, 24, 16Kbits/s channel. The principal application of 24 and 16Kbits/s channels is as overload channels carrying voice in Digital Circuitry Multiplication Equipment (DCME). The principal application of 40Kbits/s channels is carrying data modem signals in DCME, especially for modems operating at greater than 4800 bits/s.

An A-Law or Mu-Law PCM input signal is converted to uniform PCM; a difference signal is obtained by subtracting an estimate of the input signal itself. An adaptive 31-, 15- , 7-, or 4-level quantizer is used to assign 5, 4, 3, 2 binary digits, respectively, to the value of difference signal for transmission to the decoder. An inverse quantizer produces a quantized difference signal from these same 5, 4, 3, 2 binary digits, respectively. The signal estimate is added to this quantized difference signal to produce the reconstructed version of the input signal. Both the reconstructed signal and the quantized difference signal are operated upon by an adaptive predictor which produces the estimate of the input signal, thereby completing the feedback loop.

The Decoder includes a structure identical to the feedback portion of the encoder, together with the uniform PCM to A-Law or Mu-Law conversion and a synchronous coding adjustment, (SCA). The SCA prevents cumulative distortion occurring on the synchronous tandem coding (ADPCM - PCM - ADPCM, etc. digital connections).

1.2.2 Features and Performance

The G.726 ADPCM Codec 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.

Chapter 2
Directory Structure

2.1 Required Core Directories

Figure 2-1 details required platform directories:

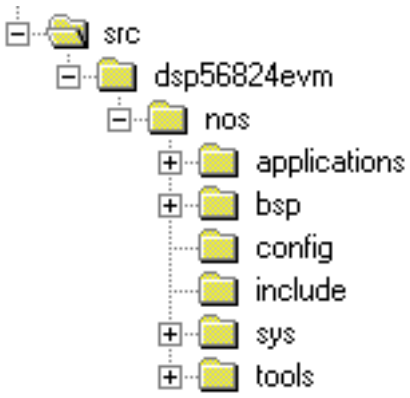

Figure 2-1. Core Directories

As shown in **Figure 2-1**, DSP56824EVM 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.726 ADPCM Codec is encapsulated in the domain-specific directory ***telephony***.

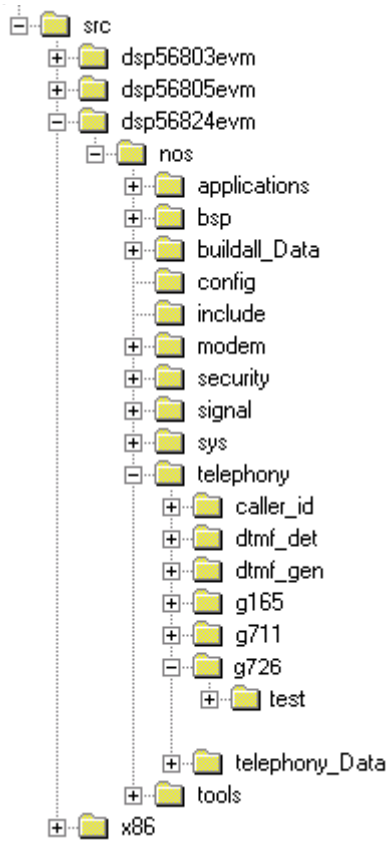

Figure 2-2. DSP56824 Directories

The ***g726*** ADPCM Codec directory includes G726 ADPCM Codec-specific algorithms. **Figure 2-3** shows the ***g726*** directory structure.

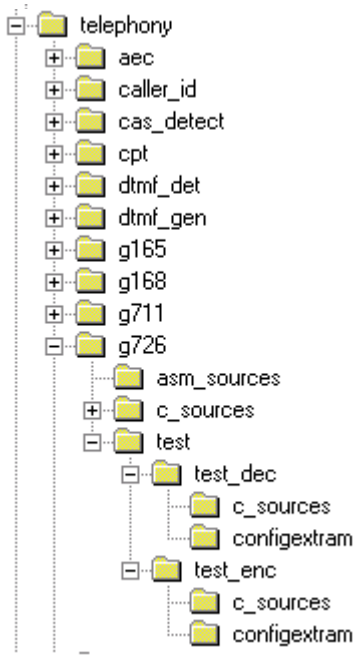

Figure 2-3. *G726* ADPCM Codec Directory Structure

The ***g726*** includes the following sub-directories:

- ***asm_sources*** includes all asm source files
- ***c_sources*** includes APIs for the G726 Encoder/Decoder
- ***test*** includes test directories for the encoder and decoder
- ***test_enc*** (encoder) and ***test_dec*** (decoder) files include *c_source* files and configuration necessary for testing G726 Encoder and Decoder library modules
  - — ***c_sources*** contains an example test code for the G726 Encoder/Decoder
  - — ***config*** contains configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to G726 Encoder/Decoder testing

2.3 Demo Directory Structure

Figure 2-4 demonstrates how the G.726 ADPCM Codec demo (***g726***) is encapsulated in the ***telephony*** directory under ***applications***.

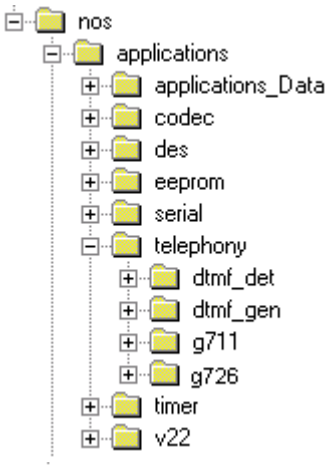

Figure 2-4. SDK Application Directory Structure

The ***g726*** directory includes G.726 ADPCM Codec demo-specific algorithms. Figure 2-5 shows the ***g726*** directory structure.

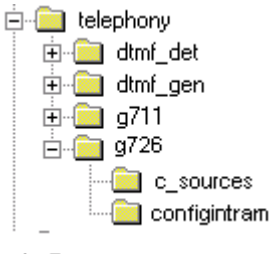

Figure 2-5. *g726* Directory Structure

- ***g726*** includes c sources and configuration necessary for testing G.726 ADPCM Codec demo modules
  - — ***c_sources*** contains an example demo code for the G.726 ADPCM Codec
  - — ***configintram*** contains configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to the G.726 ADPCM Codec demo

Chapter 3
G.726 ADPCM Codec Library Interfaces

3.1 G.726 ADPCM Codec Services

The principal application of 24 and 16 Kbits/s channels is for overload channels carrying voice in Digital Circuitry Multiplication Equipment, (DCME). The principal application of 40 Kbits/s channels is to carry data modem signals in DCME, especially for modems operating at greater than 4800 bits/s.

3.2 Interface

The C interface for the G.726 ADPCM Codec library service is defined in the C header files *g726.h*, shown in **Code Example 3-1** for 56800E targets and **Code Example 3-2** for 56800 targets.

SDK defines the G.726 ADPCM Codec library:

G.726 ADPCM Codec Library (***g726.lib***)

A description of the interfaces and services provided follow.

Code Example 3-1. C Header File *g726.h for 56800E targets*

```
/* File g726.h */

#ifndef _G726_H
#define _G726_H

/*************************************************************************
   FOR DECODER ONLY
 *************************************************************************/

/**************************
 Foundational Include Files
**************************/

#include "port.h"

/* Flag_RATE takes following values */
#define  G726_DEC_RATE_40       0
```

```
#define  G726_DEC_RATE_32       1
#define  G726_DEC_RATE_24       2
#define  G726_DEC_RATE_16       3

/* Flag_LAW takes following values */
#define  G726_DEC_MU_LAW        0
#define  G726_DEC_A_LAW         1

/*********************************
Commands for G726 Decoder control
*********************************/

/* Set the out Data RATE to 40,32,24 or 14 Kbps */
#define  G726_DEC_SET_RATE_40   1
#define  G726_DEC_SET_RATE_32   2
#define  G726_DEC_SET_RATE_24   3
#define  G726_DEC_SET_RATE_16   4

/* Set the LAW to MU-LAW or A-LAW */
#define  G726_DEC_SET_MU_LAW    5
#define  G726_DEC_SET_A_LAW     6

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = MU-LAW */
#define  G726_DEC_SET_MU_40     7
#define  G726_DEC_SET_MU_32     8
#define  G726_DEC_SET_MU_24     9
#define  G726_DEC_SET_MU_16     10

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = A-LAW */
#define  G726_DEC_SET_A_40      11
#define  G726_DEC_SET_A_32      12
#define  G726_DEC_SET_A_24      13
#define  G726_DEC_SET_A_16      14


/***********************************************
     Structure for G726 Decoder Configuration
***********************************************/

typedef struct
{
    UWord16            Flag_RATE;
    UWord16            Flag_LAW;
} G726_Dec_sConfigure;

typedef struct
{
    UWord16            Flag_RATE;
    UWord16            Flag_LAW;
    Int16              *DEC_QUANTAB;
    Int16              *DEC_IQUANTAB;
    Int16              *DEC_FITAB;
    Int16              *DEC_WITAB;
```

```
    Int16               DEC_QUAN_MASK;
    Int16               DEC_ANTILG_MASK;
    Frac16              SP_R;
    Frac16              PP_R[8];
    Frac16              DEC_DQI;
    Frac16              SEZ_R;
    Frac16              SE_R;
    Frac16              Y_R;
    Frac16              I_R;
    Frac16              SR_R;
    Frac16              COEF_R[8];
    Int16               TR_R;
    Int16               PK_R[2];
    Int16               TDP_R;
    Int16               TD_R;
    Frac16              DMS_R;
    Frac16              DML_R;
    Frac16              AP_R;
    Frac16              YU_R;
    Frac16              YL_R[2];
    Frac16              IMAG;
    Frac16              SIGPK;
    Frac16              LAW;
    Frac16              OUT_RATE;
    Frac16              *DATA_R;
    Frac16              DATAPTR_R;
    Int16               DUMMY;

} G726_Dec_sHandle;

/***************************
 Function Prototypes
***************************/

EXPORT G726_Dec_sHandle * G726DecCreate ( G726_Dec_sConfigure   *pConfig);

EXPORT Result   G726DecInit ( G726_Dec_sHandle      *pG726Dec,
                              G726_Dec_sConfigure   *pConfig);

EXPORT Result   G726Decoder ( G726_Dec_sHandle      *pG726Dec,
                              unsigned char         *pInSamples,
                              unsigned char         *pOutSamples,
                              UWord16               NumberSamples);

EXPORT Result   G726DecControl(  G726_Dec_sHandle   *pG726Dec,
                                 UWord16            Command);

EXPORT void G726DecDestroy ( G726_Dec_sHandle    *pG726Dec);


/*********************************************************************
   FOR ENCODER ONLY
 *********************************************************************/

/* Flag_RATE takes following values */
```

```
#define  G726_ENC_RATE_40       0
#define  G726_ENC_RATE_32       1
#define  G726_ENC_RATE_24       2
#define  G726_ENC_RATE_16       3

/* Flag_LAW takes following values */
#define  G726_ENC_MU_LAW        0
#define  G726_ENC_A_LAW         1

/*********************************
Commands for G726 Encoder control
*********************************/

/* Set the out Data RATE to 40,32,24 or 14 Kbps */
#define  G726_ENC_SET_RATE_40   1
#define  G726_ENC_SET_RATE_32   2
#define  G726_ENC_SET_RATE_24   3
#define  G726_ENC_SET_RATE_16   4

/* Set the LAW to MU-LAW or A-LAW */
#define  G726_ENC_SET_MU_LAW    5
#define  G726_ENC_SET_A_LAW     6

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = MU-LAW */
#define  G726_ENC_SET_MU_40     7
#define  G726_ENC_SET_MU_32     8
#define  G726_ENC_SET_MU_24     9
#define  G726_ENC_SET_MU_16     10

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = A-LAW */
#define  G726_ENC_SET_A_40      11
#define  G726_ENC_SET_A_32      12
#define  G726_ENC_SET_A_24      13
#define  G726_ENC_SET_A_16      14

/***********************************************
     Structure for G726 Encoder Configuration
***********************************************/

typedef struct
{
             UWord16            Flag_RATE;
             UWord16            Flag_LAW;
} G726_Enc_sConfigure;

/***********************************************
     Structure for G726 Context information. The
     application program should not modify anything
     defined in this strcture
***********************************************/
typedef struct
{
    UWord16            Flag_RATE;
```

```
    UWord16             Flag_LAW;
    Int16               *ENC_QUANTAB;
    Int16               *ENC_IQUANTAB;
    Int16               *ENC_FITAB;
    Int16               *ENC_WITAB;
    Int16               ENC_QUAN_MASK;
    Int16               ENC_ANTILG_MASK;
    Int16               SS_T;
    Frac16              PP_T[8];
    Frac16              ENC_DQI;
    Frac16              SEZ_T;
    Frac16              SE_T;
    Frac16              Y_T;
    Frac16              I_T;
    Frac16              SR_T;
    Frac16              COEF_T[8];
    Int16               TR_T;
    Int16               PK_T[2];
    Int16               TDP_T;
    Int16               TD_T;
    Frac16              DMS_T;
    Frac16              DML_T;
    Frac16              AP_T;
    Frac16              YU_T;
    Frac16              YL_T[2];
    Int16               ENC_IMAG;
    Frac16              ENC_SIGPK;
    Frac16              ENC_LAW;
    Frac16              ENC_OUT_RATE;
    Frac16              *DATA_T;
    Frac16              DATAPTR_T;
    Int16               ENC_DUMMY;

} G726_Enc_sHandle;

/***************************
 Function Prototypes
***************************/

EXPORT G726_Enc_sHandle * G726EncCreate ( G726_Enc_sConfigure   *pConfig);

EXPORT Result   G726EncInit ( G726_Enc_sHandle      *pG726Enc,
                              G726_Enc_sConfigure   *pConfig);

EXPORT Result   G726Encoder ( G726_Enc_sHandle      *pG726Enc,
                              unsigned char         *pInSamples,
                              unsigned char         *pOutSamples,
                              UWord16                NumberSamples);

EXPORT Result   G726EncControl ( G726_Enc_sHandle    *pG726Enc,
                                 UWord16              Command);

EXPORT void G726EncDestroy ( G726_Enc_sHandle *pG726Enc);

#endif
```

Code Example 3-2. C Header File *g726.h for 56800 targets*

```
/* File g726.h */

#ifndef _G726_H
#define _G726_H

/*************************************************************************
   FOR DECODER ONLY
 *************************************************************************/

/***************************
 Foundational Include Files
***************************/

#include "port.h"

/* Flag_RATE takes following values */
#define  G726_DEC_RATE_40       0
#define  G726_DEC_RATE_32       1
#define  G726_DEC_RATE_24       2
#define  G726_DEC_RATE_16       3

/* Flag_LAW takes following values */
#define  G726_DEC_MU_LAW        0
#define  G726_DEC_A_LAW         1

/*********************************
Commands for G726 Decoder control
*********************************/

/* Set the out Data RATE to 40,32,24 or 14 Kbps */
#define  G726_DEC_SET_RATE_40   1
#define  G726_DEC_SET_RATE_32   2
#define  G726_DEC_SET_RATE_24   3
#define  G726_DEC_SET_RATE_16   4

/* Set the LAW to MU-LAW or A-LAW */
#define  G726_DEC_SET_MU_LAW    5
#define  G726_DEC_SET_A_LAW     6

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = MU-LAW */
#define  G726_DEC_SET_MU_40     7
#define  G726_DEC_SET_MU_32     8
#define  G726_DEC_SET_MU_24     9
#define  G726_DEC_SET_MU_16     10

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = A-LAW */
#define  G726_DEC_SET_A_40      11
#define  G726_DEC_SET_A_32      12
#define  G726_DEC_SET_A_24      13
#define  G726_DEC_SET_A_16      14
```

```
/***********************************************
     Structure for G726 Decoder Configuration
***********************************************/

typedef struct
{
    UWord16            Flag_RATE;
    UWord16            Flag_LAW;
} G726_Dec_sConfigure;

typedef struct
{
UWord16            Flag_RATE;
    UWord16            Flag_LAW;
    // FMULT var.s
    Frac16             *COEF_R;   //COEF_T[8];
    Frac16             DATAPTR_R;
    Frac16             *PP_R;     //PP_T[8];
    Frac16             SEZ_R;
    Frac16             SE_R;
    Frac16             AP_R;
    Frac16             YU_R;
    Frac16             YL_R[2];
    Frac16             Y_R;
    // UPB var.s
    Int16              TD_R;
    Int16              TR_R;
    Int16              PK_R[2];
    Frac16             DEC_DQI;
    Frac16             SIGPK;
    Frac16             SR_R;
    Int16              TDP_R;
    Int16              *DEC_FITAB;
    Frac16             DMS_R;
    Frac16             DML_R;
    Int16              *DEC_WITAB;
    // EXPAND var.s
    Int16              SP_R;
    Frac16             LAW;
    Int16              *DEC_QUANTAB;
    Int16              *DEC_IQUANTAB;
    Int16              DEC_QUAN_MASK;
    Int16              DEC_ANTILG_MASK;
    Int16              IMAG;
    Frac16             I_R;
    Frac16             OUT_RATE;
    Frac16             *DATA_R;
    Int16              DUMMY;

} G726_Dec_sHandle;

/**************************
 Function Prototypes
**************************/
```

```
EXPORT G726_Dec_sHandle * G726DecCreate ( G726_Dec_sConfigure   *pConfig);

EXPORT Result   G726DecInit ( G726_Dec_sHandle      *pG726Dec,
                              G726_Dec_sConfigure   *pConfig);

EXPORT Result   G726Decoder ( G726_Dec_sHandle      *pG726Dec,
                              unsigned char         *pInSamples,
                              unsigned char         *pOutSamples,
                              UWord16               NumberSamples);

EXPORT Result   G726DecControl(  G726_Dec_sHandle   *pG726Dec,
                                 UWord16            Command);

EXPORT void G726DecDestroy ( G726_Dec_sHandle    *pG726Dec);


/*********************************************************************
   FOR ENCODER ONLY
 *********************************************************************/

/* Flag_RATE takes following values */
#define  G726_ENC_RATE_40       0
#define  G726_ENC_RATE_32       1
#define  G726_ENC_RATE_24       2
#define  G726_ENC_RATE_16       3

/* Flag_LAW takes following values */
#define  G726_ENC_MU_LAW        0
#define  G726_ENC_A_LAW         1

/*********************************
Commands for G726 Encoder control
*********************************/

/* Set the out Data RATE to 40,32,24 or 14 Kbps */
#define  G726_ENC_SET_RATE_40   1
#define  G726_ENC_SET_RATE_32   2
#define  G726_ENC_SET_RATE_24   3
#define  G726_ENC_SET_RATE_16   4

/* Set the LAW to MU-LAW or A-LAW */
#define  G726_ENC_SET_MU_LAW    5
#define  G726_ENC_SET_A_LAW     6

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = MU-LAW */
#define  G726_ENC_SET_MU_40     7
#define  G726_ENC_SET_MU_32     8
#define  G726_ENC_SET_MU_24     9
#define  G726_ENC_SET_MU_16     10

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = A-LAW */
#define  G726_ENC_SET_A_40      11
#define  G726_ENC_SET_A_32      12
```

```
#define  G726_ENC_SET_A_24      13
#define  G726_ENC_SET_A_16      14

/*********************************************
     Structure for G726 Encoder Configuration
*********************************************/

typedef struct
{
             UWord16            Flag_RATE;
             UWord16            Flag_LAW;
} G726_Enc_sConfigure;

/*********************************************
     Structure for G726 Context information. The
     application program should not modify anything
     defined in this strcture
*********************************************/
typedef struct
{
UWord16            Flag_RATE;
    UWord16            Flag_LAW;
    // FMULT var.s
    Frac16             *COEF_T;   //COEF_T[8];
    Frac16             DATAPTR_T;
    Frac16             *PP_T;     //PP_T[8];
    Frac16             SEZ_T;
    Frac16             SE_T;
    Frac16             AP_T;
    Frac16             YU_T;
    Frac16             YL_T[2];
    Frac16             Y_T;
    // UPB var.s
    Int16              TD_T;
    Int16              TR_T;
    Int16              PK_T[2];
    Frac16             ENC_DQI;
    Frac16             ENC_SIGPK;
    Frac16             SR_T;
    Int16              TDP_T;
    Int16              *ENC_FITAB;
    Frac16             DMS_T;
    Frac16             DML_T;
    Int16              *ENC_WITAB;
    // EXPAND var.s
    Int16              SS_T;
    Frac16             ENC_LAW;
    Int16              *ENC_QUANTAB;
    Int16              *ENC_IQUANTAB;
    Int16              ENC_QUAN_MASK;
    Int16              ENC_ANTILG_MASK;
    Int16              ENC_IMAG;
    Frac16             I_T;
    Frac16             ENC_OUT_RATE;
    Frac16             *DATA_T;
```

```
    Int16               ENC_DUMMY;

} G726_Enc_sHandle;

/**************************
 Function Prototypes
**************************/

EXPORT G726_Enc_sHandle * G726EncCreate ( G726_Enc_sConfigure   *pConfig);

EXPORT Result   G726EncInit ( G726_Enc_sHandle      *pG726Enc,
                              G726_Enc_sConfigure   *pConfig);

EXPORT Result   G726Encoder ( G726_Enc_sHandle      *pG726Enc,
                              unsigned char         *pInSamples,
                              unsigned char         *pOutSamples,
                              UWord16                NumberSamples);

EXPORT Result   G726EncControl ( G726_Enc_sHandle    *pG726Enc,
                                 UWord16              Command);

EXPORT void G726EncDestroy ( G726_Enc_sHandle *pG726Enc);

#endif
```

3.3 Specifications

The following pages describe the G.726 ADPCM 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 *G726EncCreate*

Call(s):

```
G726_Enc_sHandle * G726EncCreate ( G726_Enc_sConfigure *pConfig)
```

Required Header: "g726.h"

Arguments:

Table 3-1. *G726EncCreate* Arguments

pConfig	in	A pointer to a data structure containing data for initializing the G.726 Encoder algorithm

Description: The *G726EncCreate* function will create an instance of G726 Encoder. Multiple instances of G726 encoder are possible and each instance allocates **73** words of data memory. The parameter *pConfig* points to a data structure of type *G726_Enc_sConfigure*; its fields initialize G.726 Encoder operation Refer to **Section 3.3.2**, *G726EncInit*, for more information. The library allocates memory dynamically using the *mem* library routines shown in **Code Example 3-3**. Memory allocation is done in the **internal** memory for DSP5685x targets and the allocation is done in the **external** memory for DSP5682x targets.

Code Example: **Code Example 3-3** depicts the external memory allocation for dsp5682x targets.

Code Example 3-3. *mem* Library for 56800E targets

```
#include "mem.h"
#include "g726.h"

G726_Enc_sHandle * G726EncCreate ( G726_Enc_sConfigure   *pConfig)
{
    G726_Enc_sHandle *pG726Enc;
    Result  result;

    pG726Enc = (G726_Enc_sHandle *) memMallocEM (sizeof (G726_Enc_sHandle));

    if (pG726Enc == NULL) return (NULL);

    pG726Enc->DATA_T = (Frac16 *) memMallocAlignedEM (24 * sizeof (Frac16));
    if (pG726Enc->DATA_T == NULL)
    {
        G726EncDestroy (pG726Enc);
        return (NULL);
    }

    result = G726EncInit (pG726Enc, pConfig);

    return (pG726Enc);
}
```

Code Example 3-4. *mem* Library for 56800 targets

```
#include "mem.h"
#include "g726.h"

G726_Dec_sHandle * G726DecCreate ( G726_Dec_sConfigure   *pConfig)
{
```

```
    G726_Dec_sHandle *pG726Dec;
    Result res;

    pG726Dec = (G726_Dec_sHandle *) memMallocIM (sizeof (G726_Dec_sHandle));
    if (pG726Dec == NULL) return (NULL);

    pG726Dec->DATA_R = (Frac16 *) memMallocAlignedIM (24 * sizeof (Frac16));
    if (pG726Dec->DATA_R == NULL)
    {
        G726DecDestroy (pG726Dec);
        return (NULL);
    }

    if (memIsAligned (pG726Dec->DATA_R, 24 * sizeof (Frac16)) == false )
    {
        G726DecDestroy (pG726Dec);
        return (NULL);
    }

    pG726Dec->COEF_R = (Frac16 *) memMallocIM (8 * sizeof (Frac16));
    if (pG726Dec->COEF_R == NULL)
    {
        G726DecDestroy (pG726Dec);
        return (NULL);
    }

    pG726Dec->PP_R = (Frac16 *) memMallocIM (8 * sizeof (Frac16));
    if (pG726Dec->PP_R == NULL)
    {
        G726DecDestroy (pG726Dec);
        return (NULL);
    }

    /* Call to init () */
    res = G726DecInit (pG726Dec, pConfig);

    return (pG726Dec);
}
```

For details on the *G726_Enc_sHandle* structure, please refer to the *g726.h* file in **Code Example 3-1**. The *pConfig* argument points to the *G726_Enc_sConfigure* structure used to configure G726 Encoder operation; see **Code Example 3-5**.

If a *G726EncCreate* function is called to create an instance, then *G726EncDestroy*, detailed in **Section 3.3.5**, should be used to destroy the instance.

Alternatively, the user can allocate memory statically which requires duplicating all statements in the *G726EncCreate* function. In this case, the user can call the *G726EncInit* function directly, bypassing the *G726EncCreate* function. If user allocates the memory statically, bypassing the call to *create()*, allocate the memory in the internal memory space to obtain the performance quoted in the data sheet.

If the user dynamically allocates memory without calling *G726EncCreate*, then the user himself must destroy the memory allocated.

Returns: Upon successful completion, *G726EncCreate* returns a pointer of type *G726_Enc_sHandle*. If the data memory is not available, *G726EncCreate* will return “NULL”.

Special Considerations:

- The G.726 Encoder application is multichannel and re-entrant.
- If *G726EncCreate* is called, then the user need not call *G726EncInit* function, as it is called internally in the *G726EncCreate* function.

Code Example 3-5. Use of the *G726EncCreate* Interface

```
#include “g726.h”
#include “mem.h”

void test_g726 (void)
{
   G726_Enc_sConfigure  *pConfig;
   G726_Enc_sHandle     *pG726Enc;
   Result result;

   /* Initialize the configuration flags */

   pConfig->Flag_RATE   = G726_ENC_RATE_40;
   pConfig->Flag_LAW    = G726_ENC_MU_LAW;

   /* Input data:  | psss qqqq | 0000 0000 | (PCM log format) */
   pConfig->Flag_Format = G726_ENC_MSB; /* This is only for dsp5682x targets */


   pConfig = (G726_Enc_sConfigure *) memMallocEM (sizeof (G726_Enc_sConfigure));

   pG726Enc = G726EncCreate (pConfig); /* Create and init encoder */
}
```

3.3.2 *G726EncInit*

Call(s):

```
Result G726EncInit ( G726_Enc_sHandle *pG726Enc, G726_Enc_sConfigure *pConfig)
```

Required Header: "g726.h"

Arguments:

Table 3-2. *G726EncInit* Arguments

pG726Enc	in	A pointer to a structure of type *G726_Enc_sHandle*
pConfig	in	A pointer to a data structure containing data for initializing the G.726 Encoder algorithm

Description: The *G726EncInit* function will initialize the G.726 Encoder algorithm. During initialization, all resources will be set to their initial values in preparation for G.726 Encoder operation. The *G726EncInit* function should be called only once before the first call to the *G726Encoder* function

The parameter *pConfig* points to a data structure of type *G726_Enc_sConfigure*; its fields initialize G.726 Encoder operation in the following manner:

Flag_RATE - To set the encoding data rate:

G726_ENC_SET_RATE_40 - Sets the encoder rate as 40 kbps
G726_ENC_SET_RATE_32 - Sets the encoder rate as 32 kbps
G726_ENC_SET_RATE_24 - Sets the encoder rate as 24 kbps
G726_ENC_SET_RATE_16 - Sets the encoder rate as 16 kbps

Flag_LAW - To select between A-law and MU-law encoding scheme:

G726_ENC_MU_LAW - Input is MU-Law encoded
G726_ENC_A_LAW - Input is A-Law encoded

Flag_Format - /* This is only for DSP5682x targets */

G726_ENC_MSB - 8-bit input data (A-Law / MU-Law encoded as set in the Flag_Format described above) is available for encoding in the Most significant part of 16-bit data word
G726_ENC_LSB - 8-bit input data (A-Law / MU-Law encoded as set in the Flag_Format described above) is available for encoding in the Least significant part of 16-bit data word

Returns: Upon successful completion, a value of "PASS" will be returned; otherwise, a value of "FAIL" will be returned.

Special Considerations:

- If *G726EncCreate* is called, then the user need not call *G726EncInit* function as it is called internally in the *G726EncCreate* function.

Code Example 3-6. Use of the *G726EncInit* Interface

```
#include “g726.h”
#include “mem.h”

void test_g726 (void)
{
   G726_Enc_sConfigure  *pConfig;
   G726_Enc_sHandle     *pG726Enc;
   Result res;

   /* Initialize the configuration flags */

   pConfig->Flag_RATE   = G726_ENC_RATE_40;
   pConfig->Flag_LAW    = G726_ENC_MU_LAW;

   /* Input data:  | psss qqqq | 0000 0000 | (PCM log format) */
   pConfig->Flag_Format = G726_ENC_MSB; /* This is only for DSP5682x targets */
   pConfig = (G726_Enc_sConfigure *) memMallocEM (sizeof (G726_Enc_sConfigure));

   /* Statically allocate the encoder-instance */
   ...
   res = G726EncInit ( pG726Enc, pConfig);
}
```

3.3.3 *G726Encoder*

Call(s):

```
Result G726Encoder ( G726_Enc_sHandle *pG726Enc,
                     unsigned char *pInSamples,
                     unsigned char *pOutSamples,
                     UWord16 NumberSamples);
```

Required Header: "g726.h"

Arguments :

Table 3-3. *G726Encoder* Arguments

pG726Enc	in	Pointer to the *G726_Enc_sHandle* structure
pInSamples	in	Pointer to input data buffer
pOutSamples	out	Pointer to output data buffer
NumberSamples	in	The number of samples as input

Description: The *G726Encoder* function will process the 64 kbps A_LAW or MU_LAW compressed speech samples and encode it to either 40, 32, 24 or 16 kbps output rate. The Encoder expects the input in either of the following formats:

input data = | psss qqqq | xxxx xxxx | (PCM log format)

input data = | xxxx xxxx | psss qqqq | (PCM log format)

Whether the input occupies the UPPER byte or the LOWER byte of a 16-bit word, needs to be specified through the flag *Flag_Format* in *G726_Enc_sConfigure*, defined in the *g726.h* file. **This is for DSP5682x targets only and does not apply for DSP5685x targets.**

The encoded samples are present in the *pOutSamples* buffer and in the following format:

output data = | ssss ssss| ssss xxxx | for 40 kbps

output data = | ssss ssss| ssss sxxx | for 32 kbps

output data = | ssss ssss| ssss ssxx | for 24 kbps

output data = | ssss ssss| ssss sssx | for 16 kbps

Returns: *G726Encoder* will return "PASS" after successful encoding.

Special Considerations: The *G726Encoder* function expects A-Law or Mu-Law coded 8-bit samples, sampled at 8 KHz only.

Code Example 3-7. Use of *G726Encoder* Interface

```
#include “g726.h”
#include “mem.h”

void test_g726 (void)
{
   G726_Enc_sConfigure  *pConfig;
   G726_Enc_sHandle     *pG726Enc;
   Result res;
   UWord16 NumSamples = 80;
   unsigned char SampleBuf[80], OutBuff[80];

   /* Initialize the configuration flags */

   pConfig->Flag_RATE   = G726_ENC_RATE_40;
   pConfig->Flag_LAW    = G726_ENC_MU_LAW;

   /* Input data:  | psss qqqq | 0000 0000 | (PCM log format) */
   pConfig->Flag_Format = G726_ENC_MSB; /* This is only for DSP5682x targets */

   pConfig = (G726_Enc_sConfigure *) memMallocEM (sizeof (G726_Enc_sConfigure));

   pG726Enc = G726EncCreate (pConfig);
   ...
   result = G726Encoder ( pG726Enc, SampleBuf, OutBuff, NumSamples);
}
```

3.3.4 *G726EncControl*

Call(s):

```
Result G726EncControl ( G726_Enc_sHandle *pG726Enc, UWord16  Command)
```

Required Header: "g726.h"

Arguments :

Table 3-4. *G726EncControl* Arguments

pG726Enc	in	Pointer to the *G726_Enc_sHandle* structure
Command	in	Variable which holds the user command for setting encoder flags

Description: The *G726EncControl* function sets the Configuration Flags for data RATE and compression LAW in the *pConfig* structure as requested by the user and calls the *G726EncInit* function to activate the changes.The set of commands, defined in the *g726.h* file, follows:

```
/* Set the out Data RATE to 40,32,24 or 16 Kbps */
#define  SET_ENC_RATE_40   1
#define  SET_ENC_RATE_32   2
#define  SET_ENC_RATE_24   3
#define  SET_ENC_RATE_16   4

/* Set the LAW to MU-LAW or A-LAW */
#define  SET_ENC_MU_LAW    5
#define  SET_ENC_A_LAW     6

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = MU-LAW */
#define  SET_ENC_MU_40     7
#define  SET_ENC_MU_32     8
#define  SET_ENC_MU_24     9
#define  SET_ENC_MU_16     10

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = A-LAW */
#define  SET_ENC_A_40      11
#define  SET_ENC_A_32      12
#define  SET_ENC_A_24      13
#define  SET_ENC_A_16      14
```

Returns: Upon successful completion of the command, the function returns "PASS"; otherwise, it returns "FAIL".

Special Considerations: This function internally calls an assembly function to change either the data RATE (40,32,24,16 Kbps), or the PCM encoding scheme (MU-LAW or A-LAW), or both.

Code Example 3-8. Use of *G726EncControl* Interface

```
#include “g726.h”
#include “mem.h”

void test_g726(void)
{
   G726_Enc_sConfigure  *pConfig;
   G726_Enc_sHandle     *pG726Enc;
   Result res;
   UWord16 NumSamples = 80;
   unsigned char SampleBuf[80], OutBuff[80];

   /* Initialize the configuration flags */

   pConfig->Flag_RATE   = G726_ENC_RATE_40;
   pConfig->Flag_LAW    = G726_ENC_MU_LAW;

   /* Input data:  | psss qqqq | 0000 0000 | (PCM log format) */
   pConfig->Flag_Format = G726_ENC_MSB; /* This is only for DSP5682x targets */

   pConfig = (G726_Enc_sConfigure *) memMallocEM (sizeof (G726_Enc_sConfigure));

   pG726Enc = G726EncCreate (pConfig);
   ...
   result = G726Encoder ( pG726Enc, SampleBuf, OutBuff, NumSamples);

   /* set encoder for 16 Kbits/s and A-Law */
   result = G726EncControl ( pG726Enc, SET_ENC_A_16);
}
```

3.3.5 *G726EncDestroy*

Call(s):

```
void G726EncDestroy ( G726_Enc_sHandle *pG726Enc)
```

Required Header: "g726.h"

Arguments:

Table 3-5. *G726EncDestroy* Arguments

pG726Enc	in	Handle to an instance of G726 Encoder generated by a call to *G726EncCreate*

Description: The *G726EncDestroy* function destroys the instance of the G726 Encoder originally created by a call to *G726EncCreate*.

If the user had bypassed the *G726EncCreate* function to create an instance on his own, *G726EncDestroy* function should not be called.

Returns: None

Special Considerations: None

Code Example 3-9. Use of the *G726EncDestroy* Interface

```
#include "g726.h"
#include "mem.h"

void test_g726(void)
{
   G726_Enc_sConfigure  *pConfig;
   G726_Enc_sHandle     *pG726Enc;
   Result res;
   UWord16 NumSamples = 80;
   unsigned char SampleBuf[80], OutBuff[80];

   /* Initialize the configuration flags */

   pConfig->Flag_RATE   = G726_ENC_RATE_40;
   pConfig->Flag_LAW    = G726_ENC_MU_LAW;

   /* Input data:  | psss qqqq | 0000 0000 | (PCM log format) */
   pConfig->Flag_Format = G726_ENC_MSB; /* This is only for DSP5682x targets */

   pConfig = (G726_Enc_sConfigure *) memMallocEM (sizeof (G726_Enc_sConfigure));

   pG726Enc = G726EncCreate (pConfig);
   ...
   result = G726Encoder ( pG726Enc, SampleBuf, OutBuff, NumSamples);

   /* set encoder for 16 Kbits/s and A-Law */
   result = G726EncControl ( pG726Enc, SET_ENC_A_16);

   G726EncDestroy (pG726Enc);
}
```

3.3.6 *G726DecCreate*

Call(s):

```
G726_Dec_sHandle * G726DecCreate ( G726_Dec_sConfigure *pConfig)
```

Required Header: "g726.h"

Arguments:

Table 3-6. *G726DecCreate* Arguments

pConfig	in	A pointer to a data structure containing data for initializing the G.726 Decoder algorithm

Description: The *G726DecCreate* function will create an instance of G726 Decoder. Multiple instances of G726 decoder are possible; each instance allocates **73** words of data memory. The parameter *pConfig* points to a data structure of type *G726_Dec_sConfigure*; its fields initialize G.726 Decoder operation. See **Section 3.3.7**, *G726DecInit*, for more information. The library allocates memory dynamically using the *mem* library routines shown in **Code Example 3-10**. Memory allocation is done in the **internal** memory for DSP5685x targets and allocation is done in the **external** memory for dsp5682x targets.

Code Example: **Code Example 3-10** depicts the external memory allocation for DSP5682x targets.

Code Example 3-10. *mem* Library

```
#include "mem.h"
#include "g726.h"

G726_Dec_sHandle * G726DecCreate ( G726_Dec_sConfigure   *pConfig)
{
    G726_Dec_sHandle *pG726Dec;
    Result result;

    pG726Dec = (G726_Dec_sHandle *) memMallocEM (sizeof (G726_Dec_sHandle));
    if (pG726Dec == NULL) return (NULL);

    pG726Dec->DATA_R = (Frac16 *) memMallocAlignedEM (24 * sizeof (Frac16));
    if (pG726Dec->DATA_R == NULL)
    {
        G726DecDestroy (pG726Dec);
        return (NULL);
    }

    result = G726DecInit (pG726Dec,pConfig);

    return (pG726Dec);
}
```

For details on the *G726_Dec_sHandle* structure, please refer to the *g726.h* file in **Code Example 3-1**. The *pConfig* argument above points to the *G726_Decc_sConfigure* structure used to configure G726 Decoder operation; see **Code Example 3-11**.

If a *G726DecCreate* function is called to create an instance, then *G726DecDestroy*, detailed in **Section 3.3.10**, should be used to destroy the instance.

Alternatively, the user can allocate memory statically, which requires duplicating all statements in the *G726DecCreate* function. In this case, the user can call the *G726DecInit* function directly, bypassing the *G726DecCreate* function. If user allocates the memory statically, bypassing the call to *create()*, allocate the memory in the internal memory space to obtain the performance quoted in the data sheet.

If the user dynamically allocates memory without calling *G726DecCreate*, then the user himself must destroy the memory allocated.

Returns: Upon successful completion, *G726DecCreate* returns a pointer of type *G726_Dec_sHandle*. If data memory is not available, *G726DecCreate* will return “NULL”.

Special Considerations:

- The G.726 Decoder application is multichannel and re-entrant.
- If *G726DecCreate* is called, then the user need not call *G726DecInit* function as it is called internally in the *G726DecCreate* function.

Code Example 3-11. Use of the *G726DecCreate* Interface

```
#include “g726.h”
#include "mem.h"

test_g726_dec (void)
{
   G726_Dec_sConfigure  *pConfig;
   G726_Dec_sHandle     *pG726Dec;
   Result result;

   pConfig = (G726_Dec_sConfigure *) memMallocEM (sizeof (G726_Dec_sConfigure));
   /* Initialize the config flags */

   pConfig->Flag_RATE    = G726_DEC_RATE_40;
   pConfig->Flag_LAW     = G726_DEC_MU_LAW;
   pConfig->Flag_Format  = G726_DEC_MSB; /* This is only for dsp5682x targets */

   pG726Dec = G726DecCreate (pConfig);
}
```

3.3.7 *G726DecInit*

Call(s):

```
Result G726DecInit ( G726_Dec_sHandle *pG726Dec, G726_Dec_sConfigure *pConfig);
```

Required Header: "g726.h"

Arguments:

Table 3-7. *G726DecInit* Arguments

pG726Dec	in	Pointer to the structure of type *G726_Dec_sHandle*
pConfig	in	Pointer to the *G726_Dec_sConfigure* structure used to configure G.726 Decoder operation.

Description: The *G726DecInit* function will initialize the G.726 Decoder algorithm. During initialization, all resources will be set to their initial values in preparation for G.726 Decoder operation. The *G726DecInit* function should be called only once before the first call to the *G726Decoder* function

The parameter *pConfig* points to a data structure of type *G726_Dec_sConfigure*; its fields initialize G.726 Decoder operation in the following manner:

Flag_RATE - To set the decoding data rate

G726_DEC_SET_RATE_40 - Sets the decoder rate as 40 kbps
G726_DEC_SET_RATE_32 - Sets the decoder rate as 32 kbps
G726_DEC_SET_RATE_24 - Sets the decoder rate as 24 kbps
G726_DEC_SET_RATE_16 - Sets the decoder rate as 16 kbps

Flag_LAW - To select between A-law and MU-law decoding scheme.

G726_DEC_MU_LAW - Need the output in MU-Law format
G726_DEC_A_LAW - Need the output in A-Law format

Flag_Format - /* This is only for DSP5682x targets */

G726_DEC_MSB - Place the 8-bit output data (A-Law / MU-Law encoded as set by the Flag_LAW above) in Most significant part of 16-bit data word
G726_DEC_LSB - Place the 8-bit output data (A-Law / MU-Law encoded as set by the Flag_LAW above) in Least significant part of 16-bit data word

Returns: Upon successful completion, a value of "PASS" will be returned; otherwise, a value of "FAIL" will be returned.

Special Considerations:

- If *G726DecCreate* is called, then the user need not call *G726DecInit* function as it is called internally in the *G726DecCreate* function.

Code Example 3-12. Use of the *G726DecInit* Interface

```
#include “g726.h”
#include "mem.h"

test_g726_dec (void)
{
   G726_Dec_sConfigure  *pConfig;
   G726_Dec_sHandle     *pG726Dec;
   Result result;

   pConfig = (G726_Dec_sConfigure *) memMallocEM (sizeof (G726_Dec_sConfigure));
   /* Initialize the config flags */

   pConfig->Flag_RATE    = G726_DEC_RATE_40;
   pConfig->Flag_LAW     = G726_DEC_MU_LAW;
   pConfig->Flag_Format  = G726_DEC_MSB; /* This is only for DSP5682x targets */

   /* Statically create the decoder-instance */
   ...
   res = G726DecInit ( pG726Dec, pConfig);
}
```

3.3.8 *G726Decoder*

Call(s):

```
Result G726Decoder ( G726_Dec_sHandle *pG726Dec,
                     unsigned char  *pInSamples,
                     unsigned char  *pOutSamples,
                     UWord16 NumberSamples);
```

Required Header: "g726.h"

Arguments :

Table 3-8. *G726Decoder* Arguments

pG726Dec	in	Pointer to the *G726_Dec_sHandle* structure
pInSamples	in	Pointer to the input data buffer
pOutSamples	out	Pointer to output buffer
NumSamples	in	The number of samples as input

Description: The *G726Decoder* function will process the 40, 32, 24 or 16 kbps input rate and decode it to 64 kbps A_LAW or MU_LAW compressed speech samples. The decoder outputs in one of the following formats:

output data = | psss qqqq | xxxx xxxx | (PCM log format)

output data = | xxxx xxxx | psss qqqq | (PCM log format)

Whether the output occupies the UPPER byte or the LOWER byte of a 16-bit word must be specified through the flag *Flag_Format* in *G726_Dec_sConfigure*, defined in the *g726.h* file. **This is for DSP5682x targets only and does not apply for DSP5685x targets.**

The decoded samples are present in the *pOutSamples* buffer.

The encoded samples are present in the *pInSamples* buffer and the encoded samples (input) are in the following format:

input data = | ssss ssss| ssss xxxx | for 40 kbps

input data = | ssss ssss| ssss sxxx | for 32 kbps

input data = | ssss ssss| ssss ssxx | for 24 kbps

input data = | ssss ssss| ssss sssx | for 16 kbps

Returns: The *G726Decoder* function will return "PASS" after successful decoding.

Special Considerations: The *G726Decoder* function expects any one of the 40, 32, 24, 16 kbps rate.

Code Example 3-13. Use of the *G726Decoder* Interface

```
#include “g726.h”
#include "mem.h"

test_g726_dec (void)
{
   G726_Dec_sConfigure  *pConfig;
   G726_Dec_sHandle     *pG726Dec;
   Result result;
   UWord16 NumSamples = 80;
   unsigned char SampleBuf[80], OutBuff[80];

   pConfig = (G726_Dec_sConfigure *) memMallocEM (sizeof (G726_Dec_sConfigure));
   /* Initialize the config flags */

   pConfig->Flag_RATE    = G726_DEC_RATE_40;
   pConfig->Flag_LAW     = G726_DEC_MU_LAW;
   pConfig->Flag_Format  = G726_DEC_MSB; /* This is only for dsp5682x targets */

   pG726Dec = G726DecCreate (pConfig);
   ...
   result = G726Decoder ( pG726Dec, SampleBuf, OutBuff, NumSamples);
}
```

3.3.9 G726DecControl

Call(s):

```
Result G726DecControl ( G726_Dec_sHandle *pG726Dec, UWord16  Command);
```

Required Header: "g726.h"

Arguments :

Table 3-9. *G726DecControl* Arguments

pConfig	in	Pointer to the *G726_Dec_sHandle* structure
Command	in	Variable which holds the user command for setting decoder flags

Description: The *G726DecControl* function sets the Configuration Flags for data RATE and compression LAW in the *pConfig* structure as requested by the user and calls the *assembly init* function to activate these changes. The set of commands, defined in the *g726.h* file, follows:

```
/* Set the out Data RATE to 40,32,24 or 16 Kbps */
#define  SET_DEC_RATE_40   1
#define  SET_DEC_RATE_32   2
#define  SET_DEC_RATE_24   3
#define  SET_DEC_RATE_16   4

/* Set the LAW to MU-LAW or A-LAW */
#define  SET_DEC_MU_LAW    5
#define  SET_DEC_A_LAW     6

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = MU-LAW */
#define  SET_DEC_MU_40     7
#define  SET_DEC_MU_32     8
#define  SET_DEC_MU_24     9
#define  SET_DEC_MU_16     10

/* Set out Data RATE = 40,32,24 or 16 KBPS
   and LAW = A-LAW */
#define  SET_DEC_A_40      11
#define  SET_DEC_A_32      12
#define  SET_DEC_A_24      13
#define  SET_DEC_A_16      14
```

Returns: Upon successful completion of the command, the function returns "PASS"; otherwise, it returns "FAIL".

Special Considerations: None

Code Example 3-14. Use of the *G726DecControl* Interface

```
#include “g726.h”
#include "mem.h"

test_g726_dec (void)
{
   G726_Dec_sConfigure  *pConfig;
   G726_Dec_sHandle     *pG726Dec;
   Result result;
   UWord16 NumSamples = 80;
   unsigned char SampleBuf[80], OutBuff[80];

   pConfig = (G726_Dec_sConfigure *) memMallocEM (sizeof (G726_Dec_sConfigure));
   /* Initialize the config flags */

   pConfig->Flag_RATE    = G726_DEC_RATE_40;
   pConfig->Flag_LAW     = G726_DEC_MU_LAW;
   pConfig->Flag_Format  = G726_DEC_MSB; /* This is only for dsp5682x targets */

   pG726Dec = G726DecCreate (pConfig);
   ...
   result = G726Decoder ( pG726Dec, SampleBuf, OutBuff, NumSamples);

   /* set Decoder for 16 Kbits/s and A-Law */
   result = G726DecControl ( pG726Dec, SET_DEC_A_16);
}
```

3.3.10 G726DecDestroy

Call(s):

```
void G726DecDestroy ( G726_Dec_sHandle *pG726Dec)
```

Required Header: "g726.h"

Arguments:

Table 3-10. *G726DecDestroy* Arguments

pG726Dec	in	Handle to an instance of *G726Decoder* generated by a call to *G726DecCreate*

Description: The *G726DecDestroy* function destroys the instance of the *G726Decoder* originally created by a call to *G726DecCreate*.

If the user bypassed the *G726DecCreate* function to create an instance on his own, the *G726DecDestroy* function should not be called.

Returns: None

Special Considerations: None

Code Example 3-15. Use of the *G726DecDestroy* Interface

```
#include "g726.h"
#include "mem.h"

test_g726_dec (void)
{
   G726_Dec_sConfigure  *pConfig;
   G726_Dec_sHandle     *pG726Dec;
   Result result;
   UWord16 NumSamples = 80;
   unsigned char SampleBuf[80], OutBuff[80];

   pConfig = (G726_Dec_sConfigure *) memMallocEM (sizeof (G726_Dec_sConfigure));
   /* Initialize the config flags */

   pConfig->Flag_RATE    = G726_DEC_RATE_40;
   pConfig->Flag_LAW     = G726_DEC_MU_LAW;
   pConfig->Flag_Format  = G726_DEC_MSB; /* This is only for dsp5682x targets */

   pG726Dec = G726DecCreate (pConfig);
   ...
   result = G726Decoder ( pG726Dec, SampleBuf, OutBuff, NumSamples);

   /* set Decoder for 16 Kbits/s and A-Law */
   result = G726DecControl ( pG726Dec, SET_DEC_A_16);

   G726DecDestroy (pG726Dec);
}
```

Chapter 4
Building the G.726 Codec Library

4.1 Building the G.726 Codec Library

The G.726 Codec library combines all of the components described in the previous sections into one library: *g726.lib*. To build this library, Metrowerks' CodeWarrior project ***g726.mcp*** is provided. This project and all the necessary components to build the G.726 Codec library are located in the ***....\nos\telephony\g726*** directory of the SDK directory structure.

There are two methods to execute the system library project build: dependency build and direct build.

4.1.1 Dependency Build

Dependency build is the easiest approach and requires no additional work on the user's part. If you add the ***g726.mcp*** project**,** shown in **Figure 4-1,** to your application project, the *g726* library will automatically build when the application is built.

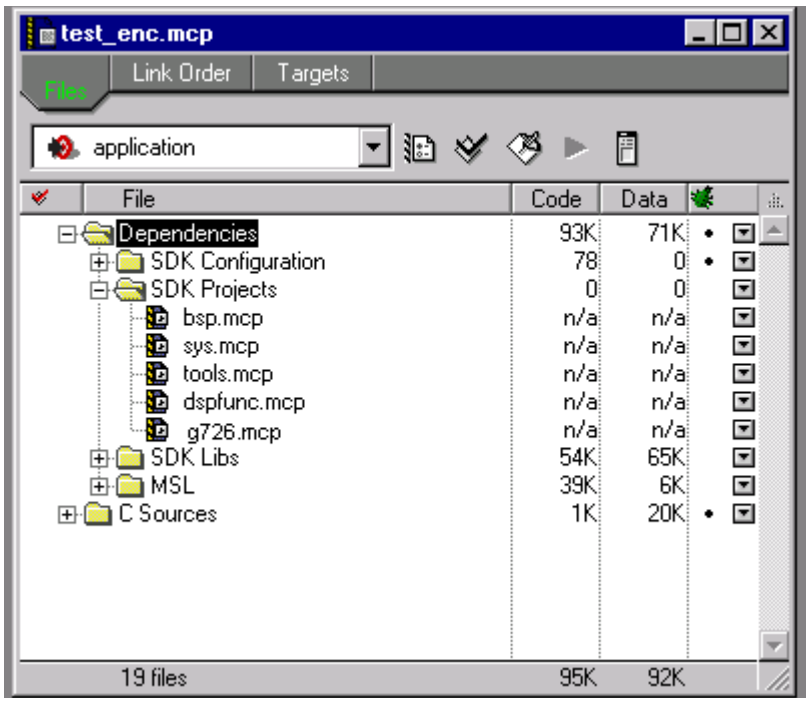

Figure 4-1. Dependency Build for G.726 Library

4.1.2 Direct Build

Direct build allows you to build the G.726 Codec library independently of any other build. To do this:

Step 1. Open the ***g726.mcp*** project, shown in **Figure 4-2**:

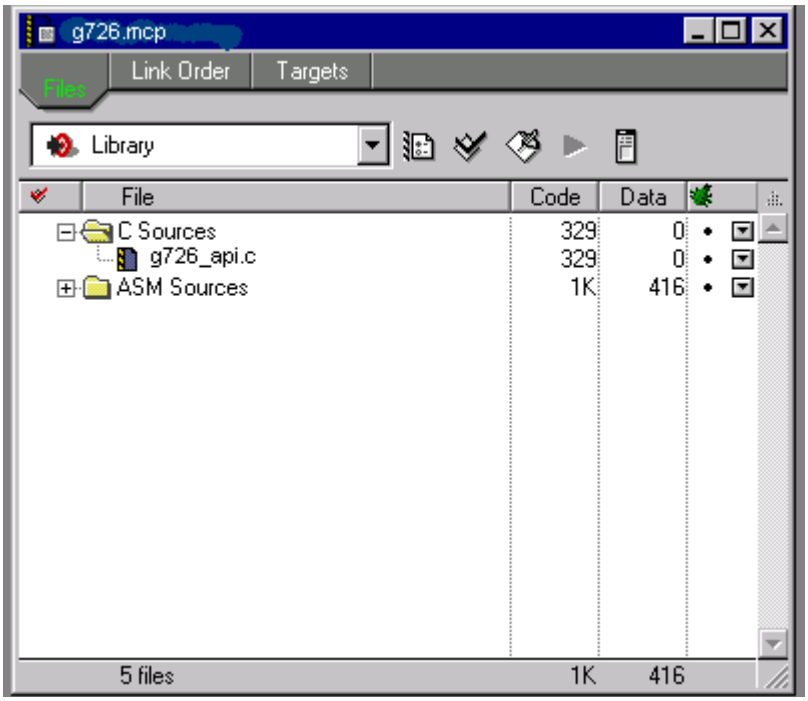

Figure 4-2. G.726 Project

Step 2. Execute the build by pressing function key [F7] or by choosing the *Make* command from the Project menu; see **Figure 4-3**.

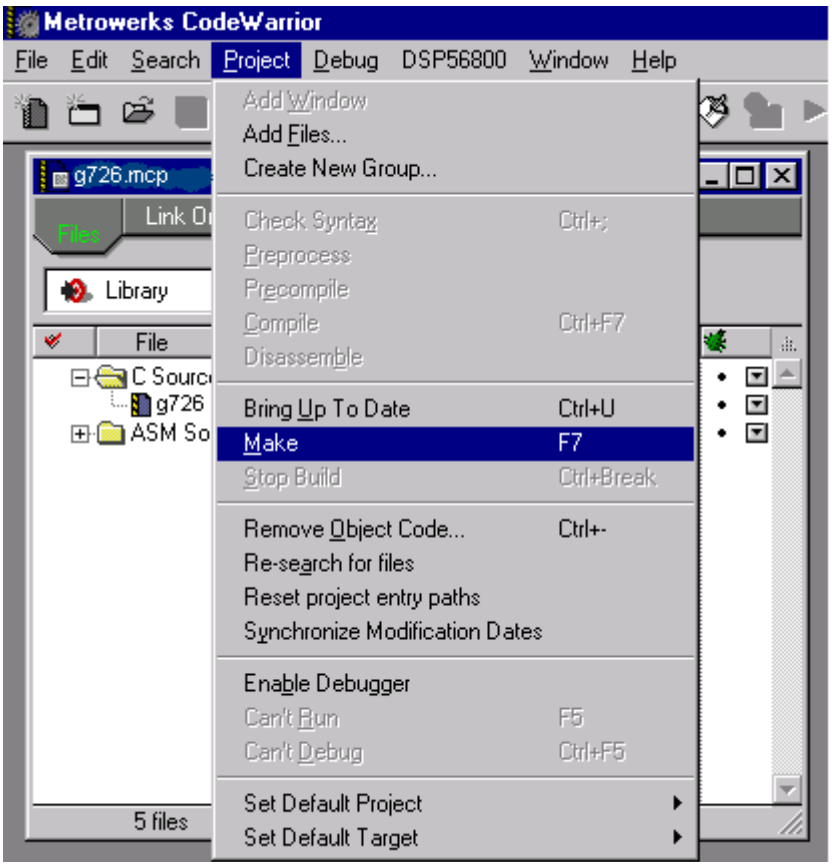

Figure 4-3. Execute *Make*

At this point, if the build is successful, a *g726.lib* library file is created in the ***...\nos\telephony\g726\Debug*** directory.

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

Chapter 5
Linking Applications with the G.726 Codec Library

5.1 G.726 Codec Library

The G.726 library consists of Applications Program Interfaces, (APIs), which provide interface between the user application and the G.726 Codec modules. To invoke G.726 Codec, APIs must be called in this order:

For Encoder:

- G726EncCreate (.....) /* To Create the instance of G726 Encoder*/
- G726EncInit (.....) /* Called within the G726EncControl function */
- G726Encoder(.....); /* can be called as per the no. of samples */
- G726EncControl (.....); /* can be called as per the requirement */
- G726EncDestroy (.....) /* To Destroy the instance created by G726EncCreate */

For Decoder:

- G726DecCreate (.....) /* To Create the instance of G726 Decoder */
- G726DecInit (.....) /* Called within the G726DecControl function */
- G726Decoder(.....); /* can be called as per the no. of samples */
- G726DecControl (.....); /* can be called as per the requirement */
- G726DecDestroy (.....) /* To Destroy the instance created by G726DecCreate */

The G.726 ADPCM library contains a section called **G726_INTERNAL_ROM**, which must be placed in the internal memory.

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 DSP56824EVM External RAM
# using both internal and external data memory (EX = 0)
# and using external program memory (Mode = 3)

MEMORY {

       .pram   (RWX) : ORIGIN = 0x0000, LENGTH = 0xFF80  # ? external program memory
       .avail  (RW)  : ORIGIN = 0x0000, LENGTH = 0x0030  # available
       .cwregs (RW)  : ORIGIN = 0x0030, LENGTH = 0x0010  # C temp registrs in
                                 CodeWarrior
       .im1    (RW)  : ORIGIN = 0x0040, LENGTH = 0x07C0  # data 1
       .rom    (R)   : ORIGIN = 0x0800, LENGTH = 0x0800  # internal data ROM
       .im2    (RW)  : ORIGIN = 0x1000, LENGTH = 0x0600  # data 2
       .hole   (R)   : ORIGIN = 0x1600, LENGTH = 0x0A00  # hole
       .data   (RW)  : ORIGIN = 0x2000, LENGTH = 0xC000  # data segment
       .em     (RW)  : ORIGIN = 0xE000, LENGTH = 0x1000  # data 3
       .stack  (RW)  : ORIGIN = 0xF000, LENGTH = 0x0F80  # stack
       .onchip1(RW)  : ORIGIN = 0xFF80, LENGTH = 0x0040  # on-chip peripheral
                                 registers
       .onchip2(RW)  : ORIGIN = 0xFFC0, LENGTH = 0x0040  # on-chip peripheral
                                 registers
}

FORCE_ACTIVE {FconfigInterruptVector}

SECTIONS {
              #
              # Data (X) Memory Layout
              #
              _EX_BIT     = 0;

              # Internal Memory Partitions (for mem.h partitions)
              _NUM_IM_PARTITIONS = 1;  # .im1 and .im2

              # External Memory Partition (for mem.h partitions)
              _NUM_EM_PARTITIONS = 1;   # .em

              .main_application_code :

              # .text sections

              #  config.c MUST be placed first, otherwise the Interrupt Vector
              #  configInterruptVector will not be located at the correct address,
                                 P:0x0000

              config.c (.text)
              * (.text)
              * (rtlib.text)
              * (fp_engine.text)
              * (user.text)
       } > .pram


       .main_application_data :
       {
```

```
                #
                # Define variables for C initialization code
                #
                F_Xdata_start_addr_in_ROM = ADDR(.rom) + SIZEOF(.rom) / 2;
                F_StackAddr               = ADDR(.stack);
                F_StackEndAddr            = ADDR(.stack) + SIZEOF(.stack) / 2  - 1;

                F_Xdata_start_addr_in_RAM = .;

                #
                # Memory layout data for SDK INCLUDE_MEMORY (mem.h) support
                #

                FmemEXbit = .;
                      WRITEH(_EX_BIT);
                FmemNumIMpartitions = .;
                      WRITEH(_NUM_IM_PARTITIONS);
                FmemNumEMpartitions = .;
                      WRITEH(_NUM_EM_PARTITIONS);
                FmemIMpartitionList = .;
                #     WRITEH(ADDR(.im1));
                #     WRITEH(SIZEOF(.im1) / 2);
                      WRITEH(ADDR(.im2));
                      WRITEH(SIZEOF(.im2) / 2);
                FmemEMpartitionList = .;
                      WRITEH(ADDR(.em));
                      WRITEH(SIZEOF(.em) /2);

                # .data sections

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

                F_Xdata_ROMtoRAM_length = 0;

                F_bss_start_addr = .;
                _BSS_ADDR = .;

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

                F_bss_length = . - _BSS_ADDR;  # Copy DATA

        } > .data

        .g726_internalROM_data :
        {
                * (G726_INTERNAL_ROM.data)
                * (G726_INTERNAL_ROM.bss)

        } > .im1

        FArchIO   = ADDR(.onchip2);
}
```

```
/* For Decoder */

# Linker.cmd file for DSP56824EVM External RAM
# using both internal and external data memory (EX = 0)
# and using external program memory (Mode = 3)

MEMORY {
      .pram   (RWX) : ORIGIN = 0x0000, LENGTH = 0xFF80  # ? external program memory
      .avail  (RW)  : ORIGIN = 0x0000, LENGTH = 0x0030  # available
      .cwregs (RW)  : ORIGIN = 0x0030, LENGTH = 0x0010  # C temp registrs in
                                 CodeWarrior
      .im1    (RW)  : ORIGIN = 0x0040, LENGTH = 0x07C0  # data 1
      .rom    (R)   : ORIGIN = 0x0800, LENGTH = 0x0800  # internal data ROM
      .im2    (RW)  : ORIGIN = 0x1000, LENGTH = 0x0600  # data 2
      .hole   (R)   : ORIGIN = 0x1600, LENGTH = 0x0A00  # hole
      .data   (RW)  : ORIGIN = 0x2000, LENGTH = 0xC000  # data segment
      .em     (RW)  : ORIGIN = 0xE000, LENGTH = 0x1000  # data 3
      .stack  (RW)  : ORIGIN = 0xF000, LENGTH = 0x0F80  # stack
      .onchip1(RW)  : ORIGIN = 0xFF80, LENGTH = 0x0040  # on-chip peripheral
                                 registers
      .onchip2(RW)  : ORIGIN = 0xFFC0, LENGTH = 0x0040  # on-chip peripheral
                                 registers
}

FORCE_ACTIVE {FconfigInterruptVector}

SECTIONS {
            #
            # Data (X) Memory Layout
            #
            _EX_BIT     = 0;

            # Internal Memory Partitions (for mem.h partitions)
            _NUM_IM_PARTITIONS = 1;  # .im1 and .im2

            # External Memory Partition (for mem.h partitions)
            _NUM_EM_PARTITIONS = 1;   # .em


            .main_application_code :
            {
                  # .text sections

                  #  config.c MUST be placed first, otherwise the Interrupt Vector
                  #  configInterruptVector will not be located at the correct
                  address, P:0x0000

                  config.c (.text)
                  * (.text)
                  * (rtlib.text)
                  * (fp_engine.text)
                  * (user.text)
            } > .pram

      .main_application_data :
      {
```

```
                #
                # Define variables for C initialization code
                #
                F_Xdata_start_addr_in_ROM = ADDR(.rom) + SIZEOF(.rom) / 2;
                F_StackAddr               = ADDR(.stack);
                F_StackEndAddr            = ADDR(.stack) + SIZEOF(.stack) / 2  - 1;

                F_Xdata_start_addr_in_RAM = .;

                #
                # Memory layout data for SDK INCLUDE_MEMORY (mem.h) support
                #

                FmemEXbit = .;
                      WRITEH(_EX_BIT);
                FmemNumIMpartitions = .;
                      WRITEH(_NUM_IM_PARTITIONS);
                FmemNumEMpartitions = .;
                      WRITEH(_NUM_EM_PARTITIONS);
                FmemIMpartitionList = .;
                #     WRITEH(ADDR(.im1));
                #     WRITEH(SIZEOF(.im1) / 2);
                      WRITEH(ADDR(.im2));
                      WRITEH(SIZEOF(.im2) / 2);
                FmemEMpartitionList = .;
                      WRITEH(ADDR(.em));
                      WRITEH(SIZEOF(.em) /2);

                # .data sections

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

                F_Xdata_ROMtoRAM_length = 0;

                F_bss_start_addr = .;
                _BSS_ADDR = .;

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

                F_bss_length = . - _BSS_ADDR;  # Copy DATA

        } > .data

        .g726_internalROM_data :
        {
                * (G726_INTERNAL_ROM.data)
                * (G726_INTERNAL_ROM.bss)
        } > .im1

        FArchIO   = ADDR(.onchip2);
}
```

Chapter 6
G.726 Applications

6.1 Test and Demo Applications

To verify the G.726 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.

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

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

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/