MICROCHIP

dsPIC® DSC G.726A Speech Encoding/Decoding Library User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2009 ==

Trademarks

Printed on recycled paper.

ISBN: 978-1-61341-296-1

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Table of Contents

Preface **5**

Chapter 1. Introduction

1.1 Library Overview 11
1.2 Features 13
1.3 System Requirements 14

Chapter 2. Installation

2.1 Installation Procedure 15
2.2 Library Files 15

Chapter 3. Quick Start Demonstration

3.1 Quick Start Demonstration for dsPIC33F Device Family 21
3.2 Quick Start Demonstration for dsPIC33E Device Family 29

Chapter 4. Application Programming Interface

4.1 Adding the Library to the Application 37
4.2 Using the Library 38
4.3 Resource Requirements 39
4.4 Library Interface 39
4.5 Application Tips 48

Chapter 5. PC Encoding Utility

5.1 Microphone 49
5.2 WAVE File 52
5.3 Output Files 54

Index **57**

Worldwide Sales and Service **58**

NOTES:

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This preface contains general information that will be useful to know before using the dsPIC® DSC G.726A Speech Encoding/Decoding Library, also referred to as the G.726A library. Items discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This document describes how to use the G.726A library as a development tool to emulate and debug firmware on a target board. This user's guide is composed of the following chapters:

- **Chapter 1. "Introduction"** – This chapter provides a brief overview of the G.726A library.
- **Chapter 2. "Installation"** – This chapter provides detailed information needed to install the G.726A library on a PC.
- **Chapter 3. "Quick Start Demonstration"** – This chapter describes the demonstrations available for the G.726A library.
- **Chapter 4. "Application Programming Interface"** – This chapter outlines how the API functions provided in the G.726A library can be included in your application software through the API.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User’s Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	“Save project before build”
Underlined, italic text with right angle bracket	A menu path	<u>*File>Save*</u>
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
`Plain Courier New`	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`C:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants (in source code)	`0xFF, ‘A’`
`Italic Courier New`	A variable argument	*`file.o`*, where `file` can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options] file [options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`

	Represents code supplied by user	`void main (void)`
		`{ ...`
		`}`

WARRANTY REGISTRATION

Please complete the enclosed Warranty Registration Card and mail it promptly. Sending in the Warranty Registration Card entitles users to receive new product updates. Interim software releases are available at the Microchip web site.

RECOMMENDED READING

This user's guide describes how to use the G.726A library. The following Microchip documents are available from the Microchip web site (www.microchip.com), and are recommended as supplemental reference resources.

dsPIC30F Family Reference Manual (DS70046)

Refer to this document for detailed information on dsPIC30F device operation. This reference manual explains the operation of the dsPIC30F Digital Signal Controller (DSC) family architecture and peripheral modules but does not cover the specifics of each device. Refer to the appropriate device data sheet for device-specific information.

dsPIC33F/PIC24H Family Reference Manual Sections

Refer to these documents for detailed information on dsPIC33F/PIC24H device operation. These reference manual sections explain the operation of the dsPIC33F/PIC24H DSC family architecture and peripheral modules, but do not cover the specifics of each device. Refer to the specific device data sheet for device-specific information.

dsPIC33E/PIC24E Family Reference Manual Sections

Refer to these documents for detailed information on dsPIC33E/PIC24E device operation. These reference manual sections explain the operation of the dsPIC33E/PIC24E DSC family architecture and peripheral modules, but do not cover the specifics of each device. Refer to the specific device data sheet for device-specific information.

16-Bit MCU and DSC Programmer's Reference Manual (DS70157)

This manual is a software developer's reference for the 16-bit PIC24F and PIC24H MCU and 16-bit dsPIC30F and dsPIC33F DSC device families. It describes the instruction set in detail and also provides general information to assist in developing software for these device families.

MPLAB® Assembler, Linker and Utilities for PIC24 MCUs and dsPIC® DSCs User's Guide (DS51317)

MPLAB Assembler for PIC24 MCUs and dsPIC® DSCs (formerly MPLAB ASM30) produces relocatable machine code from symbolic assembly language for the dsPIC DSC and PIC24 MCU device families. The assembler is a Windows console application that provides a platform for developing assembly language code. The assembler is a port of the GNU assembler from the Free Software Foundation (www.fsf.org).

MPLAB® C Compiler for PIC24 MCUs and dsPIC® DSCs User's Guide (DS51284)

This document describes the features of the optimizing C compiler, including how it works with the assembler and linker. The assembler and linker are discussed in detail in the *"MPLAB® Assembler, Linker and Utilities for PIC24 MCUs and dsPIC® DSCs User's Guide"* (DS51317).

Readme Files

For the latest information on using other tools, read the tool-specific Readme files in the `Readme` subdirectory of the MPLAB IDE installation directory. The Readme files contain updated information and known issues that may not be included in this user's guide.

THE MICROCHIP WEB SITE

Microchip provides online support through our web site at: http://www.microchip.com. This web site makes files and information easily available to customers. Accessible by most Internet browsers, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listings
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listings of seminars and events; and listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com, click **Customer Change Notification** and follow the registration instructions.

- **Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB® C compiler; MPASM™ and MPLAB 16-bit assemblers; MPLINK™ and MPLAB 16-bit object linkers; and MPLIB™ and MPLAB 16-bit object librarians.
- **Emulators** – The latest information on the Microchip MPLAB REAL ICE™ In-Circuit Emulator.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 3.
- **MPLAB IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 device programmer and the PICkit™ 3 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

DOCUMENT REVISION HISTORY

Revision A (June 2011)

This is the initial release of the dsPIC® DSC G.726A Speech Encoding/Decoding Library User's Guide.

NOTES:

Chapter 1. Introduction

This chapter introduces the dsPIC DSC G.726A Speech Encoding/Decoding Library, also referred to as the G.726A library. This library provides functionality to compress a speech signal, which is useful in applications that have limited memory or communication resources. This library supports the dsPIC33F and dsPIC33E device families.

This user's guide provides information that you can use to incorporate the G.726A library into your embedded solution.

The following topics are covered in this chapter:

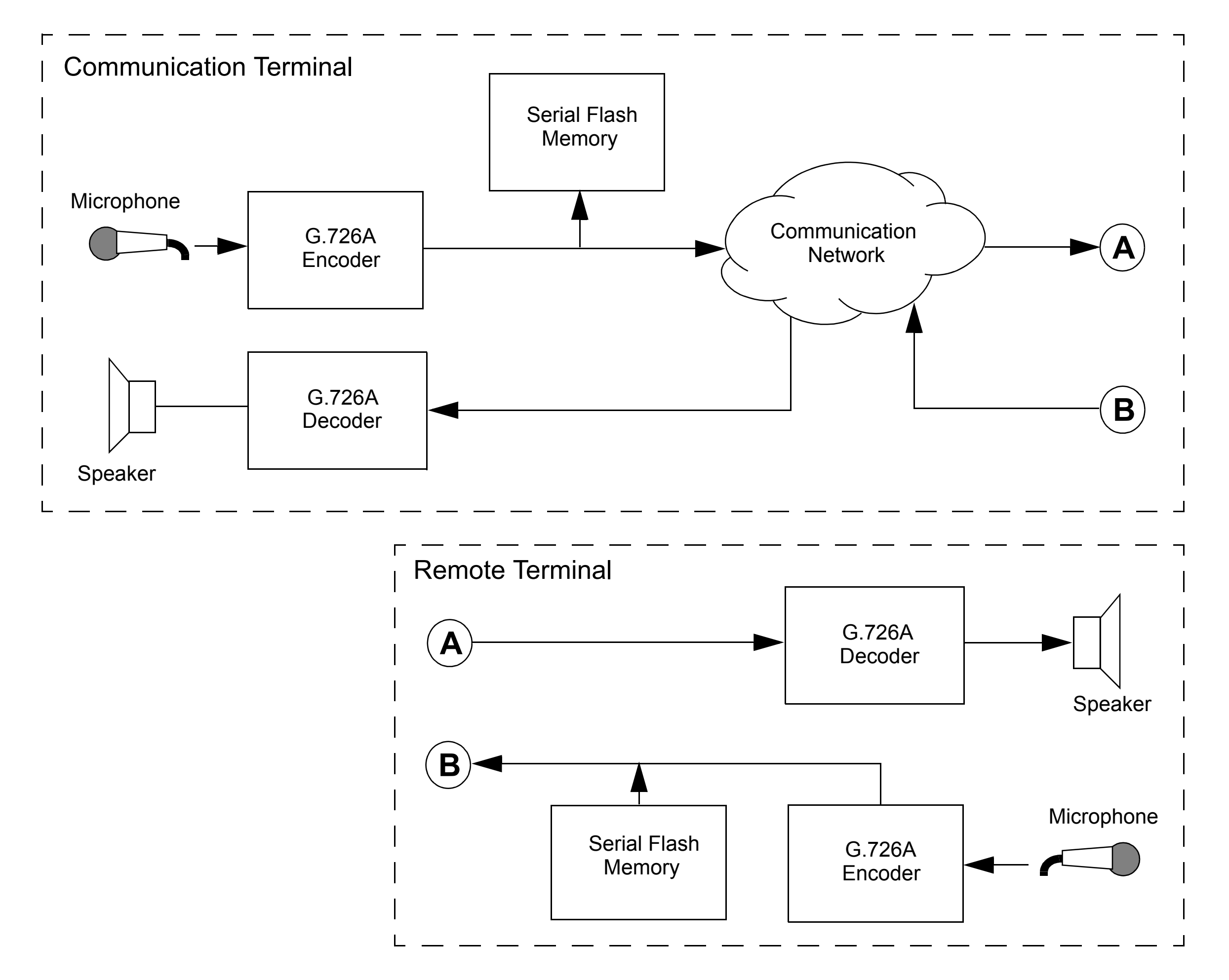
- Library Overview
- Features
- System Requirements

1.1 LIBRARY OVERVIEW

The G.726A library reduces the number of bytes required to represent a speech frame. This reduction, or compression of speech data, is specified by the compression ratio. Figure 1-1 shows a typical application of the G.726A library.

The communication terminal shown in Figure 1-1 uses the G.726A Speech Encoder to perform speech compression. The communication terminal can then store the compressed data in Flash program memory and transmit the compressed data over a communication link to a remote terminal. The communication terminal decompresses the compressed speech data that is received from the remote terminal by using the G.726A Speech Decoder, and outputs the decompressed speech signal to a local speaker. By using the G.726A library in such an application, the amount of memory required to store speech data and the communication bandwidth requirement is reduced significantly.

FIGURE 1-1: APPLICATION EXAMPLE



The G.726A library operates at a sampling rate of 8 kHz, and is suitable for the following applications:

- Voice recording and playback
- Voice over Internet Protocol (VoIP)
- Communication
- Automated announcement systems
- Intercom
- Walkie-talkie
- Any application using message playback

The G.726A library is based on the ITU-T G.726A recommendation which uses the Adaptive Differential Pulse Code Modulation (ADPCM) technique. The library is appropriate for both half-duplex and full-duplex systems. The library is written in the C language and the dsPIC DSC Assembly language.

The G.726A library operates at a sampling rate of 8 kHz and supports four output data rates. The library package contains five archives for each device family. The `libG26A_33F.a` archive file supports all four of the bit rates for the dsPIC33F devices, and the `libG726A_33E.a` archive file supports all four of the bit rates for the dsPIC33E devices. For each of the two device families, the library contains four other archive files, each optimized to support one of the available bit rates. The optimized archives can be used in applications which require only one bit rate to be supported while looking to reduce the application's Flash program memory usage.

The library Application Programming Interface (API) provides a set of initialization, encode and decode functions. The application can use the encoder and decoder independent of each other, or use them together. It is also possible for the encoder and decoder to share scratch memories. Multiple instances of the encoder and decoder can also be instantiated.

The G.726A library also contains a PC-based Encoder Utility (PCEU), which allows speech to be recorded on a PC (through the PC's microphone port), and converted to G.726A encoded files. These files can then be included into the user application as a part of the application code, thereby allowing encoded speech segments to be stored in the device Flash program memory. Additionally, the encoded data can also be stored in data EEPROM (if the target dsPIC DSC contains Data EEPROM) and RAM. The PCEU can also encode prerecorded `.wav` files and can accept multiple files as an input.

1.2 FEATURES

The G.726A library has the following features:

- Simple user interface – only one library file and one header file
- All functions are called from a C application program
- Full compliance with the Microchip C30 Compiler, Assembler and Linker
- Highly optimized assembly code that uses DSP instructions and advanced addressing modes
- Compact and concise API for easier integration with application
- Four bit rates:
  - 16 kbps at 8 kHz sampling rate – 7:1 compression ratio
  - 24 kbps at 8 kHz sampling rate – 4.67:1 compression ratio
  - 32 kbps at 8 kHz sampling rate – 3.5:1 compression ratio
  - 40 kbps at 8 kHz sampling rate – 2.8:1 compression ratio
- Library package contains archives optimized for single bit rates
- G.726A PCEU for recording and encoding speech through a PC
- Audio bandwidth: 0 kHz to 4 kHz at a sampling rate of 8 kHz
- Can be integrated with Microchip's Noise Suppression, Acoustic Echo Cancellation and Line Echo Cancellation Libraries
- Library includes:
  - Sample demonstration applications with complete source code
  - User's guide
  - HTML help files for PCEU
  - Windows Help File

1.3 SYSTEM REQUIREMENTS

The dsPIC DSC G.726A Speech Encoding/Decoding Library package installation requires a PC-compatible system with these attributes:

- 1 GHz or higher processor
- HTML browser
- 16 MB RAM (minimum)
- 40 MB available hard drive space
- Microsoft® Windows® 98, Windows 2000, Windows NT, Windows XP, or Windows 2007
- Sound card

Chapter 2. Installation

This chapter describes the various files in the G.726A library, and includes instructions for installing the library on your laptop or PC for use with dsPIC DSC programming tools.

The following topics are covered in this chapter:

- Installation Procedure
- Library Files

2.1 INSTALLATION PROCEDURE

The G.726A library is available as a zip archive file, if downloaded from the Microchip web site, or as a CD. Copy the archive file to your local hard drive, and then extract the files.

To install the library, follow these steps:

1. Double-click `G726A setup.exe`. The License Agreement screen appears.
2. Review the License Agreement and click **I Agree** to continue. The Installation Destination dialog appears.
3. Specify the location (that is, a directory) where the library should be installed, and then click **Install**.
4. Click **Close** to close the dialog. This completes the dsPIC DSC G.726A Speech Encoding/Decoding Library installation.

   The installation creates the folder, `G.726A v3.0`, which contains the files described in **2.2 "Library Files"**.

2.2 LIBRARY FILES

The dsPIC DSC G.726A Speech Encoding/Decoding Library CD or zip archive file creates a directory, `G.726A v3.0`. The directory contains these folders:

- `demos` Folder
- `docs` Folder
- `h` Folder
- `libs` Folder
- `PCEU` Folder
- Library Source Code for dsPIC33F
- Library Source Code for dsPIC33E

2.2.1 demos Folder

The demos folder contains application code examples, which demonstrate the use of G.726A library in different application scenarios.

The demos folder contains the following sub-folders:

- Communication
- Playback
- RecordPlay

2.2.1.1 Communication FOLDER

The Communication folder contains files, which demonstrate the use of the G.726A library in a communication setup that consists of two development boards communicating speech data over a serial link. Table 2-1 describes the files in this folder.

TABLE 2-1: Communication FOLDER FILES

File Name	Description
dsPIC33E G726A Communication Demo 1.hex	dsPIC33E demonstration hexadecimal file for board 1.
dsPIC33E G726A Communication Demo 2.hex	dsPIC33E demonstration hexadecimal file for board 2.
dsPIC33E G726A Communication Demo.mcp	dsPIC33E demonstration MPLAB Project file.
dsPIC33F G726A Communication Demo 1.hex	dsPIC33F demonstration hexadecimal file for board 1.
dsPIC33F G726A Communication Demo 2.hex	dsPIC33F demonstration hexadecimal file for board 2.
dsPIC33F G726A Communication Demo.mcp	dsPIC33F demonstration MPLAB Project file.
cleanup.bat	A batch file script for cleaning up intermediate build files.
h\Explorer16.h	C header file for Explorer 16 Development Board routines.
h\MEB.h	C header file for Multimedia Expansion Board (MEB) routines.
h\G726A.h	C header file defining the interface to the G.726A Library.
h\UART2Drv.h	C header file defining the interface to the UART driver.
h\WM8510CodecDrv.h	C header file defining the interface to the WM8510 codec driver.
h\WM8731CodecDrv.h	C header file defining the interface to the WM8731 codec driver.
lib\libG726A_33E.a	Library archive file for dsPIC33E.
lib\libG726A_33F.a	Library archive file for dsPIC33F.
src\Explorer16.c	C source file containing routines for the Explorer 16 Development Board.
src\MEB.c	C source file containing routines for the MEB.
src\main.c	C source file containing the main speech processing routine.
src\UART2Drv.c	C source file containing the driver routines for UART2.
src\WM8731.c	C source file containing the driver routines for WM8731 codec.
src\WM8510CodecDrv.c	C source file containing the driver routines for WM8510 codec.

2.2.1.2 Playback FOLDER

The Playback folder contains demonstration files that can be used to show the use of G.726A library, to implement a playback-only system. The demonstration will playback encoded speech data stored in Flash program memory. Table 2-2 describes the files in this folder.

TABLE 2-2: Playback FOLDER FILES

File Name	Description
dsPIC33E G726A Playback Demo.hex	dsPIC33E demonstration hexadecimal file.
dsPIC33F G726A Playback Demo.hex	dsPIC33F demonstration hexadecimal file.
dsPIC33E G726A Playback Demo.mcp	dsPIC33E MPLAB Project file.
dsPIC33F G726A Playback Demo.mcp	dsPIC33F MPLAB Project file.
cleanup.bat	A batch file script for cleaning up intermediate build files.
h\Explorer16.h	C header file for Explorer 16 Development Board routines.
h\MEB.h	C header file for MEB routines.
h\G726A.h	C header file defining the interface to the G.726A Library.
h\PgmMemory.h	C header file defining the interface to routines to read Flash program memory.
h\WM8731CodecDrv.h	C header file defining the interface to the WM8731 codec driver.
h\WM8510CodecDrv.h	C header file defining the interface to the WM8510 codec driver.
lib\libG726A_33E.a	Archive file for dsPIC33E.
lib\libG726A_33F.a	Archive file for dsPIC33F.
src\Explorer16.c	C source file containing routines for the Explorer 16 Development Board.
src\MEB.c	C source file containing routines for the MEB.
src\main.c	C source file containing the main speech processing routine.
src\SpeechSegment.s	Encoded Speech Segment file.
src\PgmMemory.s	Assembly source code containing the routines to read from Flash program memory.
src\WM8731CodecDrv.c	C source file containing the driver routines for the WM8731 codec.
src\WM8510CodecDrv.c	C source file containing the driver routines for the WM8510 codec.

2.2.1.3 RecordPlay FOLDER

The RecordPlay folder contains the demonstration files that show the use of the G.726A library to implement a voice recorder system. The demonstration will encode speech data, store it in external serial Flash memory, and then playback this data when requested. Table 2-3 describes the files in this folder.

TABLE 2-3: RecordPlay FOLDER FILES

File Name	Description
dsPIC33E G726A RecordPlay Demo.hex	dsPIC33E demonstration hexadecimal file.
dsPIC33F G726A RecordPlay Demo.hex	dsPIC33F demonstration hexadecimal file.
dsPIC33E G726A RecordPlay Demo.mcp	dsPIC33E demonstration hexadecimal file.
dsPIC33F G726A RecordPlay Demo.mcp	dsPIC33F demonstration MPLAB Project file.
cleanup.bat	A batch file script for cleaning up intermediate build files.
h\Explorer16.h	C header file for Explorer 16 Development Board routines.
h\MEB.h	C header file for MEB.
h\G726a.h	C header file defining the interface to the G.726A Library.
h\SST25VF040BDrv.h	C header file defining the interface to routines for SST25VF040B serial Flash memory.
h\SST25VF016BDrv.h	C header file defining the interface to routines for SST25VF016B serial Flash memory.
h\WM8510CodecDrv.h	C header file defining the interface to the WM8510 codec driver.
h\WM8731CodecDrv.h	C header file defining the interface to the WM8731 codec driver.
lib\libG726A_33E.a	Library archive file for dsPIC33E.
lib\libG726A_33F.a	Library archive file for dsPIC33F.
src\Explorer16.c	C source file containing routines for the Explorer 16 Development Board.
src\MEB.c	C source file containing routines for the MEB.
src\main.c	C source file containing the main speech processing routine.
src\SpeechSegment.s	Encoded Speech Segment file.
src\SST25VF040BDrv.c	C source file containing the routines for SST25VF040B serial Flash memory.
src\SST25VF016BDrv.c	C source file containing the routines for SST25VF016B serial Flash memory.
src\WM8510CodecDrv.c	C source file containing the driver routines for the WM8510 codec.
src\WM8731CodecDrv.c	C source file containing the driver routines for the WM8731 codec.

2.2.2 docs Folder

The docs folder contains the user's guide for the G.726A library. To view the document, double-click the file name. The user's guide can also be downloaded from the Microchip web site (www.microchip.com).

2.2.3 h Folder

The h folder contains the C header files, which define the Application Programming Interface (API). Table 2-4 describes the files in this folder.

TABLE 2-4: INCLUDE FILES

File Name	Description
G726A.h	Include file that contains the interface to the dsPIC DSC G.726A Speech Encoding/Decoding Library. This file must be included in the application to use any of the four modes.

2.2.4 libs Folder

The libs folder contains archive files for the G.726A library. Table 2-5 describes the files in this folder.

TABLE 2-5: LIBRARY ARCHIVE FILES

File Name	Description
libG726A_33F.a	Archive file that contains the dsPIC33F library functions. This file must be included in the application.
libG726A_33E.a	Archive file that contains the dsPIC33E library functions. This file must be included in the application.

2.2.5 PCEU Folder

The PCEU folder contains files required by the G.726A PC Encoding Utility (PCEU). Table 2-6 describes the files in this folder.

TABLE 2-6: PCEU FILES

File Name	Description
dsPICSpeechRecord.exe	PCEU executable file. Double-click this file to start the Speex PCEU utility.
SpeechRecord_G711.dll	DLL files required by the PCEU.
SpeechRecord_G726.dll	

2.2.6 Library Source Code for dsPIC33F

This folder contains the library source code used to generate the dsPIC33F archive.

2.2.7 Library Source Code for dsPIC33E

This folder contains the library source code used to generate the dsPIC33E archive.

NOTES:

Chapter 3. Quick Start Demonstration

This chapter describes the quick start demonstration code examples for the dsPIC33F and dsPIC33E device families, which are part of the G.726A library package.

3.1 QUICK START DEMONSTRATION FOR dsPIC33F DEVICE FAMILY

The following demonstration examples are covered in this section:

- Communication Demonstration
- Playback Demonstration
- Record Play Demonstration

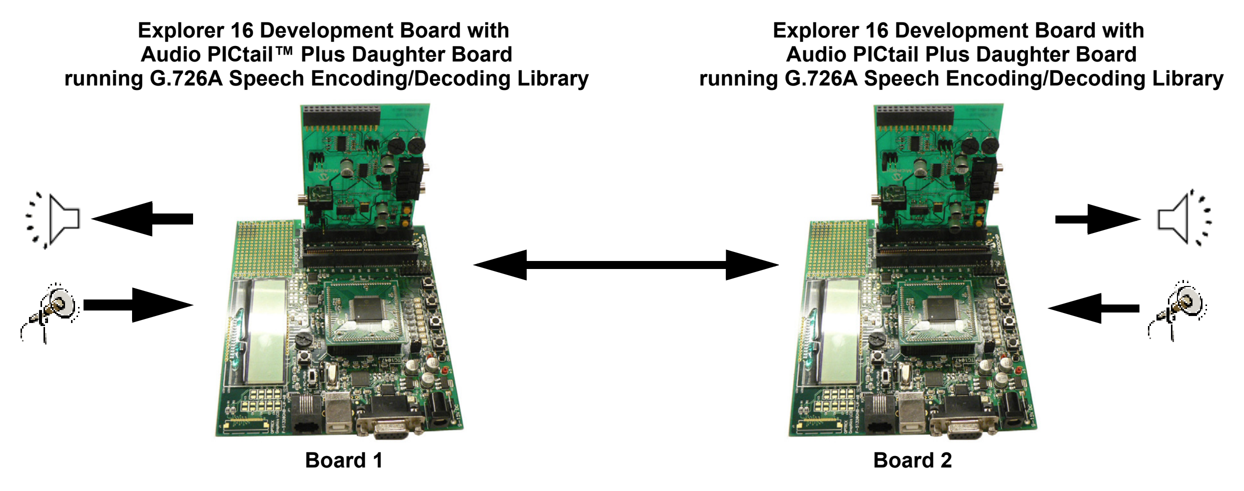
3.1.1 Communication Demonstration

The Communication demonstration shows how the G.726A library can be used to reduce the required bandwidth in a full-duplex communication type of application running on a dsPIC33F device.

3.1.1.1 DEMONSTRATION SUMMARY

The Communication code example requires the use of two Explorer 16 Development Boards with two Audio PICtail™ Plus Daughter Boards (not included with the software license), which are set up as shown in Figure 3-1.

FIGURE 3-1: G.726A COMMUNICATION DEMONSTRATION SETUP


Explorer 16 Development Board with
Audio PICtail™ Plus Daughter Board
running G.726A Speech Encoding/Decoding Library
Explorer 16 Development Board with
Audio PICtail Plus Daughter Board
running G.726A Speech Encoding/Decoding Library
Board 1
Board 2

Headsets are connected to the Audio PICtail Plus Daughter Boards. When a user speaks into the headset connected to board 1, the WM8510 codec on the Audio PICtail Plus Daughter Board will sample and convert the data, and provide it to the dsPIC DSC device through the DCI module. The dsPIC DSC device will compress the speech signal using the G.726A encoder, and then transmit the compressed speech frame data through the UART2 module and RS-232 transceiver to board 2.

The dsPIC DSC device on board 2 receives the encoded frame through the on-board RS-232 transceiver and the device's UART2 module. The dsPIC DSC device decodes the received frame using the G.726A decoder, and then plays out the signal on the headset through its DCI module and the on-board WM8510 codec.

For a speech signal that is sampled at 8 kHz at 16-bit resolution, the resulting data rate is 128 kbps. To communicate this data to a remote terminal over a communication link, the minimum required communication link bandwidth would be 128 kbps. By encoding the speech signal using 16 kbps mode, the minimum communication link bandwidth would be 16 kbps. This results in a bandwidth reduction of 8x. The code example invokes the `G726AEncode()` and `G726ADecode()` functions from the dsPIC DSC G.726A Speech Encoding/Decoding Library to encode and decode speech data.

3.1.1.2 DEMONSTRATION SETUP

The demonstration application is intended to run on the Explorer 16 Development Board with an Audio PICtail Plus Daughter Board (not included with the software license).

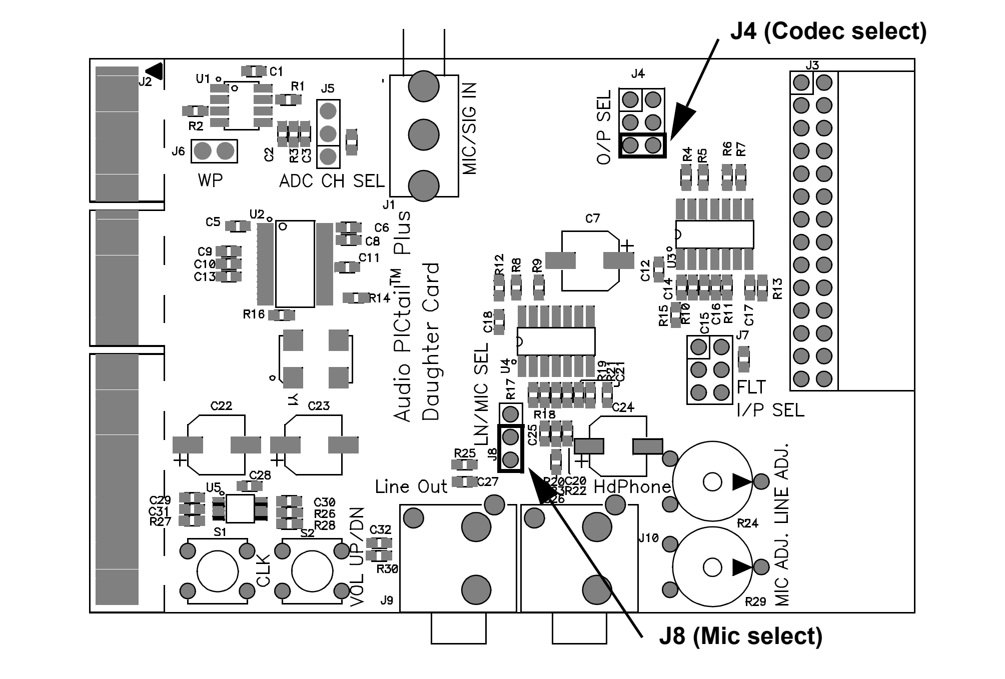
Use the procedure in the following section to set up the demonstration.

3.1.1.2.1 Configure Explorer 16 Development Board and Audio PICtail Plus Daughter Boards

Before applying power, you need to configure the boards:

1. On the Audio PICtail Plus Daughter Board 1, set jumper J4 (O/P SEL) to the codec position and set jumper J8 (LN/MIC) to the microphone position, as shown in Figure 3-2.

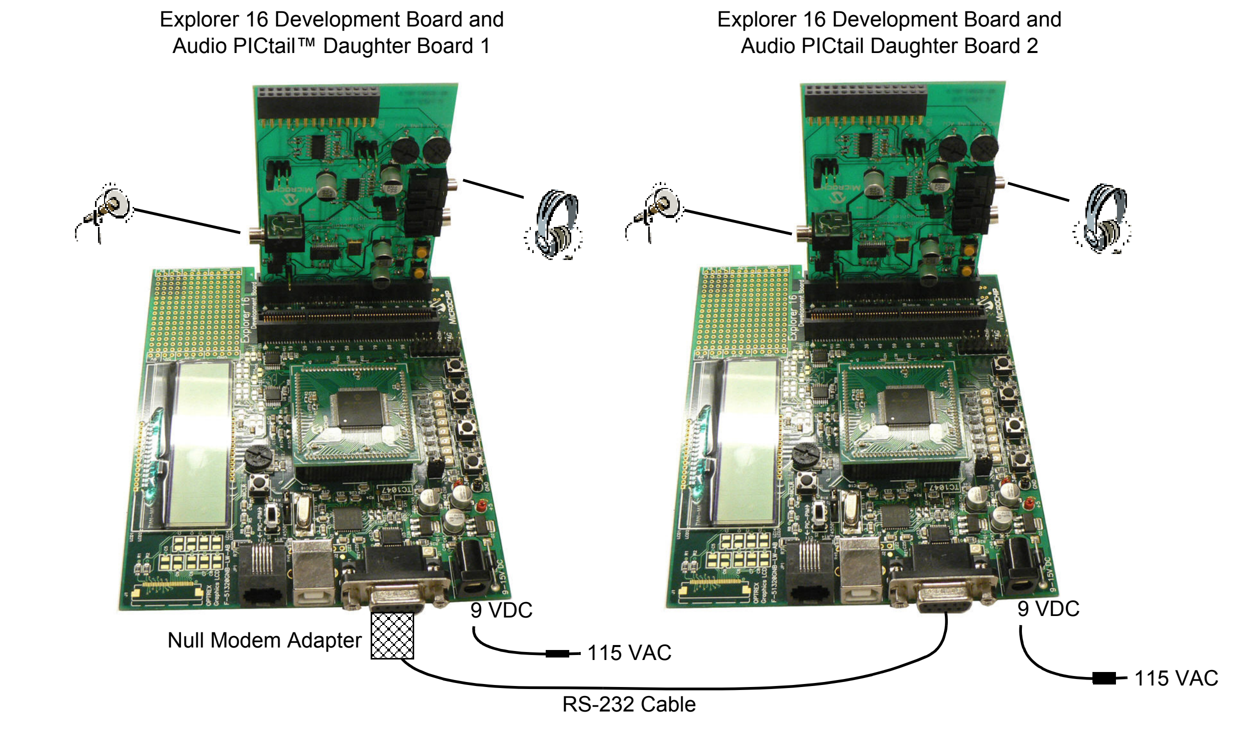
FIGURE 3-2: JUMPER POSITIONS ON AUDIO PICtail™ PLUS DAUGHTER BOARD



2. Insert the Audio PICtail Plus Daughter Board 1 into the Explorer 16 Development Board 1 PICtail Plus socket J5.
3. Connect an output device, such as headphones, to socket J10 on the Audio PICtail Plus Daughter Board 1.

4. Connect a microphone to socket J1 on the Audio PICtail Plus Daughter Board 1.
5. Repeat steps 1 though 4 for the second Audio PICtail Plus Daughter Board and Explorer 16 Development Board.
6. Connect one end of the DB9M-DB9M Null Modem Adapter to P1 on Explorer 16 Development Board 1. Then, connect one end of the RS-232 cable to the Null Modem Adapter.
7. Connect the other end of the RS-232 cable to P1 on the Explorer 16 Development Board 2.
8. Once the setup is complete, apply power to the Explorer 16 Development Boards. The setup would be similar to the one shown in Figure 3-3.

FIGURE 3-3: SETUP FOR SPEEX COMMUNICATION DEMONSTRATION


Explorer 16 Development Board and
Audio PICtail™ Daughter Board 1
Explorer 16 Development Board and
Audio PICtail Daughter Board 2
9 VDC
Null Modem Adapter
115 VAC
9 VDC
115 VAC
RS-232 Cable

3.1.1.2.2 Programming the dsPIC DSC Device

Use this process to load the Speex communication demonstration into the dsPIC DSC device on the Explorer 16 Development Board.

1. On your PC, launch MPLAB IDE and open the `dsPIC33F G726A Communication Demo.mcp` project located in the `Communication` folder. For more information on using MPLAB IDE, refer to the *"MPLAB® IDE, Simulator, Editor User's Guide"* (DS51025).
2. Select either *File > Import > dsPIC33F G726A Communication Demo 1.hex* to import the project hexadecimal file for board 1, or select *File > Import > dsPIC33F G726A Communication Demo 2.hex* to import the project hexadecimal file for board 2.
3. Select *Programmer > Connect* to link MPLAB ICD 3 to the dsPIC DSC target device. The Output window confirms that MPLAB ICD 3 is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming has succeeded.

Note: The MPLAB REAL ICE™ can be used instead of the MPLAB ICD 3.

3.1.1.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into both the devices, the application is ready to run.

Use the following procedure to run the demonstration:

1. Press the Reset button on board 1. LED D3 on the board will light up indicating that the board is waiting for synchronization to be complete.
2. Press the Reset button on board 2. Now, LED D4 on both the boards will light up indicating that the boards are synchronized and communicating encoded speech data frames.
3. Speak into the microphone connected to board 1. This speech will be heard in the output device, such as headphones, connected to board 2 and vice-versa.

3.1.1.4 DEMONSTRATION CODE DESCRIPTION

The demonstration code runs on a dsPIC DSC device, using the Primary Oscillator as the clock source with the PLL set for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This main function allocates all of the variables and arrays in data memory that are needed by the UART and WM8510 codec driver, as well as the blocks of data memory that need to be allocated for the G.726A library functions.

The main function calls the `G726AEncoderInit()` and `G726ADecoderInit()` functions from the G.726A library, which initializes the G.726A encoder and decoder algorithms to their default and initial state.

The main function also calls the `WM8510Init()` function to initialize the DCI module, the I^{2}C™ module, the WM8510 codec and the DCI interrupt. The WM8510 codec module acts as a Master, and drives the serial clock and frame synchronization lines. The DCI module acts as a Slave. The DCI module is set for the multi-channel Frame Sync Operating mode, with 16-bit data words and 16 data words or time slots per frame, of which only one transmit slot and one receive slot are used in this demonstration. The dsPIC DSC device will send the control information to the WM8510 codec through the dsPIC DSC device's I^{2}C module. Subsequently, the `WM8510Start()` function is used to enable the DCI module and I^{2}C module. The `WM8510SampleRate8KConfig()` configures the WM8510 codec for a sampling rate of 8 kHz.

The `UART2Init()` function configures the UART2 module on the dsPIC DSC for operation at the baud rate specified in `UART2Drv.h`. The driver provides a blocking interface through the `UART2Read()` and `UART2Write()` functions. The main processing uses the `UART_ioctl()` function with a `UART2_RX_NDATA` command to know when a frame of data has been received and when frame transmission has been completed.

The `BoardSychronize()` function is called to ensure that the two boards are synchronized. The UART driver is checked for a full frame of encoded data. If a frame of data has been received, the `G726ADecode()` function is called to decode this frame. The decoded speech frame is written to the WM8510 codec driver. The codec driver is polled for a frame of raw speech samples. If the frame is ready, the `G726AEncode()` function is called. The encoded frame is written to the UART2 driver for transmission.

The encoded G.726A data is packed using the `G726APack()` function. This reduces the bandwidth needed by the communication channel. Similarly, the received data is unpacked using the `G726AUnpack()` function before it is decoded.

3.1.2 Playback Demonstration

The Playback demonstration shows how the G.726A library can be used in a playback type of application where encoded frames are read from Flash program memory, decoded by the Speex decoder, and then output. By storing the encoded speech frames in Flash program memory, the application can minimize external memory usage.

3.1.2.1 DEMONSTRATION SUMMARY

The Playback code example requires the use of one Explorer 16 Development Board with one Audio PICtail Plus Daughter Board (not included with the software license). A headset is connected to the Audio PICtail Plus Daughter Board. The demonstration code reads G.726A encoded audio data, which are stored in Flash program memory. The encoded audio data is then decoded using the G.726A decoder. The decoded speech data is written to the WM8510 codec for playback on the output device, such as headphones.

The encoded speech data is obtained using the G.726A PCEU. The resulting `.s` file generated by the PCEU is included into the project and compiled. The encoded speech data now becomes a part of Flash program memory. The code example invokes the `G726ADecode()` functions from G.726A library to decode speech data.

3.1.2.2 DEMONSTRATION SETUP

The demonstration application is intended to run on the Explorer 16 Development Board with an Audio PICtail Plus Daughter Board (not included with the software license).

Use the procedure in the following section to set up the demonstration.

3.1.2.2.1 Configure Explorer 16 Development Board and Audio PICtail Plus Daughter Board

Before applying power, you need to configure the boards:

1. On the Audio PICtail Plus Daughter Board, set jumper J4 (O/P SEL) to the codec position and set jumper J8 (LN/MIC) to the microphone position.
2. Insert the Audio PICtail Plus Daughter Board into the Explorer 16 Development Board PICtail Plus socket J5.
3. Connect the output device, such as headphones, to socket J10 on the Audio PICtail Plus Daughter Board.
4. Once the setup is complete, apply power to the Explorer 16 Development Board.

3.1.2.2.2 Programming the dsPIC DSC Device

Use this process to load the Speex Playback demonstration into the dsPIC DSC device on the Explorer 16 Development Board:

1. On your PC, launch MPLAB IDE and open the `dsPIC33F G726A Playback Demo.mcp` project file located in the `Playback` folder. For more information on using MPLAB IDE, refer to the *“MPLAB® IDE, Simulator, Editor User's Guide”* (DS51025).
2. Select *File > Import > dsPIC33F G726A Playback Demo.hex* to import the project hexadecimal file.
3. Select *Programmer > Connect* to link MPLAB ICD 3 to the dsPIC DSC target device. The Output window confirms that MPLAB ICD 3 is ready.

4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming has succeeded.
5. Connect MPLAB ICD 3 to the Explorer 16 Development Board.
6. Program the dsPIC DSC device on the board.
7. When the program is loaded, disconnect MPLAB ICD 3 from the board (remove the telephone cable from the MPLAB ICD 3 connector).

Note: The MPLAB REAL ICE can be used instead of the MPLAB ICD 3.

3.1.2.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into both the devices, the application is ready to run.

Use the following procedure to run the demonstration:

1. Press the Reset button on board 1. LED D3 on the board will light up indicating that the board is waiting for synchronization to be complete.
2. Press the Reset button on board 2. Now, LED D4 on both the boards will light up indicating that the boards are synchronized and communicating encoded speech data frames.
3. Listen to the decoded speech sample being played back repeatedly on the output device, such as headphones.

3.1.2.4 DEMONSTRATION CODE DESCRIPTION

The Demonstration code runs on a dsPIC DSC device, using the Primary Oscillator as the clock source with the PLL set for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This main function allocates all of the variables and arrays in data memory that are needed by the WM8510 codec driver, as well as the blocks of data memory that need to be allocated for the G.726A library functions.

The main function calls the `G726AEncoderInit()` and `G726ADecoderInit()` functions from the G.726A library, which initializes the G.726A encoder and decoder algorithms to their default and initial state. The main function also calls the `WM8510Init()`function to initialize the DCI module, the I^{2}C module, the WM8510 codec and the DCI interrupt.

The DCI module acts as a Master and drives the serial clock and frame synchronization lines. The DCI module acts as a Slave. The DCI module is set for the multi-channel Frame Sync Operating mode, with 16-bit data words and 16 data words or time slots per frame, of which only one transmit slot and one receive slot are used in this demonstration. The dsPIC DSC device will send the control information to WM8510 codec through the dsPIC DSC device's I^{2}C module. Subsequently, the function, `WM8510Start()`, is used to enable the DCI module and the I^{2}C module. The `WM8510SampleRate8KConfig()` configures the WM8510 codec for a sampling rate of 8 kHz.

The `PgmMemOpen()` function opens a handle to Flash program memory for reading. The main loop reads one frame from the Flash program memory using the function, `PgmMemRead()`, and passes it to the `G726ADecode()` function for decoding. The decoded speech frame is written to the WM8510 codec driver for playback.

3.1.3 Record Play Demonstration

The G.726A Record Play demonstration shows the use of G.726A library in a voice recorder type of application. The demonstration emphasizes the reduction in memory requirement for storing speech data. This demonstration is an example of a half-duplex system.

3.1.4 Demonstration Summary

The Record Play demonstration code example requires the use of one Explorer 16 Development Board with one Audio PICtail Plus Daughter Board (not included with the software license). A headset is connected to the Audio PICtail Plus Daughter Board.

In Record mode, the WM8510 codec on the Audio PICtail Plus Daughter Board will sample and convert the speech signal captured by the microphone and provide it to the dsPIC DSC device through the DCI module. The dsPIC DSC device will compress the speech signal using the G.726A encoder, pack the encoded data, and then store it in serial Flash memory available on the Audio PICtail Plus Daughter Board. In Playback mode, the application reads the packed encoded speech data stored in serial Flash memory, unpacks data, and then decodes this data using the G.726A decoder. The decoded speech data is written to the WM8510 codec for playback on the output device, such as headphones.

For a speech signal that is sampled at 8 kHz at 16-bit resolution, the resulting data rate is 128 kbps. To store one minute of raw speech data in memory would require 960 KB of memory. By encoding the speech signal using G.726A at 16 kbps bit rate, the memory required to store one minute of speech is 120 KB.

The code example invokes the `G726AEncode()` and `G726Decode()` functions from G.726A library to encode and decode speech data.

3.1.4.1 DEMONSTRATION SETUP

The demonstration application is intended to run on the Explorer 16 Development Board with an Audio PICtail Plus Daughter Board (not included with the software license).

Use the procedure in the following section to set up the demonstration.

3.1.4.1.1 Configure Explorer 16 Development Board and Audio PICtail Plus Daughter Board

Before applying power, you need to configure the boards:

1. On the Audio PICtail Plus Daughter Board, set jumper J4 (O/P SEL) to the codec position and set jumper J8 (LN/MIC) to the microphone position.
2. Insert the Audio PICtail Plus Daughter Board into the Explorer 16 Development Board PICtail Plus socket J5.
3. Connect the output device, such as headphones, to socket J10 on the Audio PICtail Plus Daughter Board.
4. Connect a microphone to socket J1 on the Audio PICtail Plus Daughter Board.
5. Once the setup is complete, apply power to the Explorer 16 Development Board.

3.1.4.1.2 Programming the dsPIC DSC Device

Use this process to load the Speex Playback demonstration into the dsPIC DSC device on the Explorer 16 Development Board.

1. On your PC, launch MPLAB IDE and open the `dsPIC33F G726A Record Play Demo.mcp` project located in the `RecordPlay` folder. For more information on using MPLAB IDE, refer to the *“MPLAB® IDE, Simulator, Editor User's Guide”* (DS51025).
2. Select *File > Import > dsPIC33F G726A Record Play Demo.hex* to import the project hexadecimal file.
3. Select *Programmer > Connect* to link MPLAB ICD 3 to the dsPIC DSC target device. The Output window confirms that MPLAB ICD 3 is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming has succeeded.

5. Connect MPLAB ICD 3 to the Explorer 16 Development Board.
6. Program the dsPIC DSC device on the board.
7. When the program is loaded, disconnect MPLAB ICD 3 from the board (remove the telephone cable from the MPLAB ICD 3 connector).

Note: The MPLAB REAL ICE can be used instead of the MPLAB ICD 3.

3.1.4.2 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into the device, the application is ready to run.

Use the following procedure to run the demonstration:

1. Press the Reset button on the board. LED D3 on the board will light up indicating that the application is running
2. Press switch S3 to start the erase and record process. LED D4 will light up indicating the serial Flash memory is being erased. After the erase is done, LED D4 turns off and LED D5 will switch on indicating that the recording is in progress. The microphone signal will be captured and stored in serial Flash memory.
3. Press Switch S6. This will stop the Record mode and start the Playback mode. LED D5 turns off and LED D6 lights up.
4. Press switch S3 to start recording again.

3.1.4.3 DEMONSTRATION CODE DESCRIPTION

The demonstration code runs on a dsPIC DSC device, using the Primary Oscillator as the clock source with the PLL set for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This main function allocates all of the variables and arrays in data memory that are needed by the WM8510 codec driver, the SST25VF040B serial Flash memory driver as well as the blocks of data memory that need to be allocated for the G.726A library functions.

The main function calls the `G726AEncoderInit()` and `G726ADecoderInit()` functions from the G.726A library, which initializes the G.726A encoder and decoder algorithm to its default and initial state.

The main function also calls the `WM8510Init()`function to initialize the DCI and I^{2}C modules, the WM8510 codec and the DCI interrupt. The WM8510 codec module acts as a Master and drives the serial clock and frame synchronization lines. The DCI module acts as a Slave. The DCI module is set for the multi-channel Frame Sync Operating mode, with 16-bit data words and 16 data words or time slots per frame, of which only one transmit slot and one receive slot are used in this demonstration. The dsPIC DSC device will send the control information to WM8510 codec through the I^{2}C module. Subsequently, the `WM8510Start()` function is used to enable the DCI the I^{2}C modules. The function, `WM8510SampleRate8KConfig()`, configures the WM8510 codec for a sampling rate of 8 kHz.

The `SST25VF040BInit()` function is called to initialize the serial Flash memory. The `SST25VF040BStart()` function enables the SPI module, which communicates with the serial Flash memory.

The main function polls the switches to determine if Record mode or Playback mode is selected. If Record mode is selected, the `SST25VF040BIOCtl()` function is called with the chip erase command. When the chip erase is complete, the current speech frame is encoded using the `G726AEncode()`function and written to the Flash program memory using the `SST25VF040BWrite()` function. The speech frame is also provided to the WM8510 codec driver for output to the headphones. If Playback mode is selected, the serial Flash memory is read using the `SST25VF040BRead()` function, decoded using the `G726ADecode()` function, and written to the WM8510 codec driver for output to the headphones.

3.2 QUICK START DEMONSTRATION FOR dsPIC33E DEVICE FAMILY

The following topics are covered in this section:

- Communication Demonstration
- Playback Demonstration
- Record Play Demonstration

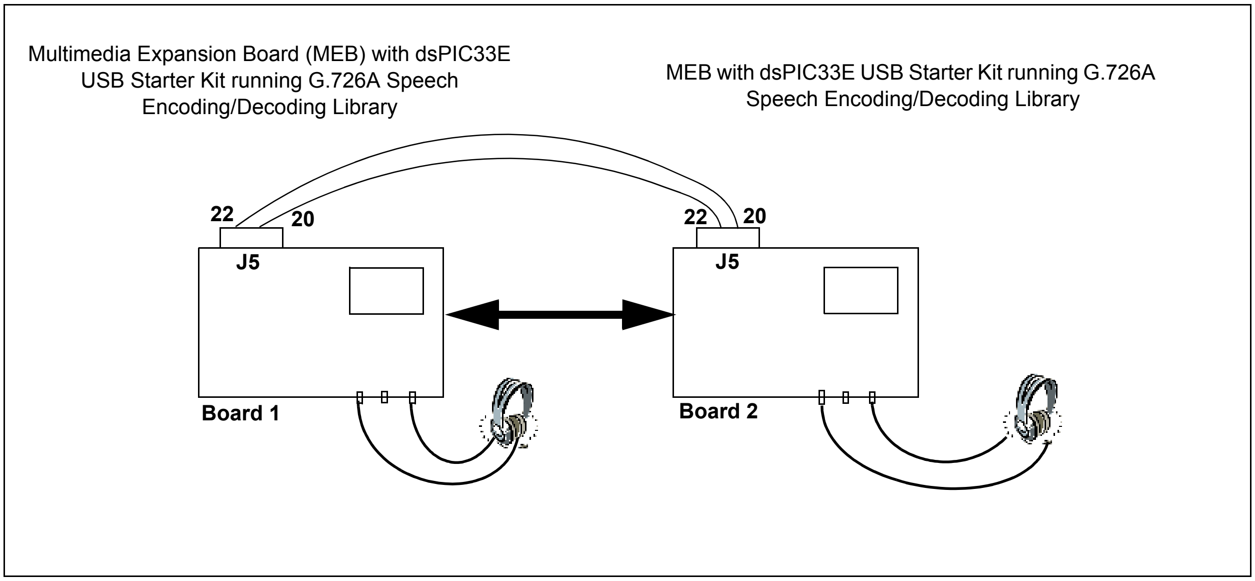
3.2.1 Communication Demonstration

The Communication demonstration shows how the G.726A library can be used to reduce the required bandwidth in a full-duplex communication type of application running on a dsPIC33E device.

3.2.1.1 DEMONSTRATION SUMMARY

The Communication code example requires the use of two Multimedia Expansion Boards (MEB) with two dsPIC33E USB Starter Kits (not included with the software license), which are set up as shown in Figure 3-4.

FIGURE 3-4: G.726A COMMUNICATION DEMONSTRATION SETUP


Multimedia Expansion Board (MEB) with dsPIC33E USB Starter Kit running G.726A Speech Encoding/Decoding Library
MEB with dsPIC33E USB Starter Kit running G.726A Speech Encoding/Decoding Library
22
20
22
20
J5
J5
Board 1
Board 2

Headsets are connected to the MEBs. When a user speaks into the headset connected to board 1, the WM8731 codec on the MEB will sample and convert the data, and provide it to the dsPIC DSC device through the DCI module. The dsPIC DSC device will compress the speech signal using the G.726A encoder, and then transmit the compressed speech frame data through the UART2 module and I/O expansion connector to board 2.

The dsPIC DSC device on board 2 receives the encoded frame through the on-board I/O expansion connector and the device's UART2 module. The dsPIC DSC device decodes the received frame using the G.726A decoder, and then plays out the signal on the headset through its DCI module and the on-board WM8731 codec.

For a speech signal that is sampled at 8 kHz at 16-bit resolution, the resulting data rate is 128 kbps. To communicate this data to a remote terminal over a communication link, the minimum required communication link bandwidth would be 128 kbps. By encoding the speech signal using 16 kbps mode, the minimum communication link bandwidth would be 16 kbps. This results in a bandwidth reduction of 8x.

The code example invokes the `G726AEncode()` and `G726ADecode()` functions from G.726A library to encode and decode speech data.

3.2.1.2 DEMONSTRATION SETUP

Use the procedure in the following section to set up the demonstration.

3.2.1.2.1 Configure MEBs and dsPIC33E USB Starter Kits

Before applying power, you need to configure the boards:

1. Insert a dsPIC33E USB Starter Kit into the starter kit connector on MEB 1.
2. Connect the dsPIC33E USB Starter Kit to a PC using the USB A-to-mini-B cable provided with the Starter Kit.
3. Connect the output device, such as headphones, to MEB 1.
4. Connect a microphone to MEB 1.
5. Repeat steps 1 though 4 for the second dsPIC33E USB Starter Kit and MEB.
6. Using a single-strand wire, connect pin 20 of the I/O expansion connector on MEB 1 to pin 22 of the I/O expansion connector on MEB 2.
7. Using another single-strand wire, connect pin 22 of the I/O expansion connector on MEB 1 to pin 20 of the I/O expansion connector on MEB 2.
8. Once the setup is complete, it should be similar to the one shown in Figure 3-4.

3.2.1.2.2 Programming the dsPIC DSC Device

Use this process to load the G.726A communication demonstration into the dsPIC DSC device on the MEB.

1. Using MPLAB IDE, open the `dsPIC33E G726A Communication Demo.mcp` project located in the `Communication` folder. For more information on using MPLAB IDE, refer to the *"MPLAB® IDE, Simulator, Editor User's Guide"* (DS51025).
2. Select either *File > Import > dsPIC33E G726A Communication Demo 1.hex* to import the project hexadecimal file for board 1, or select *File > Import > dsPIC33E G726A Communication Demo 2.hex* to import the project hexadecimal file for board 2.
3. Select *Programmer > Starter Kit on Board* as the programmer, and then select *Connect* to link to the dsPIC DSC target device. The Output window confirms that the target device is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming has succeeded.

Note: After programming, unplug and reconnect the USB cable to the starter kit, to ensure that the WM8731 audio codec can be reconfigured.

5. Repeat steps 2 through 4 for the second board.

3.2.1.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into both the devices, the application is ready to run.

Use the following procedure to run the demonstration:

1. Using MPLAB IDE, reset and run board 1. LED D1 on the board will light up indicating that the board is waiting for synchronization to be complete.
2. Using MPLAB IDE, reset and run board 2. Now, LED D2 on both the boards will light up indicating that the boards are synchronized and communicating encoded speech data frames.
3. Speak into the microphone connected to board 1. This speech will be heard in the output device, such as headphones, connected to board 2 and vice-versa.

Note: If only one PC is available for running the demonstration, program board 2, disconnect the USB cable from board after programming, and then use an external 9V power supply to reset and run board 2 (after the program in board 1 is already running).

3.2.1.4 DEMONSTRATION CODE DESCRIPTION

The Demonstration code runs on a dsPIC DSC device, using the Primary Oscillator as the clock source with the PLL set for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This main function allocates all of the variables and arrays in data memory that are needed by the UART and WM8731 codec driver, as well as the blocks of data memory that need to be allocated for the G.726A library functions.

The main function calls the `G726AEncoderInit()` and `G726ADecoderInit()` functions from the G.726A library, which initializes the G.726A encoder and decoder algorithms to their default and initial state.

The main function also calls the `WM8731Init()` function to initialize the DCI and I^2C modules, the WM8731 codec and the DCI interrupt. This configures the WM8731 codec for a sampling rate of 8 kHz. The WM8731 codec module acts as a Master and drives the serial clock and frame synchronization lines. The DCI module acts as a Slave. The DCI module is set for the multi-channel Frame Sync Operating mode, with 16-bit data words and two data words or time slots per frame (only 1 transmit slot and 1 receive slot are used in this demonstartion). The dsPIC DSC device will send the control information to the WM8731 codec through the I^2C module. Subsequently, the `WM8731Start()` function is used to enable the DCI and I^2C modules.

The `UART2Init()` function configures the UART2 module on the dsPIC DSC for operation at the baud rate specified in `UART2Drv.h`. The driver provides a blocking interface through the `UART2Read()` and `UART2Write()` functions. The main processing uses the `UART_ioctl()` function with a `UART2_RX_NDATA` command to know when a frame of data has been received and when frame transmission has been completed.

The `BoardSychronize()` function is called to ensure that the two boards are synchronized. The UART driver is checked for a full frame of encoded data. If a frame of data has been received, the `G726ADecode()` function is called to decode this frame. The decoded speech frame is written to the WM8731 codec driver. The codec driver is polled for a frame of raw speech samples. If the frame is ready, the function, `G726AEncode()`, is called. The encoded frame is written to the UART2 driver for transmission.

The encoded G.726A data is packed using the `G726APack()` function, which reduces the bandwidth needed by the communication channel. Similarly, the received data is unpacked using the `G726AUnpack()` function before it is decoded.

3.2.2 Playback Demonstration

The Playback demonstration shows how the G.726A library can be used in a playback type of application where encoded frames are read from Flash program memory, decoded by the Speex decoder, and then output. By storing the encoded speech frames in Flash program memory, the application can minimize external memory usage.

3.2.2.1 DEMONSTRATION SUMMARY

The Playback code example requires the use of one MEB with one dsPIC33E USB Starter Kit (not included with the software license). A headset is connected to the MEB. The demonstration code reads G.726A encoded audio data, which are stored in Flash program memory. The encoded audio data is then decoded using the G.726A decoder. The decoded speech data is written to the WM8731 codec for playback on the output device, such as headphones.

The encoded speech data is obtained using the G.726A PCEU. The resulting `.s` file generated by the PCEU is included into the project and compiled. The encoded speech data now becomes a part of Flash program memory. The code example invokes the `G726ADecode()` function from G.726A library to decode speech data.

3.2.2.2 DEMONSTRATION SETUP

The demonstration application is intended to run on the MEB with a dsPIC33E USB Starter Kit (not included with the software license).

Use the procedure in the following section to set up the demonstration.

3.2.2.2.1 Configure MEB and dsPIC33E USB Starter Kit

Before applying power, you need to configure the boards:

1. Insert a dsPIC33E USB Starter Kit into the starter kit connector on the MEB.
2. Connect the dsPIC33E USB Starter Kit to a PC using the USB A-to-mini-B cable provided with the dsPIC33E USB Starter Kit.
3. Connect an output device, such as headphones, to the MEB.

3.2.2.2.2 Programming the dsPIC DSC Device

Use this process to load the G.726A Playback demonstration into the dsPIC DSC device on the Explorer 16 Development Board:

1. On your PC, launch MPLAB IDE and open the `dsPIC33E G726A Playback Demo.mcp` project file located in the `Playback` folder. For more information on using MPLAB IDE, refer to the *"MPLAB® IDE, Simulator, Editor User's Guide"* (DS51025).
2. Select *File > Import > dsPIC33E G726A Playback Demo.hex* to import the project hexadecimal file
3. Select *Programmer > Starter Kit on Board* as the programmer, and then select *Programmer > Connect* to link to the dsPIC DSC target device. The Output window confirms that the target device is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming has succeeded.

> **Note:** If the USB cable is already plugged in, then unplug and reconnect the USB cable to the dsPIC33E USB Starter Kit, to ensure that the WM8731 audio codec can be reconfigured.

3.2.2.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into the device, the application is ready to run.

Use the following procedure to run the demonstration:

1. Using MPLAB IDE, reset and run the board.
2. Listen to the speech sample being played back repeatedly on the output device, such as headphones.

3.2.2.4 DEMONSTRATION CODE DESCRIPTION

The demonstration code runs on a dsPIC DSC device, using the Primary Oscillator as the clock source with the PLL set for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This main function allocates all of the variables and arrays in data memory that are needed by the WM8731 codec driver, as well as the blocks of data memory that need to be allocated for the dsPIC DSC G.726A Speech Encoding/Decoding Library functions.

The main function calls the `G726AEncoderInit()` and `G726ADecoderInit()` functions from the G.726A library, which initializes the G.726A encoder and decoder algorithms to their default and initial state.

The main function also calls the `WM8731Init()`function to initialize the DCI and I^2C modules, the WM8731 codec and the DCI interrupt. This configures the WM8731 codec for a sampling rate of 8 kHz. The DCI module acts as a Master, and drives the serial clock and frame synchronization lines. The DCI module acts as a Slave. The DCI module is set for the multi-channel Frame Sync Operating mode, with 16-bit data words and 2 data words or time slots per frame (only one transmit slot and one receive slot are used in this demonstration). The dsPIC DSC device will send the control information to WM8731 codec through the I^2C module. Subsequently, the `WM8731Start()` function is used to enable the DCI and I^2C modules.

The `PgmMemOpen()` function opens a handle to Flash program memory for reading. The main loop reads one frame from the Flash program memory using the `PgmMemRead()` and passes it to the `G726ADecode()` function for decoding. The decoded speech frame is written to the WM8731 codec driver for playback.

3.2.3 Record Play Demonstration

The G.726A Record Play demonstration shows the use of the G.726A library in a voice recorder type of application. The demonstration emphasizes the reduction in memory requirement for storing speech data. This demonstration is an example of a half-duplex system.

3.2.3.1 DEMONSTRATION SUMMARY

The Record Play demonstration code example requires the use of one MEB with one dsPIC33E USB Starter Kit (not included with the software license). A headset is connected to the MEB.

In Record mode, the WM8731 codec on the MEB will sample and convert the speech signal captured by the microphone, and provide it to the dsPIC DSC device through the DCI module. The dsPIC DSC device will compress the speech signal using the G.726A encoder, pack the encoded data, and then store it in the serial Flash memory available on the MEB. In Playback mode, the application reads the packed encoded speech data stored in serial Flash memory, unpacks the data, and then decodes this data using the G.726A decoder. The decoded speech data is written to the WM8731 codec for playback on the headphones.

For a speech signal that is sampled at 8 kHz at 16-bit resolution, the resulting data rate is 128 kbps. To store one minute of raw speech data in memory would require 960 KB of memory. By encoding the speech signal using G.726A 16 kbps bit rate, the memory required to store one minute of speech is 120 KB. The code example invokes the `G726AEncode()` and `G726Decode()` functions from G.726A library to encode and decode speech data.

3.2.3.2 DEMONSTRATION SETUP

The demonstration application is intended to run on the MEB with a dsPIC33E USB Starter Kit (not included with the software license).

Use the procedure outlined in the following section to set up the demonstration.

3.2.3.2.1 Configure MEB and dsPIC33E USB Starter Kit

Before applying power, you need to configure the boards:

1. Insert a dsPIC33E USB Starter Kit into the starter kit connector on the MEB.
2. Connect the dsPIC33E USB Starter Kit to a PC using the USB A-to-mini-B cable provided with the dsPIC33E USB Starter Kit.
3. Connect an output device, such as headphones, to the MEB.

3.2.3.2.2 Programming the dsPIC DSC Device

Use this process to load the Speex Playback demonstration into the dsPIC DSC device on the Explorer 16 Development Board.

1. On your PC, launch MPLAB IDE and open the `dsPIC33E G726A Playback Demo.mcp` project file located in the `Playback` folder. For more information on using MPLAB IDE, *"MPLAB® IDE, Simulator, Editor User's Guide"* (DS51025).
2. Select *File > Import > dsPIC33E G726A Playback Demo.hex* to import the project hexadecimal file.
3. Select *Programmer > Starter Kit on Board* as the programmer, and then select *Programmer > Connect* to link to the dsPIC DSC target device. The Output window confirms that the target device is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming has succeeded.

Note:	If the USB cable is already plugged in, then unplug and reconnect the USB cable to the dsPIC33E USB Starter Kit, to ensure that the WM8731 audio codec can be reconfigured.

3.2.3.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into the device, the application is ready to run.

Use the following procedure to run the demonstration:

1. Using the MPLAB IDE, reset and run the board.
2. Press switch S1 for a few seconds, to start the erase and record process. Release the switch when LED D1 lights up indicating the serial Flash memory is being erased. After the erase is done, LED D1 turns off and LED D2 will switch on indicating that recording is in progress. The microphone signal will be captured and stored in serial Flash memory.
3. Press Switch S1 again for a few seconds. This will stop the Record mode and start the Playback mode. Release the switch when LED D2 turns off and LED D3 lights up.
4. To start recording again, press switch S1 again, as described in step 2.

3.2.3.4 DEMONSTRATION CODE DESCRIPTION

The demonstration code runs on a dsPIC DSC device, using the Primary Oscillator as the clock source with the PLL set for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This main function allocates all the variables and arrays in data memory that are needed by the WM8731 codec driver, the SST25VF016B serial Flash memory driver as well as the blocks of data memory that need to be allocated for the G.726A library functions.

The main function calls the `G726AEncoderInit()` and `G726ADecoderInit()` functions from the G.726A library, which initializes the G.726A encoder and decoder algorithm to its default and initial state.

The main function also calls the `WM8731Init()`function to initialize the DCI and I^{2}C modules, the WM8731 codec and the DCI interrupt. This configures the WM8731 codec for a sampling rate of 8 kHz. The WM8731 codec module acts as a Master and drives the serial clock and frame synchronization lines. The DCI module acts as a Slave. The DCI module is set for the multi-channel Frame Sync Operating mode, with 16-bit data words and 2 data words or time slots per frame (only one transmit slot and one receive slot are used in this demonstration).

The dsPIC DSC device will send the control information to WM8731 codec through the I²C module. Subsequently, the `WM8510Start()` function is used to enable the DCI and I²C modules.

The `SST25VF016BInit()` function is called to initialize the serial Flash memory. The `SST25VF016BStart()` function enables the SPI module, which communicates with the serial Flash memory.

The main function polls the switches to determine if Record mode or Playback mode is selected. If Record mode is selected, the `SST25VF016BIOCtl()` function is called with the chip erase command. When the chip erase is complete, the current speech frame is encoded using the `G726AEncode()`function and written to Flash program memory using the `SST25VF016BWrite()` function. The speech frame is also provided to the WM8731 codec driver for output to the headphones. If Playback mode is selected, the serial Flash memory is read using the `SST25VF016BRead()` function, decoded using the `G726ADecode()` function, and then written to the WM8731 codec driver for output to the headphones.

NOTES:

Chapter 4. Application Programming Interface

This chapter describes the Application Programming Interface (API) functions that are available in the G.726A library.

The following topics are covered in this chapter:

- Adding the Library to the Application
- Using the Library
- Resource Requirements
- Library Interface
- Application Tips

4.1 ADDING THE LIBRARY TO THE APPLICATION

To use the G.726A library in an application, the library archive must be added to the application project workspace. The `G726A.h` file must be included in the application code. Consider the following when adding the library files to an application:

- If the application only requires a one bit rate, a bit rate specific archive can be added to the project.
- If the application requires more than one bit rate, the `libG726A_33F.a` or `libG726A_33E.a` file should be added to the project.

Use the following procedure to add the library to the application:

1. In the application, MPLAB workspace, right-click **Library Files** in the Project Window and select **Add files**.
2. Browse to the location of the desired G.726A library archive (available in the `libs` folder of the installation directory).
3. Select the desired file, and then click **Open**. The library is added to the application.

To use the library functions, include the `G726A.h` file in the application source code. This file can be copied from the `h` folder (located in the library installation directory) to the application project folder.

4.2 USING THE LIBRARY

The G.726A library has been designed to be usable in a re-entrant environment. This feature enables the algorithm to process multiple channels of audio. Since the encoder and the decoder algorithms operate independent of each other, an application does not have to instantiate a decoder or encoder if it is not needed. Each encoder or decoder must have it own state memory which retains information about the encoder or decoder between calls.

Use the following procedure to enable the use of G.726A library:

1. Configure the size of the frame to be processed by configuring the `G726A_FRAME_SIZE` macro in the `G726A.h` file.

   For example, if the encoder must encode 10 samples in one call, then set the value of the macro to 10. Example 4-1 shows the rest of the steps required to use the encoder and decoder.

EXAMPLE 4-1: LIBRARY USAGE EXAMPLE

```
include "G726A.h"

unsigned char encoder1_16kbps[G726A_ENCODER_SIZE];   /* Step 1 */

unsigned char encoder2_40kbps[G726A_ENCODER_SIZE];
unsigned char decoder1_16kbps[G726A_DECODER_SIZE];   /* Step 2 */
unsigned char decoder2_40kbps[G726A_DECODER_SIZE];

int input1[G726_FRAME_SIZE];                         /* Step 3 */
int input2[G726_FRAME_SIZE];
unsigned char output1[G726A_FRAME_SIZE];
unsigned char output2[G726A_FRAME_SIZE];
int i;

int main(void)
{
G726AEncoderInit(encoder1_16kbps, G726_16KBPS, G726A_FRAME_SIZE); /* Step 4 */
G726AEncoderInit(encoder1_40kbps, G726_40KBPS, G726A_FRAME_SIZE);
G726ADecoderInit(decoder1_16kbps, G726_16KBPS, G726A_FRAME_SIZE); /* Step 5 */
G726ADecoderInit(decoder1_40kbps, G726_40KBPS, G726A_FRAME_SIZE);

for(i = 0; i < G726A_FRAME_SIZE; i ++)
{
input1[i] = input1[i] >> 2;                          /* Step 6 */
input2[i] = input2[i] >> 2;
}

G726AEncode(encoder1_16kbps, input1, output1);       /* Step 7 */
G726AEncode(encoder1_40kbps, input2, output2);
G726ADecode(decoder1_16kbps, output1, input1);       /* Step 8 */
G726ADecode(decoder1_40kbps, output2, input2);
```

2. Allocate memory for the required number of encoders. In Example 4-1, two encoders are instantiated.
3. Allocate memory for the required number of decoders. In Example 4-1, two decoders are instantiated.
4. Allocate memory for input and output buffers. Note the data type of input and output buffers.
5. Initialize all the encoders by calling the `G726AEncoderInit()` function. The encoder bit rate and the frame size must be specified. All the encoders must be initialized before they are used.
6. Initialize all the encoders by calling the `G726ADecoderInit()` function. The decoder bit rate and the frame size must be specified. All the decoders must be initialized before they are used.
7. Shift the input data to the right, to have a 14-bit value. This step is optional, if the input data is already a 14-bit value.
8. Use the `G726AEncode()` function call to encode the input frame.

4.3 RESOURCE REQUIREMENTS

The resource requirements for running the G.726A library depends on which library is selected, either the Narrowband-only version (`libspeex_8k_33F.a` and `speex_8k.h`) or the Wideband and Narrowband version (`libspeex_33F.a` and `speex.h`). The resource requirements for each of these versions are provided in the following sections and tables.

TABLE 4-1: PROGRAM MEMORY USAGE

Resource	Size (bytes)	Section
Code and Tables	3054 + 540 = 3594 (dsPIC33F) 3921 + 0 = 3921 (dsPIC33E)	`.text` and `.const`

TABLE 4-2: DATA MEMORY USAGE

Function	Size (bytes)	Alignment	Section
Encoder State Memory	152	2	X data memory
Decoder State Memory	152	2	X data memory
Tables in Data Memory	360 (dsPIC33E only)	2	X data memory

The Data memory usage specifications do not include memory required for input and output buffers. The size of these buffers is user definable.

TABLE 4-3: DYNAMIC MEMORY

Section	Size (bytes)
`Heap`	0
`Stack`	< 60

TABLE 4-4: COMPUTATIONAL SPEED

Function	MIPS	Typical Call Frequency
`G726AEncode ()`	7	Once every frame
`G726ADecoder ()`	7	Once every frame

4.4 LIBRARY INTERFACE

This section describes the dsPIC DSC G.726A Speech Encoding/Decoding Library interface. The following topics are described in this section:

- Encoder Functions
- Decoder Functions
- Types
- Macros

4.4.1 Encoder Functions

Table 4-5 lists the functions related to the G.726A Encoder.

TABLE 4-5: ENCODER FUNCTIONS

Function	Description
`G726AEncode`	This routine encodes one input frame.
`G726AEncoderInit`	This routine allows initializes a G.726A encoder to operate at the specified output bit rate and input frame size.

G726AEncode

Description

This routine encodes one input frame. The size of the frame and the bit rate to be used should be defined during encoder initialization. Irrespective of the bit rate used, each output sample requires one byte. Therefore, the output array should be an unsigned character array of the same size as the input array. Note that the input to the encode function should be a 14-bit signed integer as specified by the ITU-T G.726A specification.

Include

```
G726A.h
```

Prototype

```
void G726AEncode (
   unsigned char * encoder,
   int * input Frame,
   unsigned char * output Frame
);
```

Preconditions

The encoder state memory should have been initialized.

Arguments

`encoder`	encoder state memory.
`inputFrame`	input array containing samples to be encoded. The samples should be 14-bit signed integer values. The size of the array should match the encoder frame size specified during encoder initialization.
`outputFrame`	output array where the encoded samples will be stored. Each byte of output array will store one encoded sample. Depending on the selected encoder bit rate, only certain number of Least Significant Bytes (LSBs) in the output sample byte will be significant. Refer to the `G726A_BIT_RATES` type for more details.

Return Value

None.

Remarks

None.

Code Example

```
// This code snippet demonstrates how to use the
// G.726A encoder. The encoder is first initialized
// and then the encode function is used to encode
// the input. Note the input and output array size
// and types. The input should be right shifted by
// 2 bits for 16 bit values. This will make the input
// 14 bit as required by the encoder.

unsigned char encoder16kbps[G726A_ENCODER_SIZE];
int input[G726A_FRAME_SIZE];
unsigned char output[G726A_FRAME_SIZE];
int i;

G726AEncoderInit(encoder16kbps, G726A_16KBPS, G726A_FRAME_SIZE);

for(i = 0; i < G726A_FRAME_SIZE; i ++)
{
    // Not necessary if its known that input
    // is 14 bits or less.

    input[i] = input[i] >> 2;
}

G726AEncode(encoder16kbps, input, output);
```

G726AEncoderInit

Description

This routine initializes a G.726A encoder to operate at the specified output bit rate and input frame size. Multiple encoders can instantiated, each encoder operating at its own bit rate and it own input frame size.

Include

```
G726A.h
```

Prototype

```
void G726AEncoderInit (
   unsigned char * encoder,
   G726A_BIT_RATES rate,
   int framesize
);
```

Preconditons

The encoder state memory for each decoder should have been allocated.

Arguments

`encoder`	encoder state memory.
`rate`	specifies the bit rate at which the encoder will operate.
`frameSize`	specifies the encoder input frame size in samples.

Return Value

None.

Remarks

None.

Code Example

```
// This code snippet shows how to initialize two G.726A
// encoders, one to operate at 16 kbps and the
// other to work at 24 kbps. Note the state
// memory sizes for both encoders is the same. The
// input frame size for each encoder can be different.

unsigned char encoder16kbps[G726A_ENCODER_SIZE];
unsigned char encoder24kbps[G726A_ENCODER_SIZE];

G726AEncoderInit(encoder16kbps, G726A_16KBPS, G726A_FRAME_SIZE);
G726AEncoderInit(encoder24kbps, G726A_24KBPS, G726A_FRAME_SIZE);
```

4.4.2 Decoder Functions

Table 4-6 lists the functions related to the G.726A decoder.

TABLE 4-6: DECODER FUNCTIONS

Function	Description
G726ADecode	This routine decodes one input frame.
G726ADecoderInit	This routine allows initialization of a G.726A decoder to operate at the specified output bit rate and input frame size.

G726ADecode

Description

This routine decodes one input frame. The size of the frame and the bit rate to be used should have been defined during decoder initialization. The input array should be an unsigned character array of the same size as the output array. Note that the output of the decode function will be a 14-bit signed integer as specified by the ITU-T G.726A specification.

Include

```
G726A.h
```

Prototype

```
void G726ADecode (
    unsigned char * decoder,
    unsigned char * inputFrame
    int * outputFrame
);
```

Preconditions

The decoder state memory should have been initialized.

Arguments

decoder	decoder state memory.
inputFrame	input array containing samples to be decoded. The samples should have been encoded using the same bit rate as decoder bit rate. The encoded samples should be arranged such that one element of the array holds one encoded sample.
outputFrame	output array where the decoded samples will be stored. The decoded value is a 14-bit signed integer value.

Return Value

None.

Remarks

None.

Code Example

```
// G.726A decoder. The decoder is first initialized
// and then the decode function is used to decode
// the input. Note the input and output array
// size and types. The output array can be left
// shifted by 2 bits to scale up the 14 bit output
// value to 16 bit.

unsigned char decoder16kbps[G726A_DECODER_SIZE];
int output[G726A_FRAME_SIZE];
unsigned char input[G726A_FRAME_SIZE];

G726ADecoderInit(decoder16kbps, G726A_16KBPS, G726A_FRAME_SIZE);
G726ADecode(decoder16kbps, input, output);

for(i = 0; i < G726A_FRAME_SIZE; i ++)
{
    // Not necessary if the input to encoder
    // which generated the encoded samples
    // where not right shifted.

    output[i] = output[i] << 2;
}
```

G726ADecoderInit

Description

This routine initializes a G.726A decoder to operate at the specified output bit rate and input frame size. Multiple decoders can instantiated, each decoder operating at its own bit rate and it own input frame size.

Include

```
G726A.h
```

Prototype

```
void G726ADecoderInit (
    unsigned char * decoder,
    G726A_BIT_RATES rate,
    int framesize
);
```

Preconditions

The decoder state memory for each decoder should have been allocated.

Arguments

`decoder`	decoder state memory.
`rate`	specifies the bit rate at which the decoder will operate.
`frameSize`	specifies the decoder input frame size in samples.

Return Value

None.

Remarks

None.

Code Example

```
// This code snippet shows how to initialize two G.726A
// decoders, one to operate at 16 kbps and the
// other to work at 24 kbps. Note the state
// memory sizes for both decoders is the same. The
// input frame size for each decoder can be different.

unsigned char decoder16kbps[G726A_DECODER_SIZE];
unsigned char decoder24kbps[G726A_DECODER_SIZE];

G726ADecoderInit(decoder16kbps, G726A_16KBPS, G726A_FRAME_SIZE);
G726ADecoderInit(decoder24kbps, G726A_24KBPS, G726A_FRAME_SIZE);
```

4.4.3 Types

Table 4-7 provides the data type used by the dsPIC DSC G.726A Speech Encoding/Decoding Library.

TABLE 4-7: DATA TYPE

Name	Description
`G726A_BIT_RATES`	This enumeration identifies the supported G.726A encoder and decoder bit rates.

Include

```
G726A.h
```

G726A_BIT_RATES

Description

G726A_BIT_RATES

This enumeration identifies the supported G.726A encoder and decoder bit rates. The bit rates for the encoder and decoder are specified at the time of encoder and decoder initialization.

Include

```
G726A.h
```

Prototype

```
typedef enum {
  G726A_16KBPS = 2,
  G726A_24KBPS = 3,
  G726A_32KBPS = 4,
  G726A_40KBPS = 5
} G726A_BIT_RATES;
```

TABLE 4-8: MEMBERS

Members	Description
`G726A_16KBPS = 2`	G.726A Encoder/Decoder work at 16 kbps. 2 LSBs of encoded byte are significant Compression ratio 7:1.
`G726A_24KBPS = 3`	G.726A Encoder/Decoder work at 24 kbps. 3 LSBs of encoded byte are significant Compression ratio 4.6:1.
`G726A_32KBPS = 4`	G.726A Encoder/Decoder work at 32 kbps. 4 LSBs of encoded byte are significant Compression ratio 3.5:1.
`G726A_40KBPS = 5`	G.726A Encoder/Decoder work at 40 kbps. 5 LSBs of encoded byte are significant Compression ratio 2.8:1.

4.4.4 Macros

Table 4-9 provides the macros used by the dsPIC DSC G.726A Speech Encoding/Decoding Library.

TABLE 4-9: MACROS

Name	Description
G726A_Decoder_SIZE	This defines the size of the G.726A decoder state structure in bytes.
G726A_Encoder_SIZE	This defines the size of the G.726A encoder state structure in bytes.
G726A_FRAME_SIZE	This defines the size of the G.726A encoder input frame size in samples, and is user definable. Minimum size is 1.

Include

```
G726A.h
```

G726A_Decoder_SIZE

Description

This defines the size of the G.726A decoder state structure in bytes.

Include

```
G726A.h
```

Prototype

```
#define G726A_DECODER_SIZE 76
```

G726A_Encoder_SIZE

Description

This defines the size of the G.726A encoder state structure in bytes.

Include

```
G726A.h
```

Prototype

```
#define G726A_ENCODER_SIZE 76
```

G726A_FRAME_SIZE

Description

This defines the size of the G.726A encoder input frame size in samples and is user definable. Minimum size is 1.

Include

```
G726A.h
```

Prototype

```
#define G726A_FRAME_SIZE 8
```

4.5 APPLICATION TIPS

The following are a few tips to consider when using the G.726A library.

1. The optimum input signal levels for testing audio systems are generally considered to be between -10 dBm0 and -30 dBm0. If the digital input speech levels have peaks that are up to three-fourth of full range, good use is being made of the available precision; levels higher than this carry a risk of amplitude clipping.
2. It is possible for the Encoder and Decoder to share scratch memories. For this to occur, the Encoder and Decoder functions should be invoked in the same process thread or task. That is, the Encoder or Decoder sharing scratch memories should not be invoked in an Interrupt Service Routine (ISR). To use the same scratch memories, allocate the Encoder X and Y scratch memories and set up the Decoder to use these.
3. While using multiple encoders or decoders, it is possible for the encoders and decoders to share the same scratch memory. The precautions mentioned in tip 2 should be followed.

Chapter 5. PC Encoding Utility

The G.726A library includes a PC Encoding Utility (PCEU) also called the dsPIC33F Speech Encoding Utility, which allows users to encode a microphone signal or a `.wav` file and generate a file containing G.726A encoded samples. This generated file can then be readily included in the user application to store the G.726A encoded samples either in the device Flash program memory or RAM. This is suitable for Playback type of applications where a pre-recorded message is played back on the output device. The dsPIC33F Speech Encoding Utility is available in the `PCEU` folder of the library installation directory.

The following sections provide the steps to be followed, to encode a microphone signal connected to the PC sound card and to encode a `.wav` file.

- Microphone
- WAVE File
- Output Files

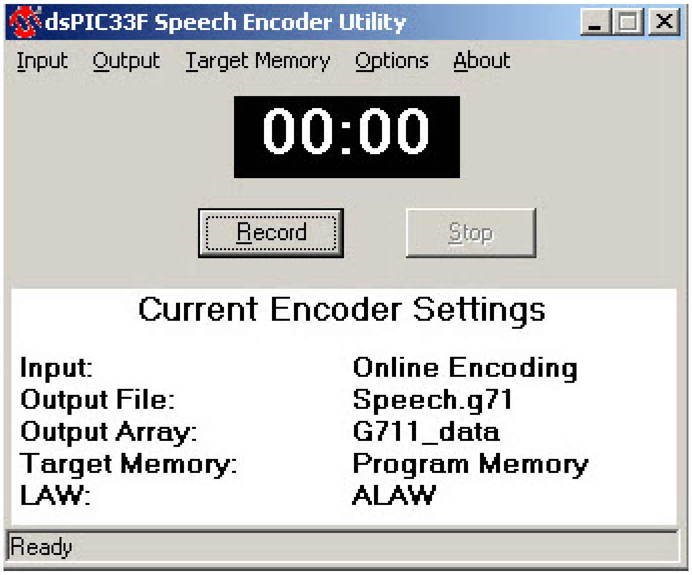
> **Note:** The dsPIC33F Speech Encoding Utility can be used to generate the encoded speech for either dsPIC33F or dsPIC33E.

5.1 MICROPHONE

This section provides the steps to be followed to use the dsPIC33F Speech Encoding Utility, to encode a signal from the microphone connected to the PC sound card.

1. Connect a compatible microphone to the microphone input of the PC sound card. Use any of the available system utilities to ensure nominal microphone gain.
2. Double-click `dsPICSpeechRecord.exe` in the `PCEU` folder of the library installation directory to start the utility.

FIGURE 5-1: dsPIC33F SPEECH ENCODER UTILITY



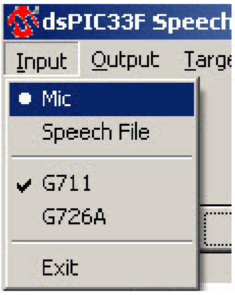
Current Encoder Settings

Input: Online Encoding
Output File: Speech.g71
Output Array: G711_data
Target Memory: Program Memory
LAW: ALAW

Ready

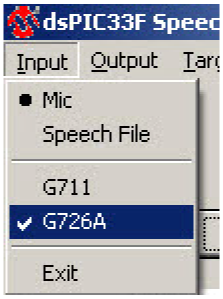
3. Select *Input > Mic* to select the microphone connected to the PC sound card.

FIGURE 5-2: INPUT SOURCE SELECTION



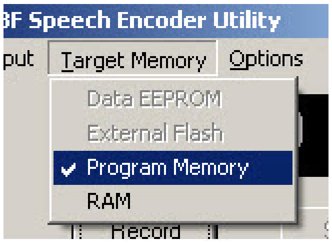
4. Select *Input > G726A* to select the G.726A encoder, to encode the microphone signal.

FIGURE 5-3: ENCODER SELECTION



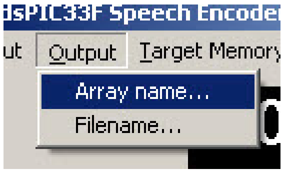
5. Select either *Target Memory > Program Memory* to store the encoded speech in Flash program memory, or *Target Memory > RAM* to store the encoded speech in RAM. The generated encoded samples can be either stored in device RAM or in Flash program memory. Depending on the selected option, the PCEU either generates a `.c` file for RAM, or generates a `.s` file for Flash program memory.

FIGURE 5-4: TARGET MEMORY SELECTION



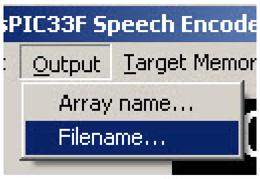
6. Select *Output > Array name* to specify the name of the encoded samples array in the generated file.

FIGURE 5-5: ARRAY NAME SELECTION



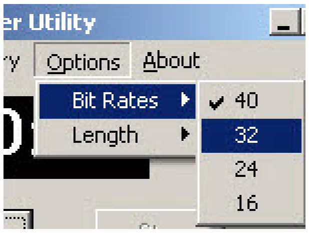
7. Select Output > Filename to specify the name of the generated file and the path where the generated file will be stored.

FIGURE 5-6: FILE NAME SELECTION



8. Select Options > Bit Rates to specify the G.726A bit rate to be used while generating the encoded samples.

FIGURE 5-7: BIT RATE SELECTION



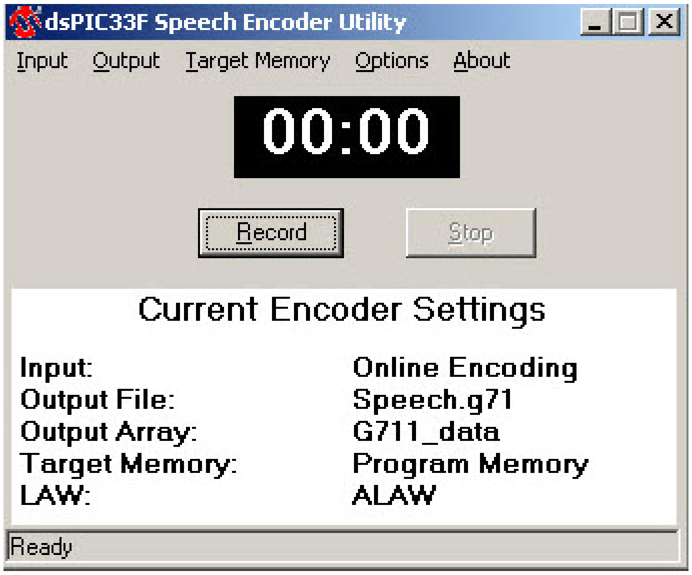
9. Click **Record**. The recording of the microphone signal starts and the user can talk into the microphone. The signals will be encoded using the G.726A encoder. The timer on the PCEU interface will start counting the length of the speech segment.
10. Click **Stop** to stop the recording. The encoding process is complete and the generated files are available for inclusion into a dsPIC DSC application. Refer to **5.3 "Output Files"** for more information on the generated file.

5.2 WAVE FILE

This section provides the steps to use the dsPIC33F Speech Encoding Utility, to encode a WAVE (`.wav`) file. Note that the `.wav` file should be in the 16-bit at 8 kHz format.

1. Double-click `dsPICSpeechRecord.exe` in the `PCEU` folder of the library installation directory to start the utility.

FIGURE 5-8: dsPIC33F SPEECH ENCODER UTILITY

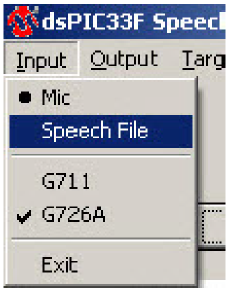


Current Encoder Settings

Input:	Online Encoding
Output File:	Speech.g71
Output Array:	G711_data
Target Memory:	Program Memory
LAW:	ALAW

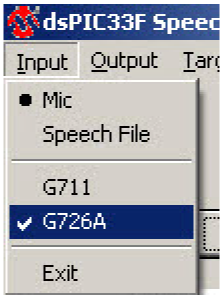
2. Select *Input > Speech File* to select a `.wav` file as the source input for the utility.

FIGURE 5-9: INPUT SOURCE SELECTION



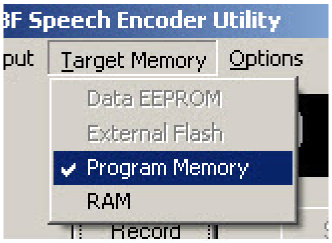
3. Select *Input > G726A* to select the G.726A encoder, to encode the microphone signal.

FIGURE 5-10: ENCODER SELECTION



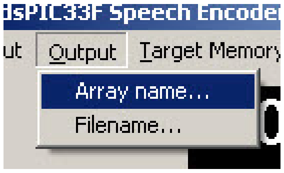
4. Select either *Target Memory > Program Memory* to store the encoded speech in Flash program memory, or *Target Memory > RAM* to store the encoded speech in RAM. The generated encoded samples can be either stored in device RAM or in Flash program memory. Depending on the selected option, the PCEU either generates a `.c` file for RAM, or a `.s` file for Flash program memory.

FIGURE 5-11: TARGET MEMORY SELECTION



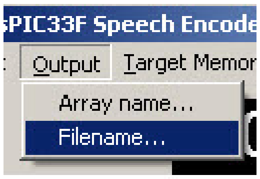
5. Select *Output > Array name* to specify the name of the encoded samples array in the generated file.

FIGURE 5-12: ARRAY NAME SELECTION



6. Select *Output > Filename* to specify the name of the generated file and the path where the generated file will be stored.

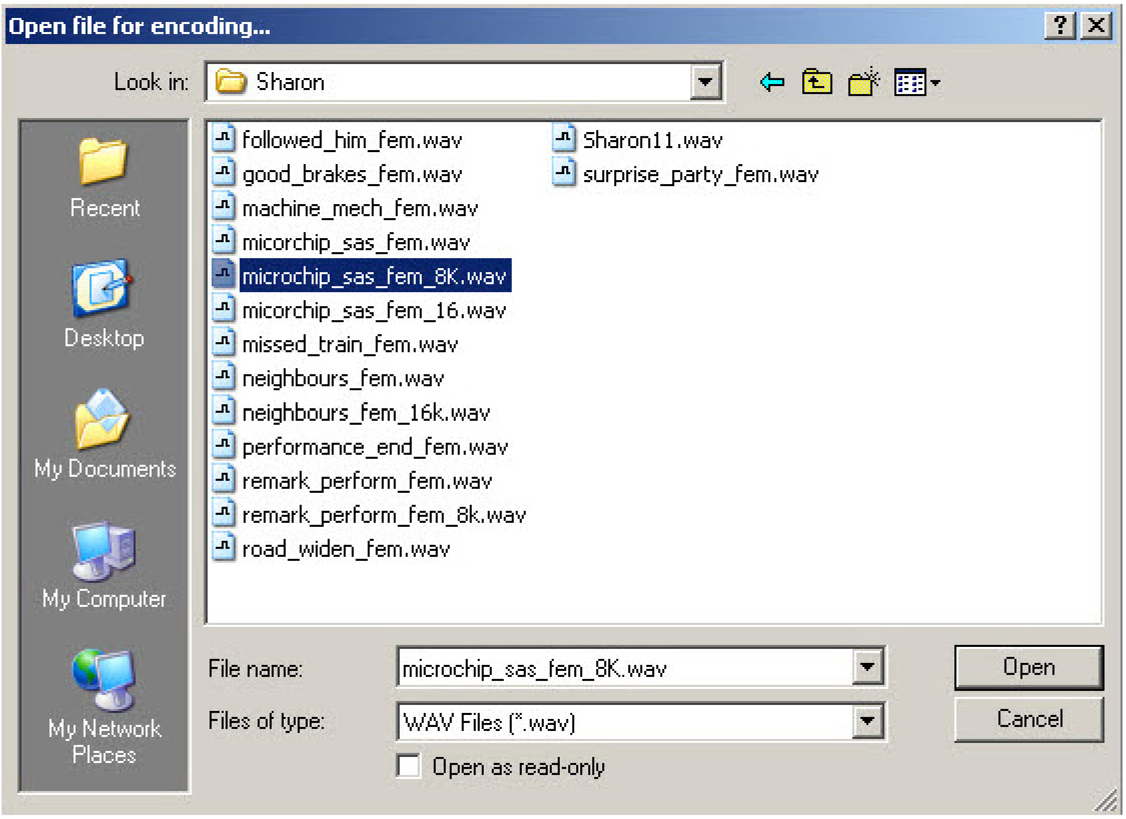
FIGURE 5-13: FILE NAME SELECTION



7. Select *Options > Bit Rates* to specify the G.726A bit rate to be used while generating the encoded samples.

8. Click **Encode**. The Open file for encoding dialog appears.

FIGURE 5-14: WAVE FILE SELECTION



Look in: Sharon

File name: microchip_sas_fem_8K.wav

Files of type: WAV Files (*.wav)

☐ Open as read-only

9. Select the `.wav` file to be encoded, and then click **Open**.
10. The PCEU encodes the file and the generated files are available for inclusion into a dsPIC DSC application. Refer to **5.3 “Output Files”** for more information on the generated file.

5.3 OUTPUT FILES

The dsPIC33F Speech Encoding Utility generates the following files:

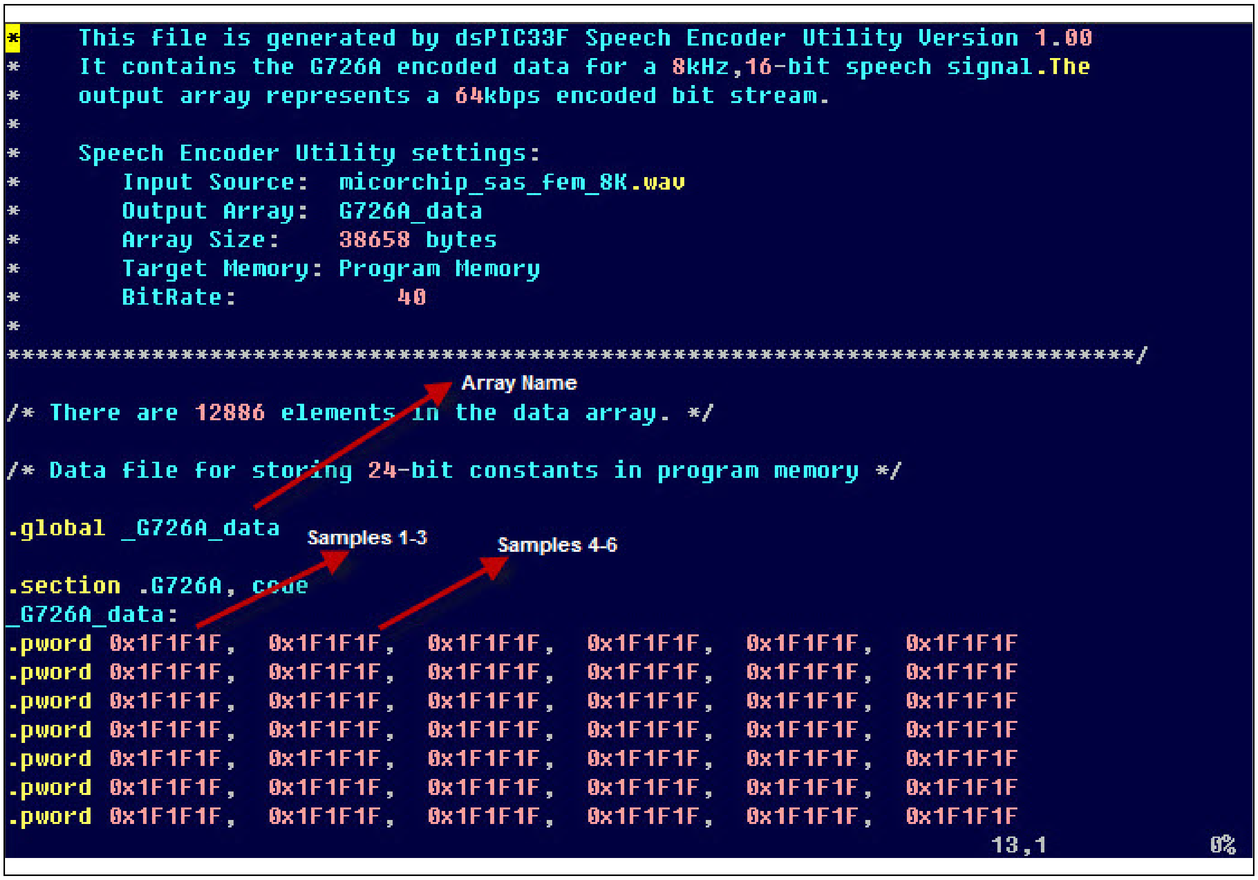
TABLE 5-1: FILES GENERATED BY SPEECH ENCODING UTILITY

File Type	Description
`.g72`	File containing G.726A encoded samples.
`.s`	This file contains the encoded samples that are generated, if the target memory is Flash program memory. This file can be included directly into the application MPLAB project.
`.c`	This file contains the encoded samples that are generated, if the target memory is Data RAM. This file can be included directly into the application MPLAB project.
`.raw`	Contains raw audio samples.

5.3.1 Encoded Samples Stored in Flash Program Memory

If the Flash program memory is selected as the target memory for the encoded samples, the PCEU generates a `.s` file. Figure 5-15 shows the snippet of the generated file.

FIGURE 5-15: GENERATED OUTPUT FILE FOR FLASH PROGRAM MEMORY



```
*    This file is generated by dsPIC33F Speech Encoder Utility Version 1.00
*    It contains the G726A encoded data for a 8kHz,16-bit speech signal.The
*    output array represents a 64kbps encoded bit stream.
*
*    Speech Encoder Utility settings:
*       Input Source:  micorchip_sas_fem_8K.wav
*       Output Array:  G726A_data
*       Array Size:    38658 bytes
*       Target Memory: Program Memory
*       BitRate:           40
*
*****************************************************************************/

/* There are 12886 elements in the data array. */

/* Data file for storing 24-bit constants in program memory */

.global _G726A_data

.section .G726A, code
_G726A_data:
.pword 0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F
.pword 0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F
.pword 0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F
.pword 0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F
.pword 0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F
.pword 0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F
.pword 0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F,  0x1F1F1F
                                                      13,1            0%
```

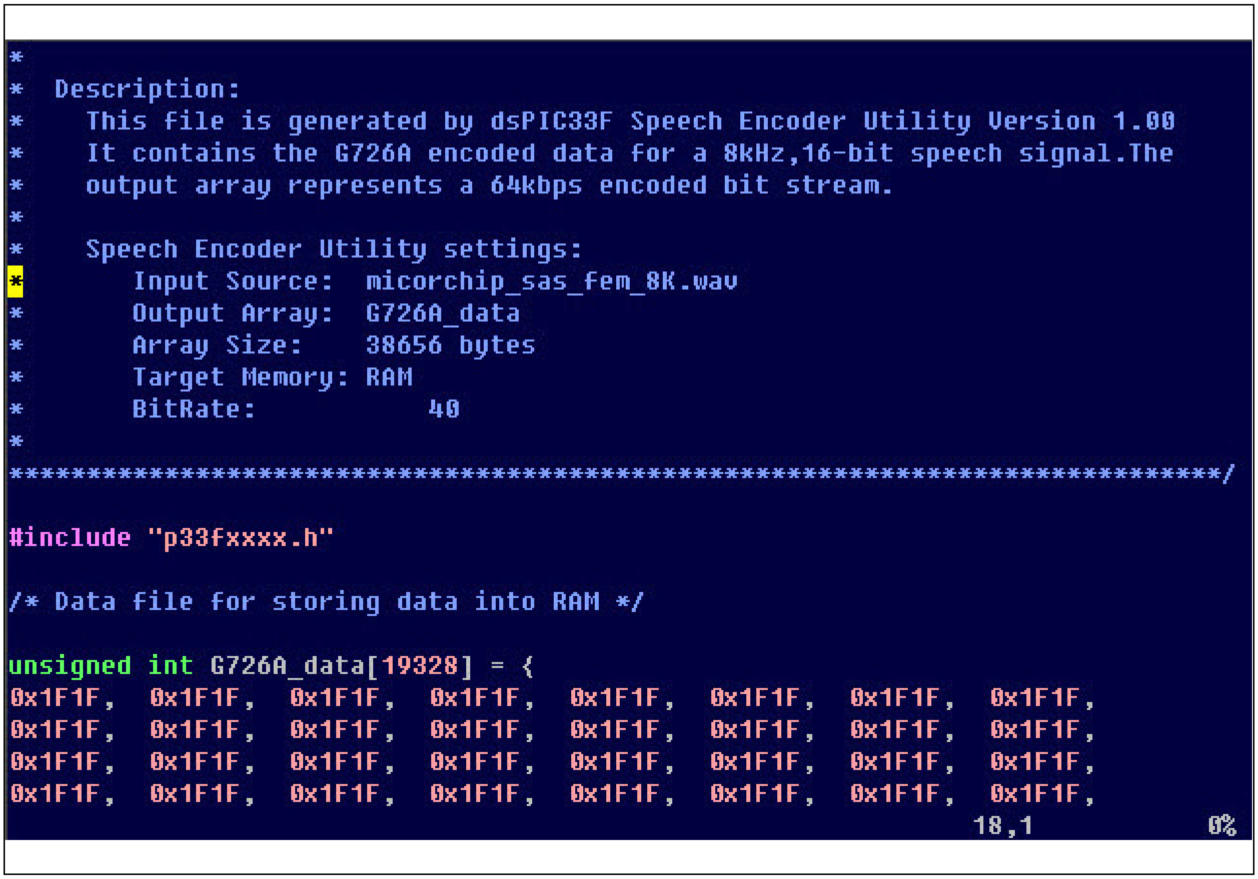
In the example shown in Figure 5-15, the encoded samples array is called `G726A_data` (the default array name used by the PCEU). The encoded samples are stored in 24-bit Flash program memory in the Little Endian byte order (first byte at lowest memory). Each encoded sample is allocated with one byte. Therefore, each Flash program memory word accommodates three encoded samples. The number of significant bits within the byte depends on the selected bit rate. Note that since all the three bytes of the Flash program memory are used for storing the encoded samples, the application program must use the `Table Read` instructions (as opposed to Program Space Visibility (PSV) technique) to read the Flash program memory.

5.3.2 Encoded Samples stored in Data RAM

If RAM is selected as the target memory for the encoded samples, the PCEU generates a `.c` file which contains an array of the encoded samples. Figure 5-16 shows a snippet of the generated file.

In the example shown in Figure 5-16, the encoded sample array is called `G726A_data` (the default array name used by the PCEU). The type of the array is an unsigned integer. Each array element stores two encoded samples. The number of significant bits within the sample depend on the selected encoding bit rate.

FIGURE 5-16: GENERATED OUTPUT FILE FOR DATA RAM



```
*
*  Description:
*    This file is generated by dsPIC33F Speech Encoder Utility Version 1.00
*    It contains the G726A encoded data for a 8kHz,16-bit speech signal.The
*    output array represents a 64kbps encoded bit stream.
*
*    Speech Encoder Utility settings:
*       Input Source:  micorchip_sas_fem_8K.wav
*       Output Array:  G726A_data
*       Array Size:    38656 bytes
*       Target Memory: RAM
*       BitRate:           40
*
******************************************************************************/

#include "p33fxxxx.h"

/* Data file for storing data into RAM */

unsigned int G726A_data[19328] = {
0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,
0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,
0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,
0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,  0x1F1F,
                                                         18,1          0%
```

Index

A

Adaptive Differential Pulse Code Modulation........... 12
API 5

C

Communication 16
Conventions 6
Customer Notification Service.................................... 8

D

demos Folder 16
Document Layout.. 5
Documentation
  Conventions... 6
  Layout 5

I

Installation
  Installing the Library... 15
Internet Address.. 8

L

Library Archive Folder... 19
Library Files 15

M

Microchip Internet Web Site 8

O

Optional Hardware
  Audio PICtail Plus Development Board 21
  Explorer 16 Development Board....................... 21

P

PCEU 19
Playback Folder 17

R

Reading, Recommended 7
Readme... 7
RecordPlay Folder 18

S

Speech Signal Compression................................... 11
System Requirements... 14

W

Warranty Registration 6

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3180
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-213-7830
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

05/02/11