dsPIC® DSC Speex Speech Encoding/Decoding Library User's Guide

ISBN: 978-1-61341-297-8

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

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

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 on-line help. Select the Help menu, and then Topics to open a list of available on-line help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the dsPIC® DSC Speex Speech Encoding/Decoding Library User's Guide. Items discussed in this preface include:

- Document Layout
- Conventions Used in this Guide
- Warranty Registration
- 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 dsPIC DSC Speex Speech Encoding/Decoding Library as a development tool to emulate and debug firmware on a target board. The document layout is as follows:

- **Chapter 1. "Introduction"**– This chapter describes the dsPIC DSC Speex Speech Encoding/Decoding Library.
- **Chapter 2. "Installation"** – This chapter provides detailed information needed to install the dsPIC DSC Speex Speech Encoding/Decoding Library on a PC.
- **Chapter 3. "Quick Start Demonstration"** – This chapter describes the demonstrations available for the dsPIC DSC Speex Speech Encoding/Decoding Library.
- **Chapter 4. "Application Programming Interface"** – This chapter outlines how the API functions provided in the dsPIC DSC Speex Speech Encoding/Decoding Library can be included in your application software through the Application Programming Interface.

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
N'Rnnnn	A number in Verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
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	`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 dsPIC DSC Speex Speech Encoding/Decoding 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 www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, 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 listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, 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 www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **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 our Web site at: http://www.microchip.com/support.

DOCUMENT REVISION HISTORY

Revision A (June 2008)

This is the initial release of this document.

Revision B (February 2011)

This revision contains the following updates:

Updated the last paragraph in **2.2.1 "demos Folder"**

- Updated the following in **3.1.1.4 "Demonstration Code Description"**:
  - Renamed the function `UART2Init()` to `UART2_open`
  - Renamed the function `UART2IsReadBusy()` to `UART2Read()`
  - Renamed the function `UART2IsWriteBusy()` to `UART2Write()`
- Updated the Return Value description for the **"Speex8KHzDecode"** API function, in **4.8 "API Functions"**
- Formatting changes and minor text updates were incorporated throughout the document

Revision C (June 2011)

In addition to new content in support of the dsPIC33E device family, this revision includes formatting changes and minor text updates, which were incorporated throughout the document.

All major documentation updates are described in Table 1-1.

TABLE 1-1: DOCUMENTATION UPDATES

Chapter	Description
"Preface"	Updated the Recommended Reading section. Updated the Development Systems Customer Change Notification Service section.
Chapter 1. "Introduction"	Updated the first paragraph in **Chapter 1. "Introduction"**. Updated **1.1 "Library Overview"**. Updated Figure 1-2 in **1.1 "Library Overview"**.
Chapter 2. "Installation"	Updated **2.1 "Installation Procedure"**. Removed **Figure 2-1** through **Figure 2-5** in **2.1 "Installation Procedure"**. Updated the version number in `Speex v3.0` to `Speex v4.0`, in **2.2 "Library Files"**. Updated **2.2.1 "demos Folder"**. Updated Table 2-1 in **2.2.1.1 "communication Folder"**. Updated Table 2-2 in **2.2.1.2 "`Playback` Folder"**. Updated Table 2-3 in **2.2.1.3 "`Recordplay` Folder"**. Renamed the title for Table 2-5 and updated the table, in **2.2.4 "`libs` Folder"**.
Chapter 3. "Quick Start Demonstration"	Updated the existing section in **Chapter 3. "Quick Start Demonstration"** to **3.1 "Quick Start Demonstration for dsPIC33F Device Family"**. Updated the first paragraph in **3.1.1 "Communication Demonstration"**. Updated step 1 and step 2 in **3.1.1.2.2 "Programming the dsPIC DSC Device"**. Updated step 1 and step 2 in **3.1.2.2.2 "Programming the dsPIC DSC Device"**. Updated step 2 in **3.1.2.3 "Demonstration Procedure"**. Updated step 1 and step 2 in **3.1.3.2.2 "Programming the dsPIC DSC Device"**. Added **3.2 "Quick Start Demonstration for dsPIC33E Device Family"**. Updated step 7 and removed step 8 in **3.2.1.2.1 "Configure MEBs and dsPIC33E USB Starter Kits"**. Removed **Figure 3-6** in **3.2.1.2.1 "Configure MEBs and dsPIC33E USB Starter Kits"**.

TABLE 1-1: DOCUMENTATION UPDATES (CONTINUED)

Chapter	Description
Chapter 4. "Application Programming Interface"	Updated **4.1 "Adding the Library to an Application"**. Removed **Figure 4-1** through **Figure 4-3** in **4.1 "Adding the Library to an Application"**. Removed **Figure 4-1** in **4.2 "Memory Model Compile Options"**. Updated **4.3.1 "Narrowband Speex Coding Steps"**. Added a note in **4.3.1 "Narrowband Speex Coding Steps"**. Updated **4.3.2 "Wideband Speex Coding Steps"**. Added a note in **4.3.2 "Wideband Speex Coding Steps"**. Updated **4.7 "Resource Requirements"**. Updated Table 4-2, Table 4-5 and Table 4-6 in **4.7.1 "Narrowband-Only Library"**. Updated Table 4-8 and Table 4-10 in **4.7.2 "Wideband and Narrowband Speex"**.

Chapter 1. Introduction

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

This user's guide provides information that you can use to incorporate the dsPIC DSC Speex Speech Encoding/Decoding Library into your embedded solution.

The following topics are covered in this chapter:

- Library Overview
- Features
- System Requirements

1.1 LIBRARY OVERVIEW

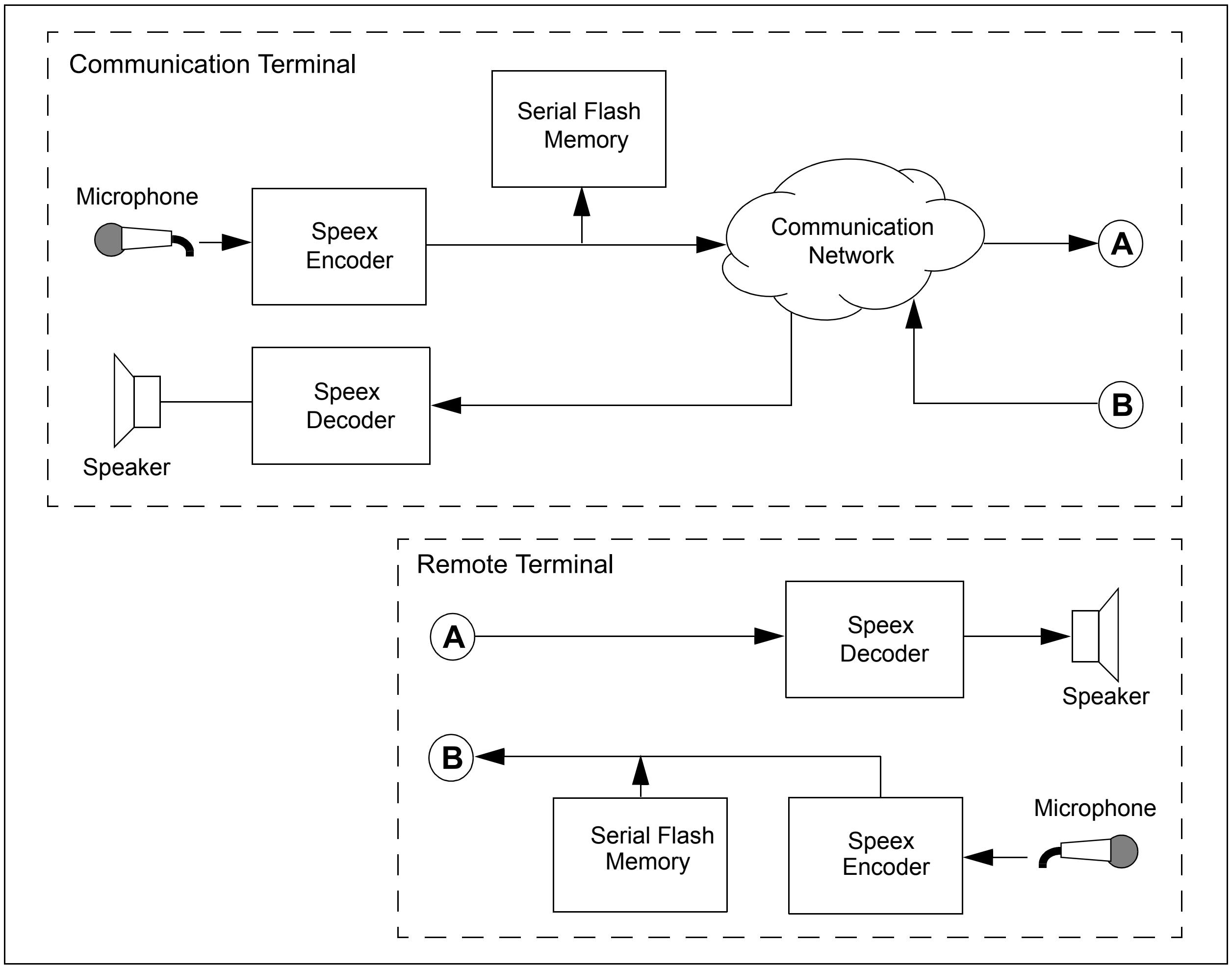
The dsPIC DSC Speex Speech Encoding/Decoding Library reduces the number of bytes required to represent a speech frame. This reduction, or compression of speech data, is specified by the compression ratio. Figure 1-1 illustrates a typical application of the dsPIC DSC Speex Speech Encoding/Decoding Library.

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

The dsPIC DSC Speex Speech Encoding/Decoding Library operates at a sampling rate of either 16 kHz or 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

FIGURE 1-1: APPLICATION EXAMPLE



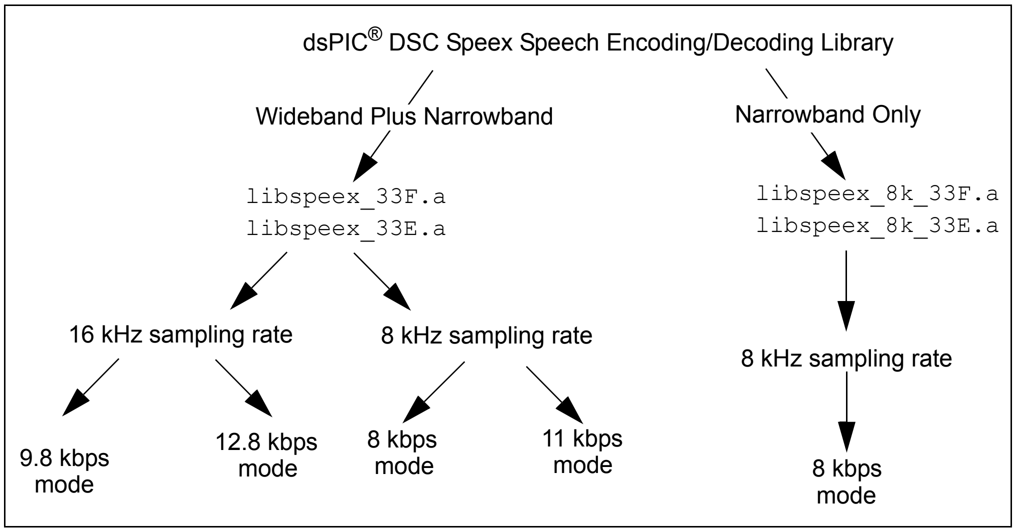
The dsPIC DSC Speex Speech Encoding/Decoding Library is based on the Speex Speech Coder, which is an open-source speech compression algorithm and uses the Code Excited Linear Prediction (CELP) technique. This technique provides a reasonable trade-off between performance and computational complexity. The library is appropriate for both half-duplex and full-duplex systems. The library is written in both C language and the Assembly language.

The dsPIC DSC Speex Speech Encoding/Decoding Library supports four output data rates (modes), two of which require a 16 kHz sampling rate (Wideband). The remaining two are supported at an 8 kHz sampling rate (Narrowband). Additionally, the library also contains a version of the Speex coder, which is optimized for the 8 kbps mode.

The optimized library can be used on devices with 8 KB RAM. Figure 1-2 illustrates the organization of the dsPIC DSC Speex Speech Encoding/Decoding Library.

The library archive file, `libspeex_33F.a` for dsPIC33F devices and `libspeex_33E.a` for dsPIC33E devices, supports all the four modes (Narrowband and Wideband). The library archive file, `libspeex_8k_33F.a` for dsPIC33F devices and `libspeex_8k_33E.a` for dsPIC33E devices, is a Narrowband-only version of Speex and is optimized for the 8 kbps mode.

FIGURE 1-2: dsPIC® DSC SPEEX SPEECH ENCODING/DECODING LIBRARY ORGANIZATION



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 dsPIC DSC Speex Speech Encoding/Decoding 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 Speex 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 WAVE (`.wav`) files, and can accept multiple files as input.

1.2 FEATURES

The dsPIC DSC Speex Speech Encoding/Decoding Library has the following features:

- Simple user interface with 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 output modes:
  - 8 kbps at 8 kHz sampling rate (Narrowband) – 16:1 compression ratio
  - 11 kbps at 8 kHz sampling rate (Narrowband) – 11.63:1 compression ratio
  - 9.8 kbps at 16 kHz sampling rate (Wideband) – 26.12:1 compression ratio
  - 12.8 kbps at 16 kHz sampling rate (Wideband) – 20:1 compression ratio
- Library contains a Speex version optimized for 8 kbps mode
- Speex PCEU for recording and encoding speech through a PC
- Audio bandwidth of 0 kHz to 8 kHz with sampling rate of up to 16 kHz
- Can be integrated with Microchip's Noise Suppression, Acoustic Echo Cancellation and Line Echo Cancellation Libraries
- Library delivered on CD, which includes:
  - Sample demonstration applications with complete source code
  - User's guide
  - HTML help files for PCEU

1.3 SYSTEM REQUIREMENTS

The dsPIC DSC Speex Speech Encoding/Decoding Library 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 dsPIC DSC Speex Speech Encoding/Decoding 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 dsPIC DSC Speex Speech Encoding/Decoding 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 `Speex setup.exe`. The license agreement appears in a new window.
2. Review the license agreement and click **I Agree** to continue. The Installation Destination dialog appears.
3. Specify the location (i.e., directory) where the library is to be installed, and then click **Install**.
4. Click **Close** to close the dialog. This completes the dsPIC DSC Speex Speech Encoding/Decoding Library installation.

The installation process creates the folder, `Speex v4.0`, which contains the files described in **2.2 "Library Files"**.

2.2 LIBRARY FILES

The dsPIC DSC Speex Speech Encoding/Decoding Library CD or zip archive file creates a directory titled `Speex v4.0`.

This directory contains the following folders:

- `demos`
- `doc`
- `h`
- `libs`
- `PCEU`

2.2.1 `demos` Folder

The `demos` folder contains application code examples, which demonstrate the use of the Narrowband and Wideband modes of Speex in different application scenarios. The `demos` folder contains the following folders and sub-folders:

- `Narrowband`
  - `Communication`
  - `Playback`
  - `RecordPlay`
- `Wideband`
  - `Communication`
  - `Playback`
  - `RecordPlay`

The only difference between the Narrowband and the Wideband demonstrations is the Speex mode that is utilized. The archive file used for Narrowband demonstrations is `libspeex_8k_33F.a` for dsPIC33F or `libspeex_8k_33E.a` for dsPIC33E, and the include file is `speex_8k.h`. The archive file used for Wideband demonstrations is `libspeex_33F.a` for dsPIC33F or `libspeex_33E.a` for dsPIC33E, and the include file is `speex.h`. The demonstration folder also contains a playback application example for dsPIC30F devices.

2.2.1.1 `Communication` FOLDER

The `Communication` folder contains files, which demonstrate the use of the dsPIC DSC Speex Speech Encoding/Decoding 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

File Name	Description
`dsPIC33F Communication Demo 1.hex`	Demonstration hexadecimal file for board 1 on dsPIC33F.
`dsPIC33F Communication Demo 2.hex`	Demonstration hexadecimal file for board 2 on dsPIC33F.
`dsPIC33E Communication Demo 1.hex`	Demonstration hexadecimal file for board 1 on dsPIC33E.
`dsPIC33E Communication Demo 2.hex`	Demonstration hexadecimal file for board 2 on dsPIC33E.
`dsPIC33F Communication Demo.mcp`	Demonstration MPLAB Project file for dsPIC33F.
`dsPIC33E Communication Demo.mcp`	Demonstration MPLAB Project file for dsPIC33E.
`cleanup.bat`	A batch file script for cleaning the 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\speex_8k.h`	C header file defining the interface to Narrowband Speex 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\libspeex_8k_33F.a`	Narrowband Speex Library archive file for dsPIC33F.
`lib\libspeex_8k_33E.a`	Narrowband Speex Library archive file for dsPIC33E.
`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\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.

2.2.1.2 `Playback` FOLDER

The `Playback` folder contains demonstration files that can be used to show the use of dsPIC DSC Speex Speech Encoding/Decoding 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
`dsPIC33F Playback Demo.hex`	Demonstration hexadecimal file for dsPIC33F.
`dsPIC33E Playback Demo.hex`	Demonstration hexadecimal file for dsPIC33E.
`dsPIC33F Playback Demo.mcp`	Demonstration MPLAB Project file for dsPIC33F.
`dsPIC33E Playback Demo.mcp`	Demonstration MPLAB Project file for dsPIC33E.
`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\speex_8k.h`	C header file defining the interface to the Narrowband Speex Library.
`h\PgmMemory.h`	C header file defining the interface to routines to read Flash program 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\libspeex_8k_33F.a`	Narrowband Speex Library archive file for dsPIC33F.
`lib\libspeex_8k_33E.a`	Narrowband Speex Library archive file for dsPIC33E.
`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\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.

2.2.1.3 Recordplay FOLDER

The RecordPlay demonstration shows the use of the dsPIC DSC Speex Speech Encoding/Decoding Library to implement a voice recorder system. The demonstration will encode speech data, store it in external serial Flash memory, and will playback this data when requested. Table 2-3 describes the files in this folder.

TABLE 2-3: RecordPlay FOLDER FILES

File Name	Description
dsPIC33F RecordPlay Demo.hex	Demonstration hexadecimal file for dsPIC33F.
dsPIC33E RecordPlay Demo.hex	Demonstration hexadecimal file for dsPIC33E.
dsPIC33F RecordPlay Demo.mcp	Demonstration MPLAB Project file for dsPIC33F.
dsPIC33E RecordPlay Demo.mcp	Demonstration MPLAB workspace for dsPIC33E.
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 the MEB.
h\speex_8k.h	C header file defining the interface to the Narrowband Speex 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\WM87310CodecDrv.h	C header file defining the interface to the WM8731 codec driver.
lib\libspeex_8k_33F.a	Narrowband Speex library archive file for dsPIC33F.
lib\libspeex_8k_33E.a	Narrowband Speex library archive file for dsPIC33E.
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.h	C source file containing routines for SST25VF040B serial Flash memory.
src\SST25VF016BDrv.h	C source file containing 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 `doc` Folder

The `doc` folder contains the user’s guide for the dsPIC DSC Speex Speech Encoding/Decoding Library and a help file for the Speex PC-based Encoder Utility (PCEU). To view either 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 API. Table 2-4 describes the files in this folder.

TABLE 2-4: INCLUDE FILES

File Name	Description
`libspeex.h`	Include file that contains the interface to Wideband mode and Narrowband Speex mode. This file must be included in the application to use any of the four modes.
`libspeex_8k.h`	Include file that contains the interface to the optimized Narrowband Speex mode. This file must be included in the application to use the single Narrowband mode.

2.2.4 `libs` Folder

The `libs` folder contains the library archive files for the dsPIC DSC Speex Speech Encoding/Decoding Library. Table 2-5 describes the files in this folder.

TABLE 2-5: LIBRARY ARCHIVE FILES

File Name	Description
`libspeex_33F.a`	Archive file that contains Wideband and Narrowband Speex functions. This file must be included in the application to use any of the four modes.
`libspeex_33E.a`	
`libspeex_8k_33F.a`	Archive file that contains optimized Narrowband Speex functions. This file must be included in the application to use the single Narrowband mode.
`libspeex_8k_33E.a`	

2.2.5 `PCEU` Folder

The `PCEU` folder contains files required by Speex 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.
`Speech_WB.dll`	DLL files required by the PCEU.
`SpeechRecord.dll`	

Chapter 3. Quick Start Demonstration

This chapter describes the demonstration code examples for the dsPIC33F and dsPIC33E device families, which are part of the dsPIC DSC Speex Speech Encoding/Decoding 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

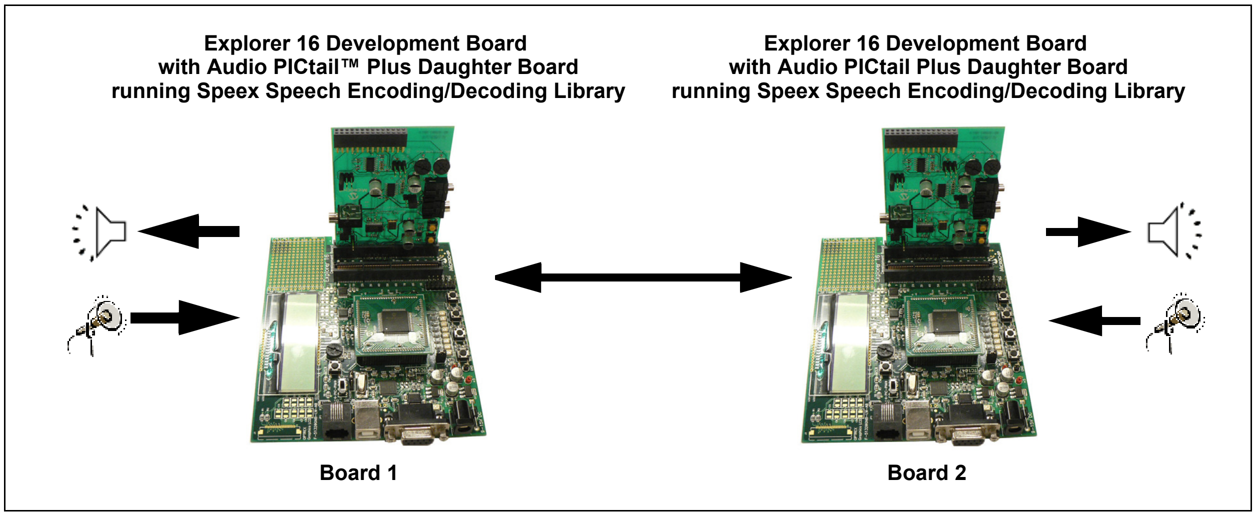
3.1.1 Communication Demonstration

The Communication demonstration shows how the dsPIC DSC Speex Speech Encoding/Decoding 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: SPEEX COMMUNICATION DEMONSTRATION SETUP


Explorer 16 Development Board
with Audio PICtail™ Plus Daughter Board
running Speex Speech Encoding/Decoding Library
Explorer 16 Development Board
with Audio PICtail Plus Daughter Board
running Speex 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 Speex encoder, and then transmit the compressed speech frame data through the UART2 module and the 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 Speex 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 a Narrowband Speex mode, for example, for 8 kbps mode, the minimum communication link bandwidth would be 8 kbps. This results in a bandwidth reduction of 16x.

The code examples invoke the `Speex16KHzEncode()` and `Speex16KHzDecode()` functions for Wideband, or `Speex8KHzEncode()` or `Speex8KHzDecode()` function for Narrowband from the dsPIC DSC Speex Speech Encoding/Decoding Library, to encode and decode speech data.

3.1.1.2 DEMONSTRATION SETUP

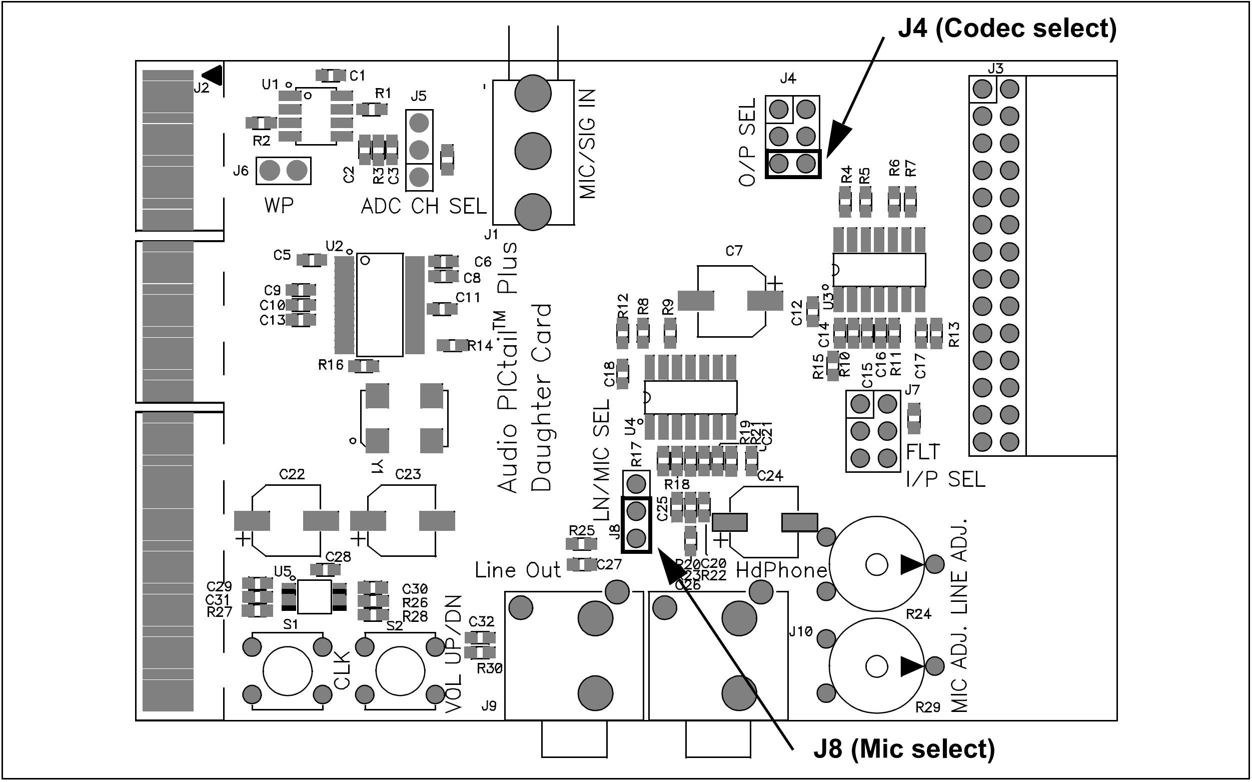
The demonstration application is intended to run on the Explorer 16 Development Board with an Audio PICtail Plus Daughter Board (not included with the software license). The procedure described in this section applies to both the Narrowband and the Wideband Speex communication demonstration. Use the procedure outlined 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:

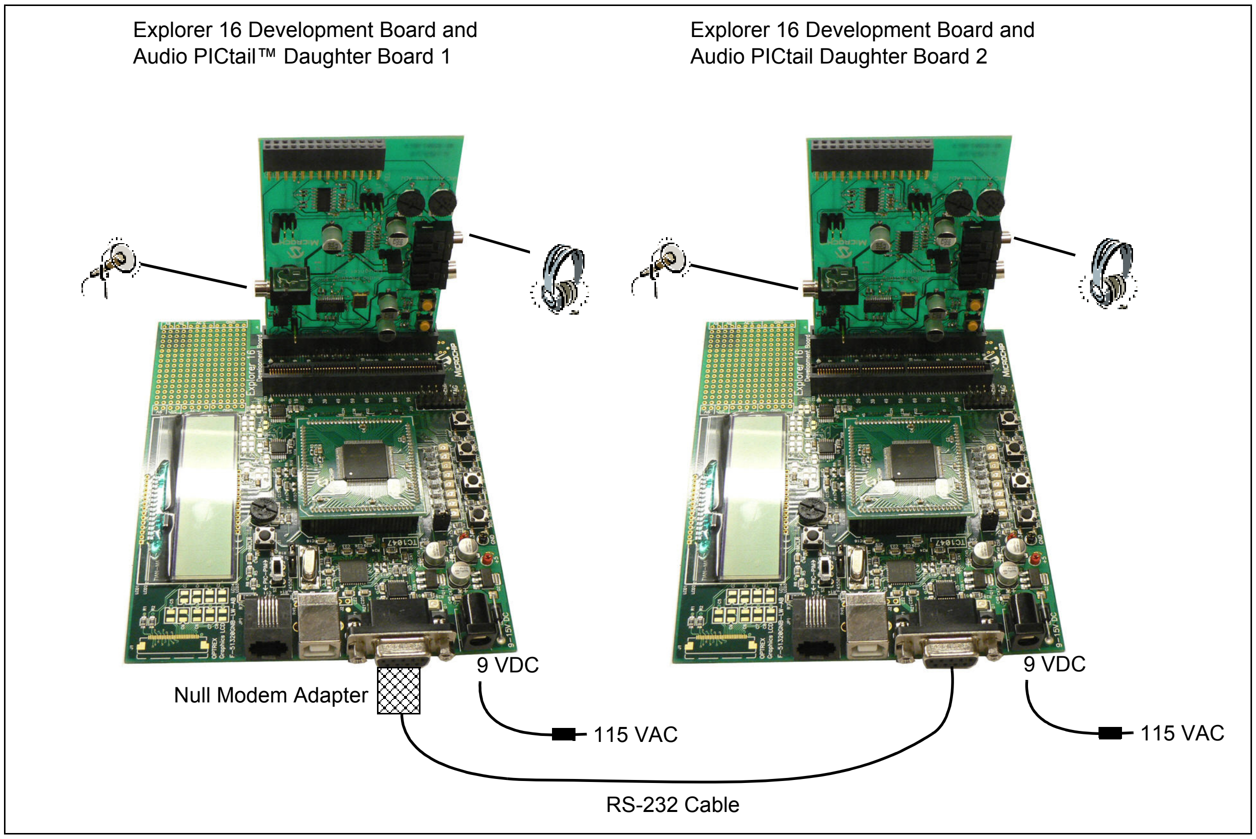
1. On the Audio PICtail Plus Daughter Board 1, set jumper J4 (O/P SEL) to the codec position and set jumper J8 (LN/MIC) to the microphone position, as shown in Figure 3-2.

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 through 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 will be similar to the setup 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
9 VDC
Null Modem Adapter
115 VAC
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 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 *File > Import > dsPIC33F Communication Demo 1.hex* to import the project hexadecimal file for board 1, or select *File > Import > dsPIC33F Communication Demo 2.hex* to import the project hexadecimal file for board 2.
3. Select *Programmer > Connect* to link the MPLAB ICD 2 to the dsPIC DSC target device. The Output window confirms that the MPLAB ICD 3 is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming was successful.
5. Connect the 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 the MPLAB ICD 3 from the board (remove the phone cable from the MPLAB ICD 3 connector).
8. Repeat steps 2 through 7 for the second board.

Note: The MPLAB® REAL ICE™ In-Circuit Emulator can be used in place of MPLAB ICD 3.

3.1.1.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into both devices, the application is now 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. Reset board 2. At this time, LED D4 on both of 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 dsPIC DSC Speex Speech Encoding/Decoding Library functions.

The main function calls the `Speex8KHzEncoderInit()` and `Speex8KHzDecoderInit()` functions (or the corresponding Wideband functions for the Wideband communication demonstration) from the dsPIC DSC Speex Speech Encoding/Decoding Library, which initializes the Speex 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^2C™ 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 WM8510 codec 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 I^2C module of the dsPIC DSC device. Subsequently, the `WM8510Start()` function is used to enable the DCI module and I^2C module. The `WM8510SampleRate8KConfig()` function configures the WM8510 codec for a sampling rate of 8 kHz. The Wideband communication demonstration calls the `WM8510SampleRate16KConfig()` function to configure the WM8510 codec for a sampling rate of 16 kHz.

The `UART2_open` function configures the UART2 module on the dsPIC DSC for operation at the baud rate specified in `UART2Drv.h`. It also enables the UART2 module for transmission and reception. The driver provides a synchronous interface through the `UART2Read()` and `UART2Write()` functions. The main processing loop uses these functions 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 `Speex8KHzDecode()` 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 `Speex8KHzEncode()` function is called. The encoded frame is written to the UART2 driver for transmission.

3.1.2 Playback Demonstration

The Playback demonstration shows how the dsPIC DSC Speex Speech Encoding/Decoding Library can be used in a playback type of application where the encoded frames are read from Flash program memory, decoded by the Speex decoder, and then played out. By storing the Speex encoded speech frames in Flash program memory, the application can minimize the 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 Speex encoded frames, which are stored in Flash program memory. Every frame is then decoded using the Speex 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 Speex PC-based Encoder Utility (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 `Speex16KHzDecode()` or `Speex8KHzDecode()` function from the dsPIC DSC Speex Speech Encoding/Decoding 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). The procedure described in this section applies to both the Narrowband and the Wideband Speex Playback demonstration.

Use the procedure outlined 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 an 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 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 Playback Demo.hex* to import the project hexadecimal file.
3. Select *Programmer > Connect* to link the MPLAB ICD 3 to the dsPIC DSC target device. The Output window confirms that the MPLAB ICD 3 is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming was successful.
5. Connect the 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 the MPLAB ICD 3 from the board (remove the telephone cable from the MPLAB ICD 3 connector).

Note: The MPLAB REAL ICE can be used in place of MPLAB ICD 3.

3.1.2.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into the device, the application is now ready to run. Use the following procedure to run the demonstration.

1. Reset the board. LED D3 on the board will light up indicating that the application is running.
2. Listen to the 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 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 dsPIC DSC Speex Speech Encoding/Decoding Library functions.

The main function calls the `Speex8KHzDecoderInit()` function (or the corresponding Wideband functions for the Wideband Playback demonstration) from the dsPIC DSC Speex Speech Encoding/Decoding Library, which initializes the Speex decoder algorithm to its 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 WM8510 codec 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()` function configures the WM8510 codec for a sampling rate of 8 kHz. The Wideband communication demonstration calls the `WM8510SampleRate16KConfig()` function to configure the WM8510 codec for a sampling rate of 16 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 `PgmMemRead()` and passes it to the `Speex8KHzDecode()` function for decoding. The decoded speech frame is written to the WM8510 codec driver for playback.

3.1.3 Record Play Demonstration

The Speex Record Play demonstration shows the use of the dsPIC DSC Speex Speech Encoding/Decoding 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.

3.1.3.1 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 Speex encoder and will then store it in the serial Flash memory available on the Audio PICtail Plus Daughter Board. In Playback mode, the application reads the encoded speech frames stored in serial Flash data and then decodes these frames using the Speex decoder. The decoded speech data is written to the WM8510 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 a Narrowband Speex mode, such as 8 kbps mode, the memory required to store one minute of speech is 60 KB.

The code examples invoke the `Speex16KHzEncode()` and `Speex16KHzDecode()` functions (for Wideband), or the `Speex8KHzEncode()`, or the `Speex8KHzDecode()` function (for Narrowband) from dsPIC DSC Speex Speech Encoding/Decoding Library to encode and decode speech data.

3.1.3.2 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). The procedure described in this section applies to both the Narrowband and the Wideband Speex Record Play demonstration.

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

3.1.3.2.1 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 (see Figure 3-2).
2. Insert the Audio PICtail Plus Daughter Board into the Explorer 16 Development Board PICtail Plus socket J5.
3. Connect the 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.

3.1.3.2.2 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, start MPLAB IDE and open the `dsPIC33F 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 Record Play Demo.hex* to import the project file.
3. Select *Programmer > Connect* to link the MPLAB ICD 3 to the target dsPIC DSC device. The Output window shows that the MPLAB ICD 3 is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming was successful.
5. Connect the 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 the MPLAB ICD 3 from the board (remove the telephone cable from the MPLAB ICD 3 connector).

Note: The MPLAB REAL ICE can be used in place of MPLAB ICD 3.

3.1.3.3 DEMONSTRATION PROCEDURE

After the demonstration application has been programmed into the device, the application is now ready to run. Use the following procedure to run the demonstration

1. Reset 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 D7 will switch on indicating that 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 D7 turns OFF and LED D8 lights up.
4. To start recording again, press switch S3.

3.1.3.4 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 of the variables and arrays in data memory that are needed by the WM8510 codec driver and the SST25VF040B serial Flash memory driver, as well as the blocks of data memory that need to be allocated for the dsPIC DSC Speex Speech Encoding/Decoding Library functions.

The main function calls the `Speex8KHzEncoderInit()` and `Speex8KHzDecoderInit()` functions (or the corresponding Wideband functions for the Wideband Record Play demonstration) from the dsPIC DSC Speex Speech Encoding/Decoding Library, which initializes the Speex decoder algorithm to its 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 WM8510 codec 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 code 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()` function configures the WM8510 codec for a sampling rate of 8 kHz. The Wideband communication demonstration calls the `WM8510SampleRate16KConfig()` function to configure the WM8510 codec for a sampling rate of 16 kHz.

The `SST25VF040BInit()` function is called to initialize the serial Flash driver memory. The `SST25VF040BStart()` function enables the dsPIC DSC device’s SPI module, which communicates with the serial Flash.

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 `Speex8KHzEncode()` function and written to Flash program memory using the `SST25VF040BWrite()` function. The speech frame is also provided to the WM8510 driver for output. If the Playback mode is selected, the serial Flash memory is read using the `SST25VF040BRead()` function, decoded using the `Speex8KHzDecode()` function, and written to the WM8510 driver for output.

3.2 QUICK START DEMONSTRATION FOR dsPIC33E DEVICE FAMILY

The following demonstration examples are covered in this section:

- Communication Demonstration
- Playback Demonstration
- Record Play Demonstration

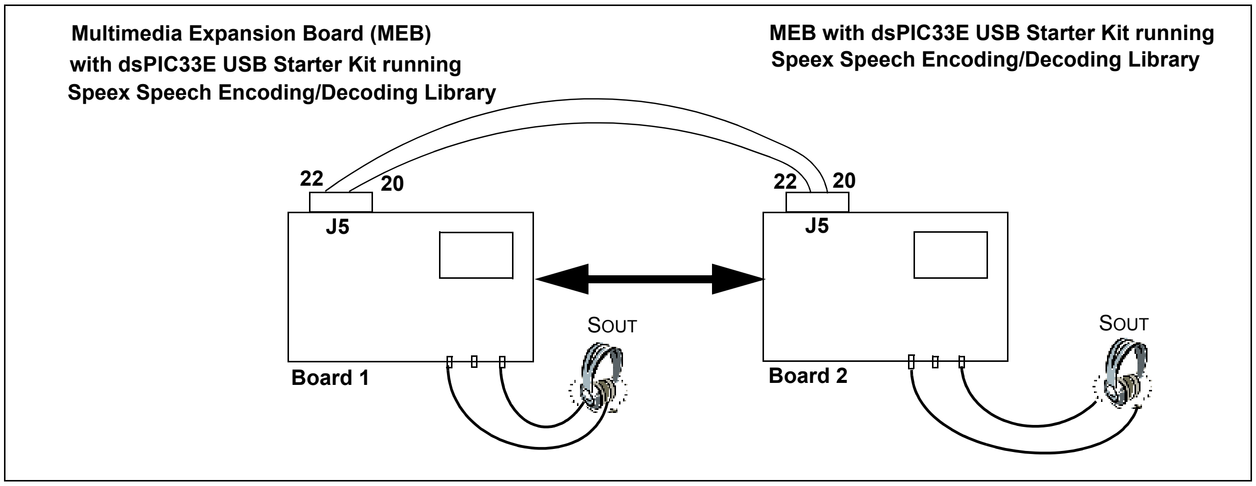
3.2.1 Communication Demonstration

The Communication demonstration shows how the dsPIC DSC Speex Speech Encoding/Decoding 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 MEBs 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: SPEEX COMMUNICATION DEMONSTRATION SETUP


Multimedia Expansion Board (MEB)
with dsPIC33E USB Starter Kit running
Speex Speech Encoding/Decoding Library
MEB with dsPIC33E USB Starter Kit running
Speex Speech Encoding/Decoding Library
22
20
J5
22
20
J5
SOUT
SOUT
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 Speex encoder and then transmit the compressed speech frame data through the UART2 module and the 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 Speex decoder, and then outputs 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 a Narrowband Speex mode, for example, for 8 kbps mode, the minimum communication link bandwidth would be 8 kbps. This results in a bandwidth reduction of 16x.

The code examples invoke the `Speex16KHzEncode()` and `Speex16KHzDecode()` `functions` (for Wideband) or `Speex8KHzEncode()` or `Speex8KHzDecode()` function (for Narrowband) from dsPIC DSC Speex Speech Encoding/Decoding Library to encode and decode speech data.

3.2.1.2 DEMONSTRATION SETUP

The demonstration application is intended to run on the MEB with a dsPIC33E USB Starter Kit (not included with the software license). The procedure described in this section applies to both the Narrowband and the Wideband Speex communication demonstration.

Use the procedure outlined 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 the 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 headphones to MEB 1.
4. Connect a microphone to MEB 1.
5. Repeat steps 1 through 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.

3.2.1.2.2 Programming the dsPIC DSC Device

Use this process to load the Speex communication demonstration into the dsPIC DSC device on the MEB.

1. On your PC, start MPLAB IDE and open the `dsPIC33F 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 *File > Import > dsPIC33E Communication Demo 1.hex* to import the project hexadecimal file for board 1, or select *File > Import > dsPIC33E 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 *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 was successful.

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 devices, the application is now ready to run. Use the following procedure to run the demonstration.

1. Using the MPLAB IDE, reset board 1. LED D1 on the board will light up indicating that the board is waiting for synchronization to be complete.
2. Using the MPLAB IDE programmer, reset board 2. LED D2 on both of 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, then program board 2. Disconnect the USB cable from board 2 after programming, and 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 by 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 dsPIC DSC Speex Speech Encoding/Decoding Library functions.

The main function calls the `Speex8KHzEncoderInit()`and `Speex8KHzDecoderInit()` functions (or the corresponding Wideband functions for the Wideband communication demonstration) from the dsPIC DSC Speex Speech Encoding/Decoding Library, which initializes the Speex encoder and decoder algorithms to their default and initial state.

The main function also calls the `WM8731Init()`function to initialize the DCI module, the I^{2}C module, the WM8731 codec and the DCI interrupt. The WM8731 codec acts as a Master and drives the serial clock and frame synchronization lines. 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, that is, only one transmit slot and one receive slot are used in this demonstration.

The dsPIC DSC device will send the control information to the WM8731 codec through the I^{2}C module of the dsPIC DSC device. Subsequently, the `WM8731Start()`function is used to enable the DCI module and I^{2}C module. The initialization code configures the WM8731 codec for a sampling rate of 8 kHz in case of Narrowband mode, and a sampling rate of 16 kHz in case of Wideband mode.

The `UART2_open` function configures the UART2 module on the dsPIC DSC for operation at the baud rate specified in `UART2Drv.h`. It also enables the UART2 module for transmission and reception. The driver provides a synchronous interface through the `UART2Read()` and `UART2Write()` functions. The main processing loop uses these functions 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 `Speex8KHzDecode()` 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 `Speex8KHzEncode()` function is called. The encoded frame is written to the UART2 driver for transmission.

3.2.2 Playback Demonstration

The Playback demonstration shows how the dsPIC DSC Speex Speech Encoding/Decoding 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 played out. By storing Speex encoded speech frames in Flash program memory, the application can minimize the 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 Audio PICtail Plus Daughter Board. The demonstration code reads the Speex encoded frames, which are stored in Flash program memory. Every frame is then decoded using the Speex decoder. The decoded speech data is written to the WM8731 codec for playback on the headphones.

The encoded speech data is obtained using the Speex 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 `Speex16KHzDecode()` or `Speex8KHzDecode()` function from the dsPIC DSC Speex Speech Encoding/Decoding 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). The procedure described in this section applies to both the Narrowband and the Wideband Speex Playback demonstration. Use the procedure outlined 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 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 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 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 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 was successful.

Note: After programming, unplug and reconnect the USB cable to the 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 now ready to run. Use the following procedure to run the demonstration.

1. Using the 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 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 Speex Speech Encoding/Decoding Library functions.

The main function calls the `Speex8KHzDecoderInit()` function (or the corresponding Wideband functions for the Wideband Playback demonstration) from the dsPIC DSC Speex Speech Encoding/Decoding Library, which initializes the Speex decoder algorithm to its default and initial state.

The main function also calls the `WM8731Init()` function to initialize the DCI module, the I^{2}C module, the WM8731 codec and the DCI interrupt. The WM8731 codec acts as a Master and drives the serial clock and frame synchronization lines. 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 (that is, only one transmit slot and one receive slot are used in this demonstration). The dsPIC DSC device will send the control information to the WM8731 codec through the dsPIC DSC device's I^{2}C module. Subsequently, the `WM8731Start()` function is used to enable the DCI module and I^{2}C module. The initialization code configures the WM8731 codec for a sampling rate of 8 kHz in case of Narrowband mode, and a sampling rate of 16 kHz in case of Wideband mode.

The `PgmMemOpen()` function opens a handle to the Flash program memory for reading. The main loop reads one frame from the Flash program memory using the `PgmMemRead()` and passes it to the `Speex8KHzDecode()` function for decoding. The decoded speech frame is written to the WM8731 codec driver for playback.

3.2.3 Record Play Demonstration

The Speex Record Play demonstration shows the use of the dsPIC DSC Speex Speech Encoding/Decoding 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 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 WM8731 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 Speex encoder and will then store it in the serial Flash memory available on the Audio PICtail Plus Daughter Board. In Playback mode, the application reads the encoded speech frames stored in serial Flash data and then decodes these frames using the Speex 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 a Narrowband Speex mode, such as 8 kbps mode, the memory required to store one minute of speech is 60 KB.

The code examples invokes the `Speex16KHzEncode()` and `Speex16KHzDecode()` functions (for Wideband) or the `Speex8KHzEncode()` or `Speex8KHzDecode()` function (for Narrowband) from dsPIC DSC Speex Speech Encoding/Decoding 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). The procedure described in this section applies to both the Narrowband and the Wideband Speex Record Play demonstration. 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 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 Record Play 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 Record Play 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 was successful.

Note: After programming, unplug and reconnect the USB cable to the 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 now 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. 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 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 and the SST25VF016B serial Flash memory driver, as well as the blocks of data memory that need to be allocated for the dsPIC DSC Speex Speech Encoding/Decoding Library functions.

The main function calls the `Speex8KHzEncoderInit()` and `Speex8KHzDecoderInit()` functions (or the corresponding Wideband functions for the Wideband Record Play demonstration) from the dsPIC DSC Speex Speech Encoding/Decoding Library, which initializes the Speex decoder algorithm to its default and initial state.

The main function also calls the `WM8731Init()`function to initialize the DCI module, the I^{2}C module, the WM8731 codec and the DCI interrupt. The WM8731 codec acts as a Master and drives the serial clock and frame synchronization lines. 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 (that is, transmit slots and two receive slots are used in this demonstration). The dsPIC DSC device will send the control information to the WM8731 codec through the dsPIC DSC device's I^{2}C module. Subsequently, the `WM8731Start()` function is used to enable the DCI module and I^{2}C module.

The initialization code configures the WM8731 codec for a sampling rate of 8 kHz in the case of Narrowband mode and a sampling rate of 16 kHz in the case of Wideband mode.

The `SST25VF016BInit()` function is called to initialize the serial Flash driver memory. The `SST25VF016BStart()` function enables the dsPIC DSC device's SPI module, which communicates with the serial Flash.

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 `Speex8KHzEncode()` function and written to Flash program memory using the `SST25VF016BWrite()` function. The speech frame is also provided to the WM8731 driver for output. If the Playback mode is selected, the serial Flash memory is read using the `SST25VF016BRead()` function, decoded using the `Speex8KHzDecode()` function, and written to the WM8731 driver for output.

Chapter 4. Application Programming Interface

This chapter describes in detail the Application Programming Interface (API) functions that are available to the dsPIC DSC Speex Speech Encoding/Decoding Library.

The following topics are covered in this chapter:

4.1 ADDING THE LIBRARY TO AN APPLICATION

To use the dsPIC DSC Speex Speech Encoding/Decoding Library in an application, the library archive must be added to the application project workspace. The applicable include `.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 the 8 kHz sampling rate speech coding (Narrowband) functions with an output data rate of 8 kbps, the files `libspeex_8k_33F.a` or `libspeex_8k_33E.a`, and `speek_8k.h` can be included into the application
- If the application requires a 16 kHz sampling rate with support for multiple modes, the files, `libspeex_33F.a` or `libspeex_33E.a`, and `speex.h` must be included in the application

The procedure for adding the Wideband Speex Library archive to the project is shown below. Note that the procedure for adding the Narrowband Speex Library archive is similar, the file selected would then be `libspeex_8k.a`. The `libspeex.a` library archive also supports two Narrowband modes.

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 speex library archive (available in the `libs` folder in the installation directory).
3. Select the desired file, and then click **Open**. The library is now added to the application.

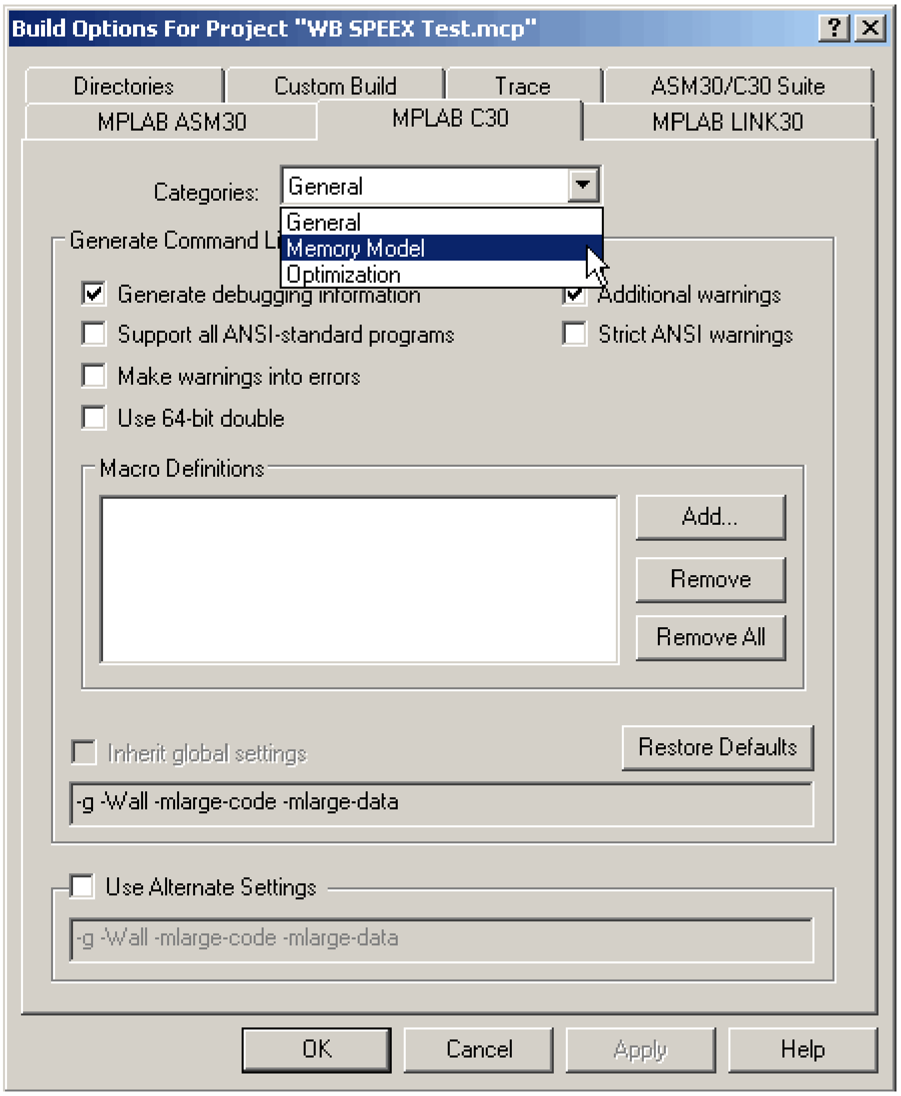
To use the library functions, include the `speex.h` file, in the application source code. This file can be copied from the `h` folder (located in the installation directory) to the application project folder.

4.2 MEMORY MODEL COMPILE OPTIONS

While using any of the Speex library archive files in an application, the compiler should be directed to use a large memory model, as described in the following steps:

1. From the MPLAB IDE menu, select Project > Build Options > Project.
2. Click the **MPLAB C30** tab and set the following options:
    a) From the Categories drop-down list, select **Memory Model**, as shown in Figure 4-1.

FIGURE 4-1: PROJECT BUILD OPTIONS



[✓] Generate debugging information
[ ] Support all ANSI-standard programs
[ ] Make warnings into errors
[ ] Use 64-bit double
[✓] Additional warnings
[ ] Strict ANSI warnings
[ ] Inherit global settings

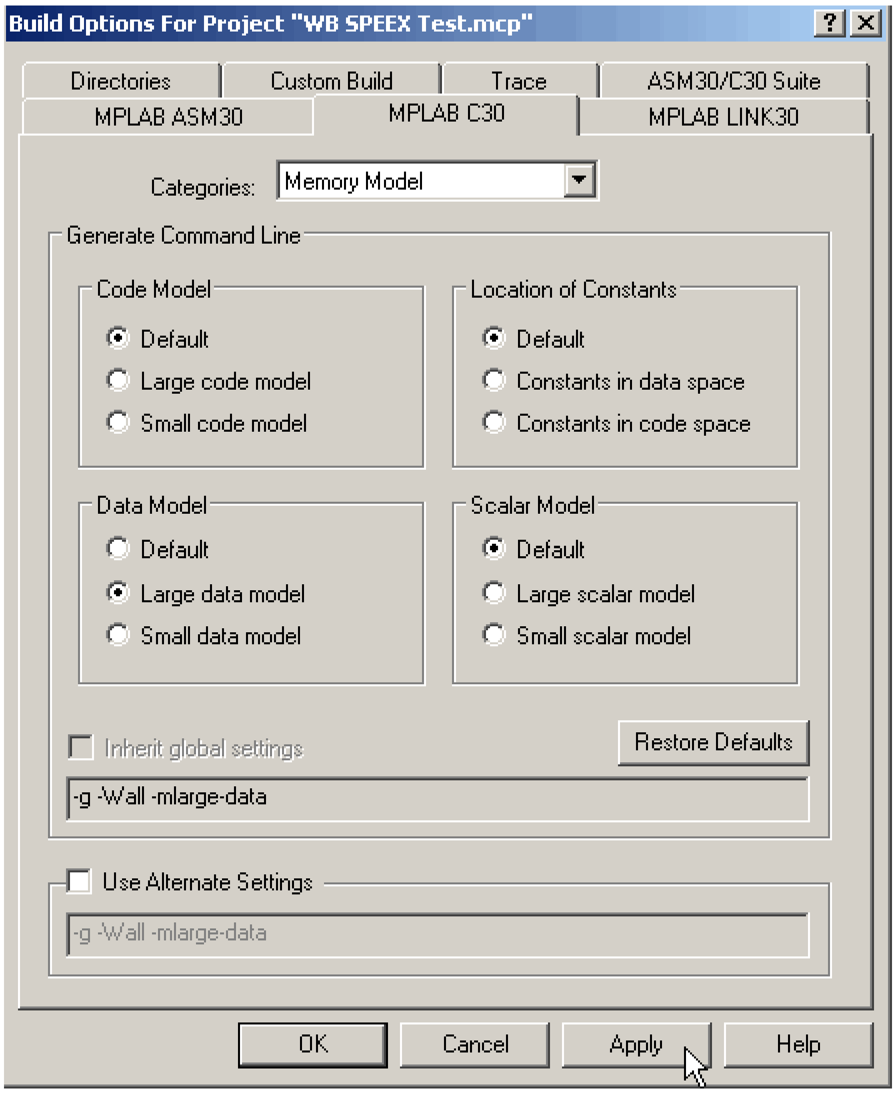
-g -Wall -mlarge-code -mlarge-data

[ ] Use Alternate Settings

-g -Wall -mlarge-code -mlarge-data

b) In the Data Model section, select **Large data model**, as shown in Figure 4-2.

FIGURE 4-2: COMPILER MEMORY MODEL SETTINGS



Categories: Memory Model

Generate Command Line

Code Model: [✓ Default] Large code model Small code model

Location of Constants: [✓ Default] Constants in data space Constants in code space

Data Model: Default [✓ Large data model] Small data model

Scalar Model: [✓ Default] Large scalar model Small scalar model

☐ Inherit global settings

-g -Wall -mlarge-data

☐ Use Alternate Settings

-g -Wall -mlarge-data

3. Click **Apply**, and then click **OK**. This completes the procedure.

4.3 USING THE LIBRARY

The dsPIC DSC Speex Speech Encoding/Decoding Library has been designed to be usable in a re-entrant environment. This enables the algorithm to process multiple channels of audio. Since the encoder and the decoder algorithms operate independently of each other, an application does not have to instantiate a decoder or encoder if it is not needed. Multiple encoders can share scratch memories. Multiple decoders can share scratch memories. The coding steps required to use the Narrowband Speex Library and Wideband Speex Library are described below.

4.3.1 Narrowband Speex Coding Steps

To use Narrowband Speex, the application must include `libspeex_8k_33F.a` or `libspeex_8k_33E.a` into the project and include `speex_8k.h` in the application code. The following coding steps must then be performed. Refer to Example 4-1 for the actual code required.

1. Allocate the memory for the Narrowband Speex Encoder. This memory is an integer array of size `NB_SPEEX_ENCODER_STATE_SIZE`. Every audio channel must have its own encoder state holder.
2. Allocate the memory for the Narrowband Speex Decoder. This memory is an integer array of size `NB_SPEEX_DECODER_STATE_SIZE`. Every audio channel must have its own decoder state holder
3. Allocate Encoder X and Y Scratch Memories. The Encoder X scratch memory is an unsigned character array of size `NB_SPEEX_ENCODER_X_SCRATCH_SIZE` and is placed in X memory aligned on an address boundary of 2 bytes. The Encoder Y scratch memory is an unsigned character array of size `NB_SPEEX_ENCODER_Y_SCRATCH_SIZE` and is placed in Y memory aligned on an address boundary of 2 bytes. Multiple encoders can share scratch memories.
4. Allocate the Decoder X and Y Scratch Memories. The Decoder X scratch memory is an unsigned character array of size `NB_SPEEX_DECODER_X_SCRATCH_SIZE` and is placed in X memory aligned on an address boundary of 2 bytes. The Decoder Y scratch memory is an unsigned character array of size `NB_SPEEX_DECODER_Y_SCRATCH_SIZE` and is placed in Y memory aligned on an address boundary of 2 bytes. Multiple decoders can share scratch memories.

Note: In some dsPIC33E devices, the Y memory is located in the Extended Data Space (EDS). In such cases, the Y scratch memory array must be tagged with the `__eds__` keyword and assigned an EDS attribute.

5. Initialize the Encoder for each audio channel. Use the `Speex8KHzEncoderInit()` function to initialize the Narrowband Speex encoder state for each audio channel.
6. Initialize the Decoder for each audio channel. Use the `Speex8KHzDecoderInit()` function to initialize the Narrowband Speex decoder state for each audio channel.
7. Encode a raw speech data frame. Use the `Speex8KHzEncode()` function to encode a frame of raw speech. Note that the size of the frame must be `NB_SPEEX_ENCODER_INPUT_SIZE`.
8. Decode a encoded speech data frame. Use the `Speex8KHzDecode()` function to encode a frame of encoded speech. Note that the size of the frame must be `NB_SPEEX_DECODER_INPUT_SIZE_8K`.

EXAMPLE 4-1: NARROWBAND SPEEX LIBRARY CODE SNIPPET

```
#include “speex_8k.h”
.
.
.
int encoder[NB_SPEEX_ENCODER_STATE_SIZE];                                  /* Step 1 */
int decoder[NB_SPEEX_DECODER_STATE_SIZE];                                  /* Step 2 */

unsigned char encXScratchMem[NB_SPEEX_ENCODER_X_SCRATCH_SIZE]; _XBSS(2);   /* Step 3 */
unsigned char encYScratchMem[NB_SPEEX_ENCODER_Y_SCRATCH_SIZE]; _YBSS(2);   /* Step 3 */
unsigned char decXScratchMem[NB_SPEEX_DECODER_X_SCRATCH_SIZE]; _XBSS(2);   /* Step 4 */
unsigned char decYScratchMem[NB_SPEEX_DECODER_Y_SCRATCH_SIZE]; _YBSS(2);   /* Step 4 */
.
.
.
Speex8KHzEncoderInit(encoder, encXScratchMem, encYScratchMem, NB_SPEEX_BITRATE_8K,
   SPEEX_NORMAL_PITCH);                                                    /* Step 5 */
Speex8KHzDecoderInit(decoder, decXScratchMem, decYScratchMem);             /* Step 6 */
.
.
.
Speex8KHzEncode(encoder, rawSamples, encodedFrame);                        /* Step 7 */
Speex8KHzDecode(decoder, encodedFrame, rawSamples);                        /* Step 8 */
```

4.3.2 Wideband Speex Coding Steps

To use Wideband Speex, the application must include `libspeex_33F.a` or `libspeex_33E.a` into the project and include `speex.h` in the application code. The following coding steps must then be performed. Refer to Example 4-2 for the actual code required.

1. Allocate the memory for the Wideband Speex Encoder. This memory is an integer array of size `WB_SPEEX_ENCODER_STATE_SIZE`. Every audio channel must have its own encoder state holder.
2. Allocate the memory for the Wideband Speex Decoder. This memory is an integer array of size `WB_SPEEX_DECODER_STATE_SIZE`. Every audio channel must have its own decoder state holder.
3. Allocate Encoder X and Y Scratch Memories. The Encoder X scratch memory is an unsigned character array of size `WB_SPEEX_ENCODER_X_SCRATCH_SIZE` and is placed in X memory aligned on an address boundary of 2 bytes. The Encoder Y scratch memory is an unsigned character array of size `WB_SPEEX_ENCODER_Y_SCRATCH_SIZE` and is placed in Y memory aligned on an address boundary of 2 bytes. Multiple encoders can share scratch memories.
4. Allocate the Decoder X and Y Scratch Memories. The Decoder X scratch memory is an unsigned character array of size `WB_SPEEX_DECODER_X_SCRATCH_SIZE` and is placed in X memory aligned on an address boundary of 2 bytes. The Decoder Y scratch memory is an unsigned character array of size `WB_SPEEX_DECODER_Y_SCRATCH_SIZE` and is placed in Y memory aligned on an address boundary of 2 bytes. Multiple decoders can share scratch memories.

Note: In some dsPIC33E devices, the Y memory is located in the Extended Data Space (EDS). In such cases, the Y scratch memory array must be tagged with the `__eds__` keyword and assigned an EDS attribute.

5. Initialize the Encoder for each audio channel. Use the `Speex16KHzEncoderInit()` function to initialize the Wideband Speex encoder state for each audio channel.
6. Initialize the Decoder for each audio channel. Use the `Speex16KHzDecoderInit()` function to initialize the Wideband Speex decoder state for each audio channel.
7. Encode a raw speech data frame. Use the `Speex16KHzEncode()` function to encode a frame of raw speech. Note that the size of the frame must be `WB_SPEEX_ENCODER_INPUT_SIZE` words.
8. Decode a encoded speech data frame. Use the `Speex16KHzDecode()` function to encode a frame of encoded speech. Depending on the selected mode, the size of the input array is either `WB_SPEEX_DECODER_INPUT_SIZE_9K8` or `WB_SPEEX_DECODER_INPUT_SIZE_12K8` bytes.

EXAMPLE 4-2: WIDEBAND SPEEX LIBRARY CODE SNIPPET

```
#include "speex.h"
.
.
.
int encoder[WB_SPEEX_ENCODER_STATE_SIZE];                                    /* Step 1 */
int decoder[WB_SPEEX_DECODER_STATE_SIZE];                                    /* Step 2 */

unsigned char encXScratchMem[WB_SPEEX_ENCODER_X_SCRATCH_SIZE]; _XBSS(2);    /* Step 3 */
unsigned char encYScratchMem[WB_SPEEX_ENCODER_Y_SCRATCH_SIZE]; _YBSS(2);    /* Step 3 */
unsigned char decXScratchMem[WB_SPEEX_DECODER_X_SCRATCH_SIZE]; _XBSS(2);    /* Step 4 */
unsigned char decYScratchMem[WB_SPEEX_DECODER_Y_SCRATCH_SIZE]; _YBSS(2);    /* Step 4 */
.
.
.
Speex16KHzEncoderInit(encoder, encXScratchMem, encYScratchMem, WB_SPEEX_BITRATE_9K8,
   SPEEX_NORMAL_PITCH);                                                      /* Step 5 */
Speex16KHzDecoderInit(decoder, decXScratchMem, decYScratchMem);              /* Step 6 */
.
.
.
Speex16KHzEncode(encoder, rawSamples, encodedFrame);                         /* Step 7 */
Speex16KHzDecode(decoder, encodedFrame, rawSamples);                         /* Step 8 */
```

4.4 REGISTER USAGE

The dsPIC DSC Speex Speech Encoding/Decoding Library uses and modifies the MODCON, CORCON, XMODSRT, XMODEND, YMODSRT, YMODEND and PSVPAG registers. These registers are saved and restored to their original values after the library has used them. If these registers are modified by the application in an Interrupt Service Routine (ISR), the application must save and restore these registers while entering and exiting the ISR. This will prevent the ISR from corrupting the execution context, if the interrupt has occurred when a library function is executing.

4.5 PACKET LOSS AND CORRUPTION

The Speex algorithm robustly handles lost packets but not corrupted ones. If the encoded packet uses an invalid mode, the decoder function will return an error code and will try to guess the correct signal. A corrupted packet (a packet that contains invalid vocoder data) could cause unpredictable decoder behavior. It is advisable in such applications to implement a higher layer error checking and correction function to ensure the integrity of the Speex packet.

4.6 PC ENCODER UTILITY (PCEU) OUTPUT FILES

The Speex PCEU generates three output files after encoding a Wave file or speech captured through the microphone. These are described in Table 4-1.

TABLE 4-1: DESCRIPTION OF FILES GENERATED BY PCEU

File Type	Description
`.s`	File containing Speex encoded data with attributes to make the data a part of the Flash program memory on the dsPIC DSC device. This file can be added to an application project.
`.spx`	File containing Speex encoded data.
`.raw`	File containing raw audio data.

The `.s` file generated by the PCEU can be included in the application workspace. After compilation, the encoded speech data now becomes a part of the Flash program memory. Consider the case of Speex encoded file encoded at a Wideband mode of 9.8 kbps. In this mode, each encoded frame has 25 bytes of data. Example 4-3 shows the starting portion of the `.s` file generated by the PCEU for this mode.

EXAMPLE 4-3: .S FILE FOR 9.8 kbps WIDEBAND MODE

```
/*********************************************************************/
/*    Speech Encoder Utility settings:                               */
/*       Input Source:   Microphone                                  */
/*       Output Array:   speex_data                                  */
/*       Array Size:     2502 bytes                                  */
/*       Target Memory:  Program Memory                              */
/*       Bitrate:        9.8 kbps                                    */
/*********************************************************************/

/* There are 834 elements in the data array. */
/* Data file for storing 24-bit constants in program memory */

.global _speex_data

.section .speex, "x"
_speex_data:
.pword 0x84DD1D,   0x390030,   0x0070CE,   0x38E71C,   0x762E00,   0x141C14
.pword 0x930EAB,   0x0000B6,   0xC41D07,   0x3C0601,   0x730BE4,   0x8530D8
.pword 0xC3A32A,   0x2EE149,   0x2B252B,    0x89B693,  0x1D97C8,   0xFBC2DE
```

To optimize memory usage, the encoder utility will use all 24 bits (3 bytes) of one program word to store data. Hence the first byte of the frame is stored at the low program word. The second byte is stored at the high program byte and the third byte of the frame is stored at the upper program byte. With reference to Example 4-3:

- 0x1D – First Byte of frame stored at low program byte
- 0xDD – Second Byte of Frame stored at high program byte
- 0x84 – Third Byte of Frame stored at upper program byte

The application program must use the `TBLRD` instructions to access the encoded data in Flash program memory since the upper program byte of the program word contains valid frame data and needs to be read.

4.7 RESOURCE REQUIREMENTS

The resource requirements for running the dsPIC DSC Speex Speech Encoding/Decoding 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.

4.7.1 Narrowband-Only Library

The Narrowband-Only version supports one mode: 8 kbps at 8 kHz sampling rate. The resource requirements for this library are as follows:

TABLE 4-2: PROGRAM MEMORY USAGE

Resource	Size in Bytes	Section
Encoder and Decoder	29712 (dsPIC30F/dsPIC33F) 30657 (dsPIC33E)	`.text` and `.const`

TABLE 4-3: REQUIRED INPUT AND OUTPUT BUFFERS

Resource	Size in Bytes	Alignment	Section
Encoder Input Buffer	320	2	X data memory
Encoder Output Buffer (8 kbps mode)	20	2	X data memory
Decoder Output Buffer	320	2	X data memory

Note: The Decoder Output Buffer is not required as the Encoder Input Buffer is reused by the Decoder for output.

TABLE 4-4: STATE AND SCRATCH MEMORIES

Resource	Size in Bytes	Alignment	Section
Encoder State Memory	1642	2	X data memory
Decoder State Memory	1394	2	X data memory
Encoder Scratch X Memory	800	2	X data memory
Encoder Scratch Y Memory	1500	2	Y data memory
Decoder Scratch X Memory	82	2	X data memory
Decoder Scratch Y Memory	850	2	Y data memory

TABLE 4-5: TABLES AND CONSTANTS IN RAM

Resource	Size in Bytes	Alignment	Section
Encoder and Decoder	122 (dsPIC30F/dsPIC33F) 2606 (dsPIC33E)	2	X data memory

TABLE 4-6: MIPS

Function	MIPS	Typical Call Frequency
`Speex8KHzEncoderInit()`	0.01	Once
`Speex8KHzDecoderInit()`	0.02	Once
`Speex8KHzEncode()`	17.8 (dsPIC30F/dsPIC33F) 22.8 (dsPIC33E)	20 ms
`Speex8KHzDecode()`	2.1 (dsPIC30F/dsPIC33F) 2.7 (dsPIC33E)	20 ms

TABLE 4-7: DATA FORMAT

I/O Type	Data Type
Encoder Input	16-bit linear Pulse Code Modulation (PCM) data
Decoder Output	16-bit linear PCM data

4.7.2 Wideband and Narrowband Speex

The Wideband and Narrowband version supports the following modes:

- 8 kbps at 8 kHz sampling rate
- 11 kbps at 8 kHz sampling rate
- 9.8 kbps at 16 kHz sampling rate
- 12.8 kbps at 16 kHz sampling rate

The resource requirements for this library are as follows:

TABLE 4-8: PROGRAM MEMORY USAGE

Resource	Size in Bytes	Section
Wideband Encoder and Wideband Decoder	38685 (dsPIC33F) 32289 (dsPIC33E)	`.text` and `.const`
Narrowband Encoder and Narrowband Decoder	38706 (dsPIC33F) 32289 (dsPIC33E)	`.text` and `.const`

TABLE 4-9: REQUIRED INPUT AND OUTPUT BUFFERS

Resource	Size in Bytes	Alignment	Section
Wideband Encoder Input Buffer	640	2	X data memory
Narrowband Encoder Input Buffer	320	2	X data memory
Encoder Output Buffer (8 kbps mode)	20	2	X data memory
Encoder Output Buffer (11 kbps mode)	28	2	X data memory
Encoder Output Buffer (9.8 kbps mode)	25	2	X data memory
Encoder Output Buffer (12.8 kbps mode)	32	2	X data memory
Narrowband Decoder Output Buffer	320	2	X data memory
Wideband Decoder Output Buffer	640	2	X data memory

Note: The Decoder Output Buffer is not required, if the Encoder Input Buffer is reused by the Decoder for output.

TABLE 4-10: STATE AND SCRATCH MEMORIES

Resource	Size in Bytes	Alignment	Section
Narrowband Encoder State Memory	1642	2	X data memory
Narrowband Decoder State Memory	1394	2	Y data memory
Wideband Encoder State Memory	2168	2	X data memory
Wideband Decoder State Memory	1844	2	Y data memory
Encoder Scratch X Memory	800	2	X data memory
Encoder Scratch Y Memory	2700	2	Y data memory
Decoder Scratch X Memory	82	2	X data memory
Decoder Scratch Y Memory	850	2	Y data memory

TABLE 4-11: TABLES AND CONSTANTS IN RAM

Resource	Size in Bytes	Alignment	Section
Wideband Encoder and Wideband Decoder	2810 (dsPIC33F)	2	X data memory
	714 (dsPIC33F)	2	Y data memory
Wideband Encoder and Wideband Decoder	6224 (dsPIC33E)	2	X data memory
	0 (dsPIC33E)	2	Y data memory
Narrowband Encoder and Narrowband Decoder	2810 (dsPIC33F)	2	X data memory
	714 (dsPIC33F)	2	Y data memory
Narrowband Encoder and Narrowband Decoder	6224 (dsPIC33E)	2	X data memory
	0 (dsPIC33E)	2	Y data memory

TABLE 4-12: MIPS

Function	MIPS	Typical Call Frequency
`Speex8KHzEncoderInit()`	0.01	Once
`Speex8KHzDecoderInit()`	0.02	Once
`Speex8KHzEncode()`	17.1 (dsPIC33F) 22.2 (dsPIC33E)	20 ms
`Speex8KHzDecode()`	2.1 (dsPIC33F) 2.7 (dsPIC33E)	20 ms
`Speex16KHzEncoderInit()`	0.02	Once
`Speex16KHzDecoderInit()`	0.04	Once
`Speex16KHzEncode()`	21.5 MIPS (dsPIC33F) 27.8 MIPS (dsPIC33E)	20 ms
`Speex16KHzDecode()`	4.5 (dsPIC33F) 5.6 (dsPIC33E)	20 ms

TABLE 4-13: DATA FORMAT

I/O Type	Data Type
Encoder Input	16-bit linear PCM data
Decoder Output	16-bit linear PCM data

4.8 API FUNCTIONS

This section lists and describes the API functions that are available in the dsPIC DSC Speex Speech Encoding/Decoding Library. The functions are listed below followed by their individual detailed descriptions:

- `Speex8KHzEncoderInit`
- `Speex8KHzDecoderInit`
- `Speex16KHzEncoderInit`
- `Speex16KHzDecoderInit`
- `Speex8KHzEncode`
- `Speex8KHzDecode`
- `Speex16KHzEncode`
- `Speex16KHzDecode`
- `SPEEX_NORMAL_PITCH`
- `SPEEX_LOW_PITCH`
- `NB_SPEEX_BITRATE_8K`
- `NB_SPEEX_BITRATE_11K`
- `WB_SPEEX_BITRATE_9K8`
- `WB_SPEEX_BITRATE_12K8`
- `NB_SPEEX_ENCODER_INPUT_SIZE`
- `NB_SPEEX_DECODER_OUTPUT_SIZE`
- `WB_SPEEX_ENCODER_INPUT_SIZE`
- `WB_SPEEX_DECODER_OUTPUT_SIZE`
- `NB_SPEEX_ENCODED_FRAME_SIZE_8K`
- `NB_SPEEX_ENCODED_FRAME_SIZE_11K`
- `WB_SPEEX_ENCODED_FRAME_SIZE_9K8`
- `WB_SPEEX_ENCODED_FRAME_SIZE_9K8`
- `WB_SPEEX_ENCODED_FRAME_SIZE_12K8`
- `SPEEX_ENCODER_X_SCRATCH_SIZE`
- `SPEEX_ENCODER_Y_SCRATCH_SIZE`
- `SPEEX_DECODER_X_SCRATCH_SIZE`
- `SPEEX_DECODER_Y_SCRATCH_SIZE`
- `NB_SPEEX_ENCODER_STATE_SIZE`
- `NB_SPEEX_DECODER_STATE_SIZE`
- `WB_SPEEX_ENCODER_STATE_SIZE`
- `WB_SPEEX_DECODER_STATE_SIZE`
- `SPEEX_INVALID_MODE_ID`

Speex8KHzEncoderInit

Description

Initializes the Narrowband Speex Encoder.

Include

```
speex.h
speex_8k.h
```

Prototype

```
void Speex8KHzEncoderInit(int* encoderState, unsigned char*
xScratchMem, unsigned char* yScratchMem,int bitRate,int pitch);
```

Arguments

encoderState	a pointer to the state memory for this instance of Narrowband Speex encoder
xScratchMem	a pointer to an area of scratch memory in X RAM used by the encoder
yScratchMem	a pointer to an area of scratch memory in Y RAM used by the encoder
bitRate	a value representing the bit rate at which the encoder will operate. Value can be either `NB_SPEEX_BITRATE_8K` for 8 kbps mode or `NB_SPEEX_BITRATE_11K` for 11 kbps mode.
pitch	a value representing the pitch range for the encoder. Value can be either `SPEEX_LOW_PITCH` for low pitch range or `SPEEX_NORMAL_PITCH` for normal pitch range.

Return Value

None.

Remarks

Each audio channel must have its own encoder state. Encoders can share scratch memory as long as the encode functions are called in the same context (that is, an encode function is not called in an ISR or another task).

Code Example

```
int              encoder      [NB_SPEEX_ENCODER_STATE_SIZE] _XBSS(2);
unsigned char    xScratchMem  [SPEEX_ENCODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char    yScratchMem  [SPEEX_ENCODER_Y_SCRATCH_SIZE] _YBSS(2);
.
.
.
/* Initialize an Narrowband Speex encoder at bit
 * rate of 8 kbps and normal pitch range */

Speex8KHzEncoderInit(encoder,xScratchMem,yScratchMem,NB_SPEEX_BITRATE_
8K, SPEEX_NORMAL_PITCH);
```

Speex8KHzDecoderInit

Description

Initializes the Narrowband Speex Decoder.

Include

```
speex.h
speex_8k.h
```

Prototype

```
void Speex8KHzDecoderInit(int* decoderState, unsigned char*
xScratchMem, unsigned char* yScratchMem);
```

Arguments

`decoderState`	a pointer to the state memory for this instance of Narrowband Speex decoder
`xScratchMem`	a pointer to an area of scratch memory in X RAM used by the decoder
`yScratchMem`	a pointer to an area of scratch memory in Y RAM used by the decoder

Return Value

None.

Remarks

Each audio channel must have its own decoder state. Decoders can share scratch memory as long as the decode functions are called in the same context (that is, a decode function is not called in an ISR or another task).

Code Example

```
int             decoder       [NB_SPEEX_DECODER_STATE_SIZE] _XBSS(2);
unsigned char   xScratchMem   [SPEEX_DECODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char   yScratchMem   [SPEEX_DECODER_Y_SCRATCH_SIZE] _YBSS(2);
.
.
.
/* Initialize a Narrowband Speex decoder */

Speex8KHzDecoderInit(decoder,xScratchMem,yScratchMem);
```

Speex16KHzEncoderInit

Description

Initializes the Wideband Speex Encoder.

Include

speex.h

Prototype

```
void Speex16KHzEncoderInit(int* encoderState, unsigned char*
xScratchMem, unsigned char* yScratchMem,int bitRate,int pitch);
```

Arguments

encoderState	a pointer to the state memory for this instance of Narrowband Speex encoder
xScratchMem	a pointer to an area of scratch memory in X RAM used by the encoder
yScratchMem	a pointer to an area of scratch memory in Y RAM used by the encoder
bitRate	a value representing the bit rate at which the encoder will operate. Value can be either WB_SPEEX_BITRATE_9K8 for 9.8 kbps mode or WB_SPEEX_BITRATE_12K8 for 12.8 kbps mode
pitch	a value representing the pitch range for the encoder. Value can be either SPEEX_LOW_PITCH for low pitch range or SPEEX_NORMAL_PITCH for normal pitch range

Return Value

None.

Remarks

Each audio channel must have its own encoder state. Encoders can share scratch memory as long as the encode functions are called in the same context (that is, an encode function is not called in an ISR or another task).

Code Example

```
int             encoder       [WB_SPEEX_ENCODER_STATE_SIZE] _XBSS(2);
unsigned char   xScratchMem   [SPEEX_ENCODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char   yScratchMem   [SPEEX_ENCODER_Y_SCRATCH_SIZE] _YBSS(2);

/* Initialize a Wideband Speex encoder at bit
 * rate of 12.8 kbps and normal pitch range */
.
.
.
Speex16KHzEncoderInit(encoder,xScratchMem,yScratchMem,WB_SPEEX_BITRATE
_12K8, SPEEX_NORMAL_PITCH);
```

Speex16KHzDecoderInit

Description

Initializes the Wideband Speex Decoder.

Include

`speex.h`

Prototype

```
void Speex16KHzDecoderInit(int* decoderState, unsigned char*
xScratchMem, unsigned char* yScratchMem);
```

Arguments

`decoderState`	a pointer to the state memory for this instance of Wideband Speex decoder
`xScratchMem`	a pointer to an area of scratch memory in X RAM used by the decoder
`yScratchMem`	a pointer to an area of scratch memory in Y RAM used by the decoder

Return Value

None.

Remarks

Each audio channel must have its own decoder state. Decoders can share scratch memory as long as the decode functions are called in the same context (that is, a decode function is not called in an ISR or another task).

Code Example

```
int              decoder      [WB_SPEEX_DECODER_STATE_SIZE] _XBSS(2);
unsigned char    xScratchMem  [SPEEX_DECODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char    yScratchMem  [SPEEX_DECODER_Y_SCRATCH_SIZE] _YBSS(2);
.
.
.
/* Initialize a Wideband Speex decoder */

Speex16KHzDecoderInit(decoder,xScratchMem,yScratchMem);
```

Speex8KHzEncode

Description

Invokes the Narrowband Speex Encoder to encode a frame of data.

Include

```
speex.h
speex_8k.h
```

Prototype

```
int Speex8KHzEncode(int* encoderState, int * input, char* output);
```

Arguments

`encoderState`	a pointer to the state memory for this instance of Narrowband Speex encoder
`input`	a pointer to frame of raw speech data
`output`	a pointer to the encoded frame

Return Value

Returns the size of the encoded frame in bytes.

Remarks

Size of output array depends on the encoder mode. It can be either `NB_SPEEX_ENCODER_OUTPUT_SIZE_8K` or `NB_SPEEX_ENCODER_OUTPUT_SIZE_11K`. Size of the input array must be `NB_SPEEX_ENCODER_INPUT_SIZE`.

Code Example

```
int             encoder     [NB_SPEEX_ENCODER_STATE_SIZE] _XBSS(2);
unsigned char   xScratchMem [SPEEX_ENCODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char   yScratchMem [SPEEX_ENCODER_Y_SCRATCH_SIZE] _YBSS(2);
int             inputFrame  [NB_SPEEX_ENCODER_INPUT_SIZE] _XBSS(2);
char            outputFrame [NB_SPEEX_ENCODER_OUTPUT_SIZE_8K] _XBSS(2);
int             bytes;
.
.
.
/* Initialize a Narrowband Speex encoder for operation
 * at 8 kbps and normal pitch range */

Speex8KHzEncoderInit(encoder,xScratchMem,yScratchMem,NB_SPEEX_BITRATE_
8K,SPEEX_NORMAL_PITCH);
.
.
.
/* Encode a frame of data */

bytes = Speex8KHzEncode(encoder,inputFrame,outputFrame);
```

Speex8KHzDecode

Description

Invokes the Narrowband Speex Decoder to decode an encoded frame of data.

Include

```
speex.h
speex_8k.h
```

Prototype

```
int Speex8KHzDecode(int* decoderState, char * input, int* output);
```

Arguments

`decoderState`	a pointer to the state memory for this instance of Narrowband Speex decoder
`input`	a pointer to frame of encoded data
`output`	a pointer to the decoded data

Return Value

Returns ‘`0`’ if no error was encountered. Returns `SPEEX_INVALID_MODE_ID` if the input encoded Speex frame has an unsupported mode.

Remarks

The decoder uses the information stored in the packet to obtain encoded frame pitch range and the encoded mode. `The NB_SPEEX_DECODER_OUTPUT_SIZE` macro can be used to specify the output array size.

Code Example

```
int           decoder       [NB_SPEEX_DECODER_STATE_SIZE] _XBSS(2);
unsigned char xScratchMem   [SPEEX_DECODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char yScratchMem   [SPEEX_DECODER_Y_SCRATCH_SIZE] _YBSS(2);
char          inputFrame    [NB_SPEEX_ENCODER_OUTPUT_SIZE_8K] _XBSS(2);
int           outputFrame   [NB_SPEEX_DECODER_OUTPUT_SIZE] _XBSS(2);
int           retVal;
.
.
.
/* Initialize a Narrowband Speex decoder */

Speex8KHzDecoderInit(decoder,xScratchMem,yScratchMem);
.
.
.
/* Decode a frame of data */

retVal = Speex8KHzDecode(encoder,inputFrame,outputFrame);
```

Speex16KHzEncode

Description

Invokes the Wideband Speex Encoder to encode a frame of data.

Include

`speex.h`

Prototype

```
int Speex16KHzEncode(int* encoderState, int * input, char* output);
```

Arguments

`encoderState`	a pointer to the state memory for this instance of Wideband Speex encoder
`input`	a pointer to frame of raw speech data
`output`	a pointer to the encoded frame

Return Value

Returns the size of the encoded frame in bytes.

Remarks

Size of output array depends on the encoder mode. It can be either `WB_SPEEX_ENCODER_OUTPUT_SIZE_9K8` or `WB_SPEEX_ENCODER_OUTPUT_SIZE_12K8`. Size of input array must be `WB_SPEEX_ENCODER_INPUT_SIZE`.

Code Example

```
int           encoder      [WB_SPEEX_ENCODER_STATE_SIZE] _XBSS(2);
unsigned char xScratchMem  [SPEEX_ENCODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char yScratchMem  [SPEEX_ENCODER_Y_SCRATCH_SIZE] _YBSS(2);
int           inputFrame   [WB_SPEEX_ENCODER_INPUT_SIZE] _XBSS(2);
char          outputFrame  [WB_SPEEX_ENCODER_OUTPUT_SIZE_9K8] _XBSS(2);
int           bytes;
.
.
.
/* Initialize a Wideband Speex encoder for operation
 * at 9.8 kbps and normal pitch range */

Speex16KHzEncoderInit(encoder,xScratchMem,yScratchMem,NB_SPEEX_BITRATE
_9K8,SPEEX_NORMAL_PITCH);
.
.
.
/* Encode a frame of data */

bytes = Speex16KHzEncode(encoder,inputFrame,outputFrame);
```

`Speex16KHzDecode`

Description

Invokes the Wideband Speex Decoder to decode an encoded frame of data.

Include

`speex.h`

Prototype

```
int Speex16KHzDecode(int* decoderState, char * input, int* output);
```

Arguments

`decoderState`	a pointer to the state memory for this instance of Wideband Speex decoder
`input`	a pointer to frame of encoded data
`output`	a pointer to the decoded data

Return Value

Returns '`0`' if no error was encountered. Returns `SPEEX_INVALID_MODE_ID` if the packet has an invalid mode.

Remarks

The decoder uses the information stored in the packet to obtain encoded frame pitch range and the encoded mode. The `WB_SPEEX_DECODER_OUTPUT_SIZE` macro can be used to specify the output array size.

Code Example

```
int           decoder      [WB_SPEEX_DECODER_STATE_SIZE] _XBSS(2);
unsigned char xScratchMem  [SPEEX_DECODER_X_SCRATCH_SIZE] _XBSS(2);
unsigned char yScratchMem  [SPEEX_DECODER_Y_SCRATCH_SIZE] _YBSS(2);
char          inputFrame   [WB_SPEEX_ENCODER_OUTPUT_SIZE_9K8] _XBSS(2);
int           outputFrame  [WB_SPEEX_DECODER_OUTPUT_SIZE] _XBSS(2);
int           retVal;
.
.
.
/* Initialize a Wideband Speex decoder */

Speex16KHzDecoderInit(decoder,xScratchMem,yScratchMem);
.
.
.
/* Decode a frame of data */

retVal = Speex16KHzDecode(decoder,inputFrame,outputFrame);
```

SPEEX_NORMAL_PITCH

Description

Indicates to the Speex encoder to use normal pitch range.

Value

0

SPEEX_LOW_PITCH

Description

Indicates to the Speex encoder to use low pitch range.

Value

1

NB_SPEEX_BITRATE_8K

Description

Indicates to the Narrowband Speex encoder to use the 8 kbps mode.

Value

3

NB_SPEEX_BITRATE_11K

Description

Indicates to the Narrowband Speex encoder to use the 11 kbps mode.

Value

4

WB_SPEEX_BITRATE_9K8

Description

Indicates to the Wideband Speex encoder to use the 9.8 kbps mode.

Value

3

WB_SPEEX_BITRATE_12K8

Description

Indicates to the Wideband Speex encoder to use the 12.8 kbps mode.

Value

4

NB_SPEEX_ENCODER_INPUT_SIZE

Description

Defines the size of the Narrowband Speex encoder input integer type array.

Value

160

NB_SPEEX_DECODER_OUTPUT_SIZE

Description

Defines the size of the Narrowband Speex decoder output integer type array.

Value

160

WB_SPEEX_ENCODER_INPUT_SIZE

Description

Defines the size of the Wideband Speex encoder input integer type array.

Value

320

WB_SPEEX_DECODER_OUTPUT_SIZE

Description

Defines the size of the Wideband Speex decoder output integer type array.

Value

320

NB_SPEEX_ENCODED_FRAME_SIZE_8K

Description

Defines the size of the Narrowband Speex encoder output char type array for 8 kbps mode.

Value

20

NB_SPEEX_ENCODED_FRAME_SIZE_11K

Description

Defines the size of the Narrowband Speex encoder output char type array for 11 kbps mode.

Value

28

WB_SPEEX_ENCODED_FRAME_SIZE_9K8

Description

Defines the size of the Wideband Speex encoder output char type array for 9.8 kbps mode.

Value

25

WB_SPEEX_ENCODED_FRAME_SIZE_12K8

Description

Defines the size of the Wideband Speex encoder output char type array for 12.8 kbps mode.

Value

32

`SPEEX_ENCODER_X_SCRATCH_SIZE`

Description

Defines the Speex Encoder X scratch memory (unsigned char type array) size.

Value

800

`SPEEX_ENCODER_Y_SCRATCH_SIZE`

Description

Defines the Speex Encoder Y scratch memory (unsigned char type array) size.

Value

2700 when using `libspeex.a`

1500 when using `libspeex_8k.a`

`SPEEX_DECODER_X_SCRATCH_SIZE`

Description

Defines the Speex Decoder X scratch memory (unsigned char type array) size.

Value

82

`SPEEX_DECODER_Y_SCRATCH_SIZE`

Description

Defines the Speex Decoder Y scratch memory (unsigned char type array) size.

Value

850

`NB_SPEEX_ENCODER_STATE_SIZE`

Description

Defines the Narrowband Speex Encoder State (integer array) size.

Value

0x335

NB_SPEEX_DECODER_STATE_SIZE

Description

Defines the Narrowband Speex Decoder State (integer array) size.

Value

0x2B9

WB_SPEEX_ENCODER_STATE_SIZE

Description

Defines the Wideband Speex Encoder State (integer array) size.

Value

0x43C

WB_SPEEX_DECODER_STATE_SIZE

Description

Defines the Wideband Speex Encoder State (integer array) size.

Value

0x39A

SPEEX_INVALID_MODE_ID

Description

The value returned by the decoder if the mode ID of the encoded frame is incorrect.

Value

0xFFFE

4.9 APPLICATION TIPS

The following are a few tips to consider when using the dsPIC DSC Speex Speech Encoding/Decoding Library.

1. The optimum input signal levels for testing audio systems are generally considered to be between -10 dBm0 and -30 dBm0. If digital input speech levels have peaks that are up to three-fourths 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 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.
4. For devices with 8 KB RAM (such as the dsPIC30F DSC devices), use the Narrowband-Only library archive. This implementation of a single Narrowband mode has been optimized for the 8 KB devices.

NOTES:

Index

MICROCHIP

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