dsPIC® DSC Line Echo Cancellation Library User's Guide

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

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

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

Trademarks

ISBN: 978-1-61341-359-3

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

Microchip received ISO/TS-16949:2009 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

Chapter 1. Introduction

1.1 Line Echo Cancellation Overview 10
1.2 Features 11
1.3 Host System Requirements 12

Chapter 2. Installation

2.1 Installation Procedure 14
2.2 LEC Library Files 14

Chapter 3. LEC Demonstration

3.1 LEC Demonstration for the dsPIC33E Device Family 18
3.2 LEC Demonstration for the dsPIC33E Device Family 23

Chapter 4. Application Programming Interface (API)

4.1 Adding the Line Echo Cancellation Library to an Application 29
4.2 Memory Model Compile Options 30
4.3 LEC Algorithm Overview 33
4.4 Library Usage 35
4.5 Resource Requirements 37
4.6 Line Echo Cancellation Library API Functions 38
4.7 Application Tips 62

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 Line Echo Cancellation Library. Items discussed in this chapter 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 user's guide describes how to use the dsPIC DSC Line Echo Cancellation Library. The document is organized as follows:

- **Chapter 1. "Introduction"** – This chapter introduces the dsPIC DSC Line Echo Cancellation Library and provides a brief overview of line echo cancellation and the library features. It also outlines requirements for a host PC.
- **Chapter 2. "Installation"** – This chapter provides instructions for installing the library files and describes the contents of the source files, include files, demo files and archive files.
- **Chapter 3. "LEC Demonstration"** – This chapter provides a hands-on demonstration of line echo cancellation in a working application.
- **Chapter 4. "Application Programming Interface (API)"** – This chapter outlines how the API functions provided in the dsPIC DSC Line Echo Cancellation Library can be included in your application software via 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	*File>Save*
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 Line Echo Cancellation Library. Other useful documents include:

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 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 and MCU family architecture and peripheral modules, but do not cover the specifics of each device. Refer to the appropriate device data sheet for device-specific information.

dsPIC33E/PIC24E Family Reference Manual Sections

Refer to this documents for detailed information on dsPIC33E/PIC24E device operation. These reference manual sections explain the operation of the dsPIC33E/PIC24E DSC and MCU 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 families of devices. It describes the instruction set in detail and also provides general information to assist in developing software for these devices.

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).

MPLAB® IDE Simulator, Editor User's Guide (DS51025)

Refer to this document for more information pertaining to the installation and implementation of the MPLAB Integrated Development Environment (IDE) Software.

To obtain any of these documents, visit the Microchip web site at www.microchip.com.

Microsoft® Windows® Manuals

This user's guide assumes that you are familiar with the Microsoft Windows operating system. Many excellent references exist for this software program and should be referenced for general operation of Windows.

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 Microchip in-circuit emulators. This includes the MPLAB ICE 2000, MPLAB ICE 4000 and MPLAB REAL ICE™.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debuggers, MPLAB ICD 2 and 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 and PRO MATE® II device programmers and the PICSTART® Plus and PICkit™ 1, 2 and 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
- Development Systems Information Line

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

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

DOCUMENT REVISION HISTORY

Revision A (November 2005)

Initial release of this document.

Revision B (August 2008)

This revision includes the following updates for version 3.0 of the Line Echo Cancellation Library.

This document has been renamed from "*dsPIC30F Line Echo Cancellation Library User's Guide"* to its new name of "*dsPIC® DSC Line Echo Cancellation Library User's Guide"*. There has been no change to the Microchip literature number (DS70170) other than a revision letter update.

Each chapter has been extensively reworked and reorganized to support all dsPIC DSC devices. The previous version of the document contained seven chapters, which have been consolidated as described in the following table:

MAJOR UPDATES

Chapter Name	Update Description
Chapter 1. "Introduction"	The optional accessory kit is no longer available and the related information has been removed. The reference to the user functions has been removed. The host system requirements section was updated to include the HTML browser requirement.
Chapter 2. "Installation"	The installation procedure was updated to accommodate the installation CD file changes. See **Section 2.1 "Installation Procedure"**. The Line Echo Cancellation Library files have been updated and the `inc` folder was renamed to h. See **Section 2.2 "LEC Library Files"**.
Chapter 3. "LEC Demonstration" (formerly Chapter 5. Line Echo Cancellation Demo)	The dsPICDEM™ 1.1 Plus development board jumper (J9) setting has been changed from MASTER to SLAVE. See **Section 3.1.2 "Demonstration Setup"** and Figure 3-2. Due to the removal of the optional Acoustic Accessory Kit, the reference to the 14.7456 MHz oscillator in the Demonstration Setup section has been removed. The demonstration procedure has been completely rewritten to provide more information on the state of operation. See **Section 3.1.3 "Demonstration Procedure"**.
Chapter 4. "Application Programming Interface (API)" (formerly Chapter 3, same title)	This chapter has been updated with a completely new set of API functions. See **Section 4.6 "Line Echo Cancellation Library API Functions"**. The information in the chapter formerly known as Line Echo Cancellation Algorithm was consolidated and relocated to the Application Programming Interface (API) chapter. See **Section 4.4 "Library Usage"**. The information in the formerly known Resource Requirements chapter was consolidated and relocated to the Application Programming Interface (API) chapter. See **Section 4.5 "Resource Requirements"**.

Revision C (November 2008)

- Added reference to new library archive support for dsPIC30F devices in **Section 1.2 "Features"**
- Updated demonstration file names inTable
- Added `lec_30F_api.h` to Table 2-2
- Added `leclib_30F_v3_0.a` to Table 2-3
- Updated step 1 in **Section 3.1.3 "Demonstration Procedure"**
- Updated text in **Section 3.1.4 "Demonstration Code Description"** as follows:
  - `Si3000CodecInit()` has been changed to `SI3000_open()`
  - `init_uart()` has been changed to `UART1_open()`
  - Updated UART baud rate from ~115200 to ~250000
  - Updated text in the first and second sentences of the seventh paragraph to clarify codec data buffer interaction with the codec driver
  - Updated text in the second and third sentences of the ninth paragraph to clarify main loop functionality
- Updated step 2 and last paragraph of **Section 4.1 "Adding the Line Echo Cancellation Library to an Application"** to include dsPIC30F device file references
- Added **Section 4.2 "Memory Model Compile Options"**
- Updated code in Example 4-1

Revision D (July 2011)

This revision includes the following updates:

- Updated the first paragraph in **Chapter 1. "Introduction"**, which now includes references to the dsPIC33E family of devices
- Completely revised **Chapter 2. "Installation"**
- Updated **Chapter 3. "LEC Demonstration"**, which now distinguishes between dsPIC33F and dsPIC33E devices (see **3.1 "LEC Demonstration for the dsPIC33E Device Family"** and **3.2 "LEC Demonstration for the dsPIC33E Device Family"**)
- Completely revised **4.1 "Adding the Line Echo Cancellation Library to an Application"** in **Chapter 4. "Application Programming Interface (API)"**
- Updated the process to enable the use of the LEC Library in **4.4 "Library Usage"** (a new step 2 was added)
- Added LEC Channel 1 and LEC Channel 2 data array code to Example 4-2
- Updated all tables in **4.5 "Resource Requirements"**
- Added titles to Example 4-1 and Example 4-1 in LEC_setNLPThreshold
- Updates to formatting and minor text changes have been incorporated throughout the document

Chapter 1. Introduction

This chapter introduces the dsPIC DSC Line Echo Cancellation library. This library, which supports the dsPIC33F and dsPIC33E families of devices, provides functionality to eliminate echo generated in various telephone and digital network components. This user's guide provides information you can use to incorporate Line Echo Cancellation (LEC) capability into your embedded solution. Topics covered include:

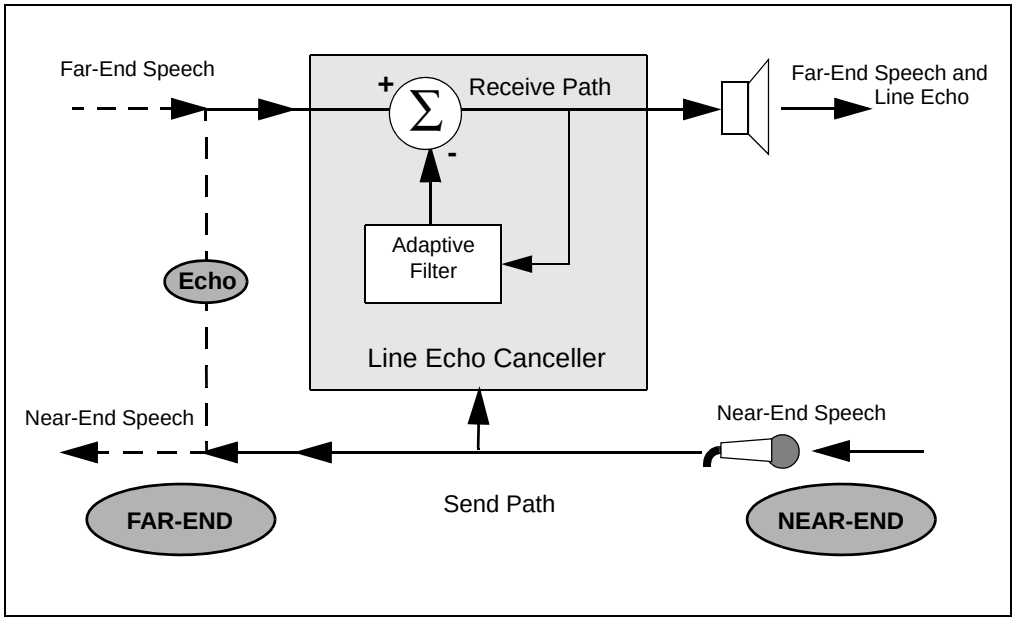
- Line Echo Cancellation Overview
- Features
- Host System Requirements

1.1 LINE ECHO CANCELLATION OVERVIEW

LEC eliminates echoes generated in the electrical path between the transmitter and the receiver in a communication system. As seen in the full-duplex communication setup shown in Figure 1-1, the "near-end" receives a signal from the network that contains an echo introduced by the communication network. Typically, echoes are the result of impedance mismatch in 2-wire to 4-wire telephone hybrids and other network components. This "far-end" line echo results in a perceptible and distracting echo effect at the near-end.

Line echo cancellation is useful for telephony applications that involve transmitting and receiving signals through a telephone hybrid. It is also useful for digital network applications such as cellular telephony and voice over internet protocol. Although the dsPIC DSC Line Echo Cancellation Library is targeted to eliminate far-end echo (as demonstrated by the demonstration application), the library functions are equally applicable to eliminating near-end echo.

FIGURE 1-1: LINE ECHO CANCELLATION IN A SPEECH AND TELEPHONY APPLICATION



The LEC library uses a fixed 8 kHz sampling rate and is especially suitable for applications such as:

- Hands-free cell phone kits
- Speaker phones
- Intercoms
- Teleconferencing systems

For hands-free cell phones intended to be used in automotive environments, such as a car cabin, this library is compatible with the G.168 Standard for Line Echo Cancellation and with other relevant ITU-T standards.

The LEC library is written almost entirely in Assembly language and is highly optimized to make extensive use of the dsPIC DSC device instruction set and advanced addressing modes. The algorithm avoids data overflow. The LEC library provides an `LEC_init` function for initializing the various data structures required by the algorithm and an `LEC_apply` function to remove the echo component from a 10 ms block of sampled 16-bit speech data. You can easily call both functions through a well-documented Application Programmer's Interface (API).

The LEC algorithm is primarily a Time Domain algorithm. The received far-end speech samples (typically sampled from a microphone) are filtered using an adaptive Finite Impulse Response (FIR) filter. The coefficients of this filter are adapted using the Normalized Least Mean Square (NLMS) algorithm, such that the filter closely models the electrical path between the transmitter and receiver (e.g., the path through a telephone hybrid), which is essentially the path traversed by the echo.

A Double Talk Detection (DTD) feature is used to avoid updating the filter coefficients when there is simultaneous speech from both ends of the communication link (double talk). As a consequence, the algorithm facilitates full-duplex communication. An optional Tone Detection feature is available, as specified by ITU-T Recommendation G.168. A Nonlinear Processor (NLP) feature is used to eliminate residual echo typically caused by non-linearity in the echo path.

1.2 FEATURES

Key features of the LEC library include:

- Simple user interface – only one library file and one header file
- All functions can be called from a C application program
- Compatible with the Microchip C30 Compiler, Assembler and Linker
- Highly optimized assembly code that uses DSP instructions and advanced addressing modes
- LEC for 16, 32, 64 or 128 ms echo delays or "tail lengths" (configurable) for dsPIC DSC devices with greater than 8 KB of RAM
- Compatible with G.168 specifications for in-car applications
- Audio Bandwidth: 0 to 4 kHz at 8 kHz sampling rate
- Convergence Rate: Up to 60 dB/sec., typically greater than 30 dB/sec
- LEC: Up to 70 dB, typically > 40 dB
- Can be used together with the Noise Suppression (NS) library
- Demo application source code is provided with the LEC library
- NLP attenuation level can be adjusted to suit application requirements
- LEC adaptation can be force-enabled or disabled by the user application
- Run-time control of key algorithm parameters is provided
- Tone detection for disabling LEC during test or measurement processes

1.3 HOST SYSTEM REQUIREMENTS

The LEC library requires a PC-compatible system with these attributes:

- Intel® Pentium® class or higher processor, or equivalent
- HTML browser
- 16 MB RAM (minimum)
- 40 MB available hard drive space (minimum)
- Microsoft® Windows® 98, Windows 2000, Windows NT, Windows XP, or Windows 2007

NOTES:

Chapter 2. Installation

This chapter describes the various files in the LEC library and includes instructions for installing the library on your laptop or PC for use with dsPIC DSC device programming tools. Topics covered include:

- Installation Procedure
- LEC Library Files

2.1 INSTALLATION PROCEDURE

To install the library, follow these steps:

1. Double-click `LEC setup.exe`. The License Agreement screen appears.
2. Review the License Agreement and click **I Agree** to continue. The Installation Destination dialog appears.
3. Specify the location (i.e., a directory) where the library should be installed, and then click **Install**.
4. Click **Close** to close the dialog. This completes the LEC library installation.

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

2.2 LEC LIBRARY FILES

The dsPIC® DSC Line Echo Cancellation Library CD creates a directory named `LEC v4.0`. This directory contains these folders:

- `demo` Folder
- `doc` Folder
- `h` Folder
- `lib` Folder

2.2.1 demo Folder

This folder contains files that are required by the dsPIC® DSC Line Echo Cancellation Library Quick Start Demonstration. Table 2-1 describes the files in this folders.

TABLE 2-1: DEMONSTRATION FILES

File Name	Description
dsPIC33F LEC demo 2.hex	Demonstration hexadecimal file for board 2 on dsPIC33F.
dsPIC33F LEC demo 1.hex	Demonstration hexadecimal file for board 1 on dsPIC33F.
dsPIC33F LEC demo.mcp	Demonstration MPLAB Project file for dsPIC33F.
dsPIC33E LEC demo 2.hex	Demonstration hexadecimal file for board 2 on dsPIC33E.
dsPIC33E LEC demo 1.hex	Demonstration hexadecimal file for board 1 on dsPIC33E.
dsPIC33E LEC demo.mcp	Demonstration MPLAB Project file for dsPIC33E.
cleanup.bat	A batch file script for cleaning the intermediate build files.
h\dsPICDEM1_1Plus.h	C header file for the dsPICDEM™ 1.1 Plus development board routines.
h\MEB.h	C header file for the Multimedia Expansion Board (MEB) routines.
h\lcd.h	C header file defining the interface to the LCD driver.
h\G711.h	C header file defining the interface to the G.711 library.
h\lec_api.h	C header file defining the interface to the LEC library
h\UART1Drv.h	C header file defining interface to UART1 driver.
h\UART2Drv.h	C header file defining interface to UART2 driver.
h\SI3000Drv.h	C header file defining the interface to the Si3000 Codec driver.
h\WM8731CodecDrv.h	C header file defining the interface to the WM8731 Codec driver.
libs\leclibv4_33F.a	LEC library archive file for dsPIC33F.
libs\leclibv4_33E.a	LEC library archive file for dsPIC33E.
src\dsPICDEM1_1Plus.c	C source files with routines for the dsPICDEM1.1 Plus Development Board.
src\MEB.c	C source files with routines for the MEB.
src\lcd_strings.c	C source files for LCD display driver.
src\main.c	C source files with the main speech processing routine.
src\UART1Drv.c	C source file with code for the UART1 peripheral.
src\UART2Drv.c	C source file with code for the UART2 peripheral.
src\SI3000Drv.c	C source file with the code for the Si3000 Codec.
src\WM8731CodecDrv.c	C source file with the code for the WM8731 Codec.
src\lcd.s	Assembly routines for communicating with the LCD controller.
src\G711.s	Assembly routines implementing the G.711 library functions.

2.2.2 doc Folder

This folder contains the user's guide for the LEC library. To view this 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

This folder contains an include file for the LEC library, as listed in Table 2-2.

TABLE 2-2: INCLUDE FILE

File Name	Description
`lec_api.h`	Include file that contains the interface to the LEC library. This file must be included in the application to use the library.

2.2.4 lib Folder

This folder contains library archive files for the LEC library as listed in Table 2-3. The archive names are suffixed with the name of the target device family: dsPIC33F or dsPIC33E.

TABLE 2-3: LIBRARY FILE

File Name	Description
`leclibv4_33F.a`	This is the LEC library archive file for dsPIC33F. This file must be included in the application in order to use the library.
`leclibv4_33E.a`	This is the LEC library archive file for dsPIC33E. This file must be included in the application in order to use the library.

NOTES:

Chapter 3. LEC Demonstration

This chapter provides a hands-on demonstration of line echo cancellation in a working application using a dsPIC33F or dsPIC33E Digital Signal Controller (DSC).

3.1 LEC DEMONSTRATION FOR THE dsPIC33E DEVICE FAMILY

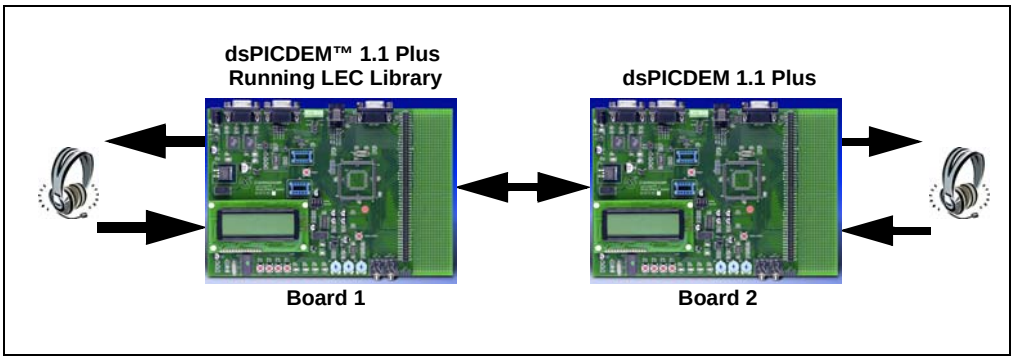
The following topics are covered in this section:

- Demonstration Summary
- Demonstration Setup
- Demonstration Procedure
- Demonstration Code Description

3.1.1 Demonstration Summary

To demonstrate the functionality of the Line Echo Cancellation (LEC) Library, a sample application based on the dsPIC33F DSC, which emulates two voice terminals engaged in communication, is provided with the library. This software requires the use of two dsPICDEM™ 1.1 Plus development boards (not included with the software license), which are set up as shown in Figure 3-1.

FIGURE 3-1: LINE ECHO CANCELLATION DEMONSTRATION



The LEC algorithm is running on board 1. In a real world application, the communication link between board 1 and board 2 would cause a line echo. To emulate this echo condition, board 2 runs a delay routine which emulates a line delay. A set of headsets is connected to both of the boards.

When a person speaks into the headset connected to board 1, the speech signal is sampled through the on-board Si3000 voice band codec and the Data Converter Interface (DCI) module of the dsPIC DSC device. The dsPIC DSC device then transmits the compressed speech signal through its UART1 module and the on-board RS-232 transceiver to board 2.

The dsPIC DSC device on board 2 receives the signal through the on-board RS-232 transceiver and the device's UART1 module. The dsPIC DSC device then outputs the signal on the speaker through its DCI module and on-board Si3000 codec.

The application running on board 2 processes the samples received from board 1, delays them to emulate the line echo, and then adds these samples to the outgoing samples to board 1. If a person is speaking into the microphone connected to board 2, the signal transmitted to board 1 is a combination of the near-end speech and the undesirable line echo (emulated in this case) of the far-end speech. This combination of speech and echo can be heard on the headset connected to board 1. In this example, board 1 represents the near-end and board 2 represents the far-end.

When started, the program initializes with line echo cancellation turned OFF, indicated by LED1 being turned OFF and OFF being written to the LCD screen. With line echo cancellation off, the signal heard in the headset connected to board 1 contains noticeable echo.

Line echo cancellation is enabled by pressing SW1. LED1 is now turned ON and ON is written to the LCD, and the speech signal heard on the headset connected to board 1 becomes echo-free.

The demonstration application program invokes the `LEC_apply` and `LEC_applyNLP` functions from the LEC library to suppress the unwanted line echo mixed with the far-end speech.

3.1.2 Demonstration Setup

The demo application is intended to run on a dsPICDEM™ 1.1 Plus Development Board (not included with the software license).

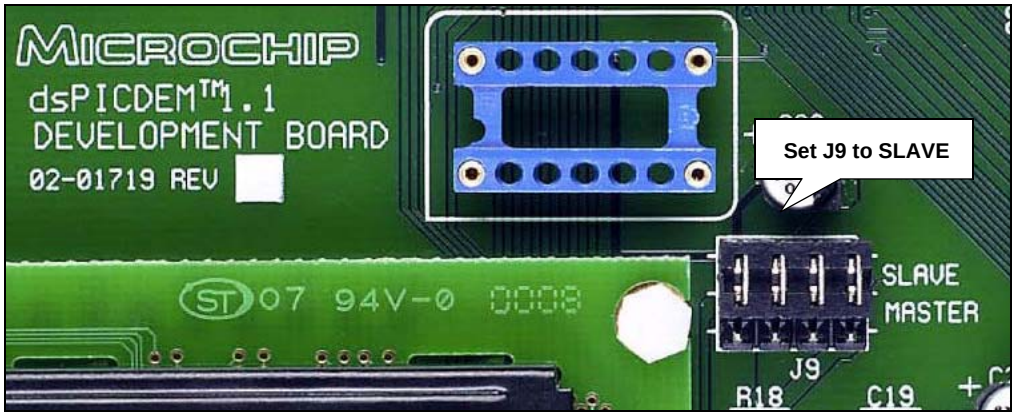
Use the procedures outlined in the following sections to set up the demonstration.

3.1.2.1 CONFIGURE dsPICDEM 1.1 PLUS DEVELOPMENT BOARDS

Before applying power, configure the board:

1. Set jumper J9 (adjacent to the oscillator socket) to the **SLAVE** position (see Figure 3-2). This setting allows the on-board Si3000 codec chip to function as a serial clock Slave.
2. Connect the headset microphone to the MIC IN (J16) jack on board 1.
3. Connect the headset speaker to the SPKR OUT (J17) jack on board 1.
4. Connect the headset microphone to the MIC IN (J16) jack on board 2.
5. Connect the headset speaker to the SPKR OUT (J17) jack on board 2.
6. Connect one end of the DB9M-DB9M Null Modem Adapter to PORTB (J5) on board 1. Then, connect one end of the RS-232 cable to the Null Modem Adapter.
7. Connect the other end of a 6 foot DB9 M/F RS-232 cable to the 'PORTB' (J5) port on board 2.

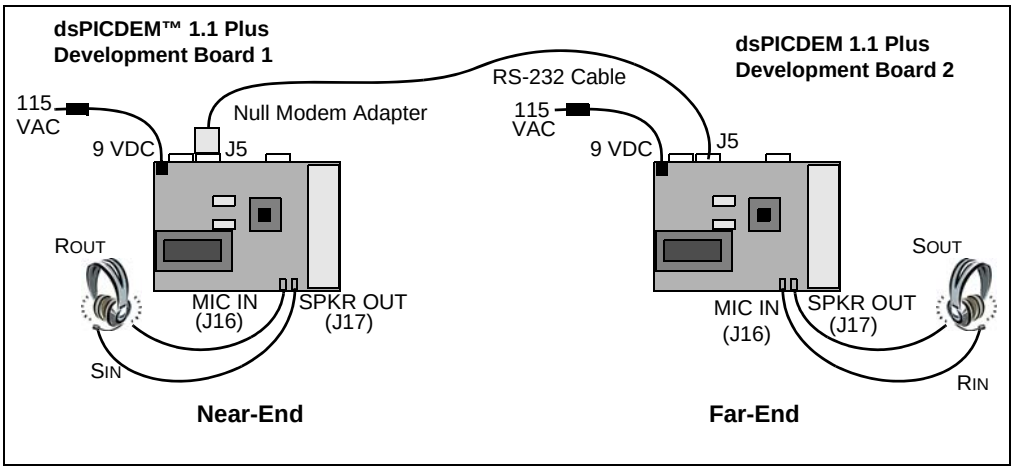
FIGURE 3-2: DEMONSTRATION BOARD SETUP



3.1.2.2 SET UP THE DEMONSTRATION

After both boards are properly configured, attach the headsets and interconnect the boards, as shown in Figure 3-3.

FIGURE 3-3: CONNECT dsPICDEM™ 1.1 BOARDS



3.1.2.3 PROGRAM THE dsPIC DSC DEVICE

Use this process to load the LEC demo into the dsPIC DSC device on the dsPICDEM 1.1 Plus Development Board.

1. On your PC, launch MPLAB IDE and open the `dsPIC33F LEC demo.mcp` project located in the `demo` folder. For more information on using MPLAB IDE, refer to the *"MPLAB® IDE User's Guide"* (DS51025).
2. Import the project `.hex` file: *File>Import>dsPIC33F LEC demo 1.hex* for board 1 or *File>Import> dsPIC33F LEC demo 2.hex* for board 2.
3. Select *Programmer>Connect* to link the MPLAB ICD 3 to the dsPIC DSC device target. 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 has succeeded.
5. Connect the MPLAB ICD 3 to the dsPICDEM 1.1 Plus 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). When you have done this, you will see the LEC information in the LCD display.
8. Repeat steps 2 through 7 for the second board.
9. Make sure that the two boards are not located too close to each other to avoid any acoustic coupling between the two boards.

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

3.1.3 Demonstration Procedure

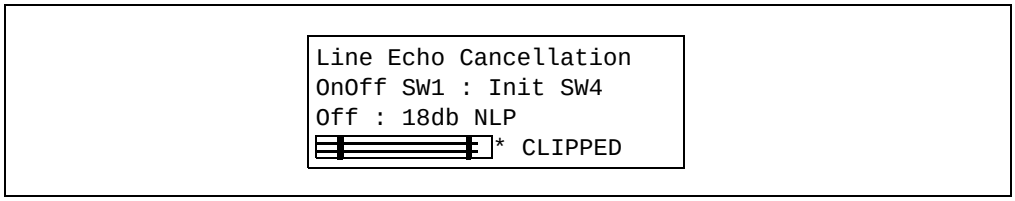
After the demonstration application has been programmed into both devices the application is now ready to run. Use the following procedure to run the demonstration:

1. Press the Reset button on board 1. Wait until LED1 through LED4 on board 1 turn ON. Now reset board 2. Wait until LED1 through LED4 on board 1 and board 2 turn OFF. This indicates that the boards are synchronized and the demo is running.
2. Put on the headset connected to the dsPICDEM 1.1 Plus Development Board 1.
3. Begin speaking into the microphone input of the headset. You should observe the top bar of the VU meter rising and falling in time with your speech. This will be mimicked by the second bar of the VU meter on board 1. On the speaker output of the headset, you should be able to hear an echo of your own speech.
4. If desired, have someone simultaneously speak into the microphone connected to dsPICDEM 1.1 Plus Development Board 2. In this case, you will hear the other person's speech as well as an echo of your own speech.
5. Press the switch marked 'SW1' on dsPICDEM 1.1 Plus Development Board 1. Observe that the LED1 on board 1 turns ON, indicating that the LEC algorithm is active.
6. Again, speak into the microphone of the headset. You should no longer hear the echo of your own speech.
7. To observe line echo cancellation during double talk, let a person speak into the microphone connected to the dsPICDEM 1.1 Plus Development Board 2. You will only hear the other person's speech, free of the echo of your own speech. You will be able to experience the difference in ease of conversation by switching the echo canceller ON and OFF while you and the other person are talking normally.

To experiment with different values of echo tail length, simply change the `LEC_ECHO_DELAY` constant defined in the `lec_api.h` include file, rebuild `LEC demo.mcp`, reprogram board 1 and rerun the demo application. The demo application relays the state of operation via the LEDs and the LCD.

While the application is loading and initializing the on-chip and off-chip peripherals, a boot screen is displayed. This is then switched to the run screen (see Figure 3-4).

FIGURE 3-4: DEMONSTRATION RUN-TIME LCD SCREEN ON BOARD 1



The run-time screen displays the following:

1. The name of the algorithm.
2. SW1 is used to turn LEC ON and OFF. SW4 is used to reinitialize the LEC algorithm.
3. The current state of the algorithm (OFF) and the NLP level selected.
4. A VU meter showing the input levels. The top bar represents the `lecIn` buffer, the bottom bar represents the `refIn` buffer. The bands show an acceptable input range. CLIPPED is displayed when either input signal is too large.

   The amount of Nonlinear Processing can be increased in 6 dB steps (up to 90 dB) by pressing SW3. It can be reduced in 6 dB steps (down to 0 dB) by pressing SW2. The default level for most applications is 18 dB.

   The LEC can be reinitialized by pressing SW4.

3.1.4 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 demo application. This main function allocates all the variables and arrays in data memory that are needed for DCI data buffering, as well as the blocks of data memory that need to be allocated for the LEC library functions.

The main function calls the `LEC_init()` function from the LEC library, which initializes the LEC algorithm to its default state.

The main function also calls the `SI3000_open()` function to initialize the DCI module, the Si3000 codec, and the DCI interrupt. The DCI module acts as a Master and drives the serial clock and frame synchronization lines. The Si3000 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.

Subsequently, the function initializes the Si3000 codec. The codec is reset, by connecting the RF6 pin of the dsPIC DSC device to the Reset pin of the Si3000, holding RF6 low for 100 cycles and then bringing it high. The codec is configured for a sample rate of 8 kHz. The MIC Gain is set to 10 dB and the Receive Gain is set to 0 dB. Both speakers are set to Active and the Transmit Gain is set to 0 dB. The Analog Attenuation parameter is set to 0 dB. After initializing all of the Si3000 control registers, a delay is introduced for calibration of the Si3000 to occur. Finally, the DCI interrupt is enabled.

UART initialization and data processing is performed by the `UART1_open()` function. The UART module is configured to generate an interrupt for every byte transmitted or received. The UART module is run at a baud rate of ~250000 bps, with an 8-bit, no parity, 1 Stop bit data format (8-N-1). In the UART Transmit and Receive Interrupt Service Routines, the corresponding interrupt flag is cleared, data is either written to U1TXREG, or read from U1RXREG and saved in a circular buffer. The UART data is converted from µ-Law to 16-bit linear and stored in `lecIn`, which is the input data frame to `LEC_apply()`.

The codec driver is polled for a full frame of data. When the codec driver indicates that a full frame of data is available, the contents of the codec data buffers are copied into the `refIn` array and the `LEC_apply()` function from the LEC Library is called with `lecIn` as the input data frame. The output of the `LEC_apply()` function is stored in `lecOut`. The `refIn` data buffer is encoded and transferred to the UART for transmission to board 2. The `lecOut` buffer is played out through the Si3000 codec on board 1.

The display on the LCD is made possible by initialization of the SPI module in the `InitSPI` function, and LCD driver functions and LCD string definitions present in the `lcd.s` and `lcd_strings.c` files, respectively.

To toggle LEC ON or OFF, external interrupts for SW1 are enabled. In the main loop, the value of `applyLEC` is read and passed to `LEC_apply()` as the enable flag. If `applyLEC` is off, LEC is still called, but the input/output buffer is not changed. This enables LEC to maintain adaptation to changes in echo path, while it is not enabled.

3.2 LEC DEMONSTRATION FOR THE dsPIC33E DEVICE FAMILY

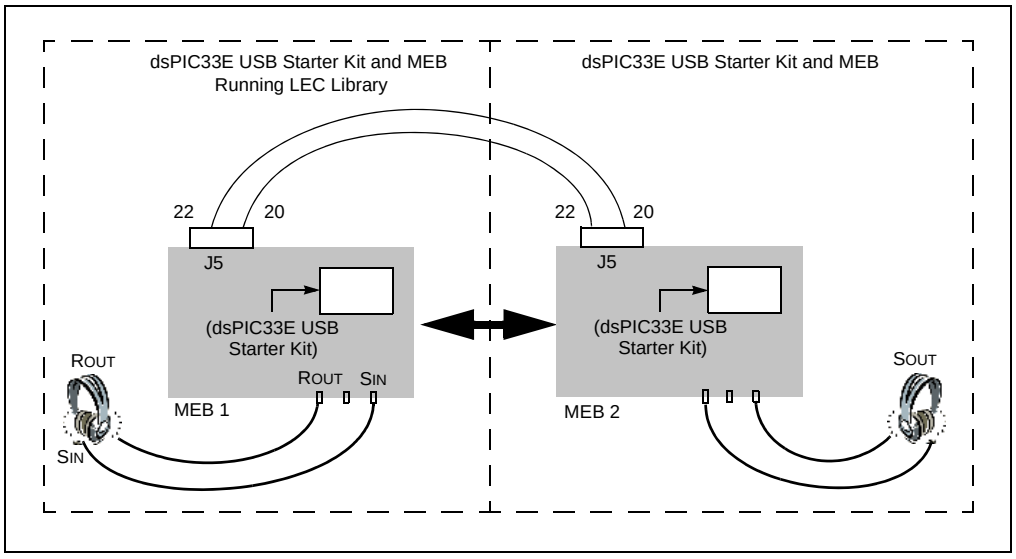
The following topics are covered in this section:

- Demonstration Summary
- Demonstration Setup
- Demonstration Procedure
- Demonstration Code Description

3.2.1 Demonstration Summary

To demonstrate the functionality of the LEC library, a sample dsPIC33E-based application emulating two speaker phones engaged in voice communication is provided with the library. This software requires the use of two dsPIC33E USB Starter Kits and two MEBs (not included with the software license), which are set up as shown in Figure 3-5.

FIGURE 3-5: LINE ECHO CANCELLATION DEMONSTRATION



The LEC algorithm is running on board 1. In a real world application, the communication link between board 1 and board 2 would cause a line echo. To emulate this echo condition, board 2 runs a delay routine that emulates a line delay. A headset is connected to both of the boards.

When a person speaks into the headset connected to board 1, the speech signal is sampled through the on-board WM8731 audio codec and the Data Converter Interface (DCI) module of the dsPIC DSC device. The dsPIC DSC device then transmits the compressed speech signal through its UART2 module and the on-board I/O expansion connector to board 2.

The dsPIC DSC device on board 2 receives the signal through the on-board RS-232 transceiver and the device's UART2 module. The dsPIC DSC device then outputs the signal on the speaker through its DCI module and on-board WM8731 audio codec.

The application running on board 2 processes the samples received from board 1, delays them to emulate the line echo, and then adds these samples to the outgoing samples to board 1. If a person is speaking into the microphone connected to board 2, the signal transmitted to board 1 is a combination of the far-end speech and the undesirable line echo (emulated in this case) of the near-end speech. This combination of speech and echo can be heard on the headset connected to board 1. In this example, board 1 represents the near-end and board 2 represents the far-end.

When started, the program initializes with LEC turned OFF, indicated by LED3 being turned OFF on the dsPIC33E USB Starter Kit. With LEC OFF, the signal heard in the headset connected to board 1 contains noticeable echo.

The LEC is enabled by pressing the switch, SW1, on the dsPIC33E USB Starter Kit. LED3 is now turned ON and the speech signal heard on the headset connected to board 1 becomes echo-free.

The demonstration application program invokes the `LEC_apply` and `LEC_applyNLP` functions from the LEC library to suppress the unwanted near-end echo mixed with the far-end speech.

3.2.2 Demonstration Setup

The demo application is intended to run on a dsPIC33E USB Starter Kit and MEB (not included with the software license). Use the procedures outlined in the following sections to set up the demonstration.

3.2.2.1 CONFIGURE MEB AND dsPIC33E USB STARTER KIT

Before applying power, you need to configure the board:

1. Insert a dsPIC33E USB Starter Kit into the starter kit connector on MEB 1, and another one into MEB 2.
2. Connect the headset microphones to the microphone jacks on MEB 1 and MEB 2.
3. Connect the headset speakers to the headphone jacks on MEB 1 and MEB 2.
4. Connect each of the two dsPIC33E starter kits to a PC using the USB 1-to-mini B cable provided with the starter kit.
5. Using a pair of single-strand wires, connect pin 20 of the I/O expansion connector (J5) on MEB 1 with pin 22 of the I/O expansion connector on MEB 2. Also, connect pin 22 of the I/O expansion connector on MEB 1 with pin 20 on MEB 2.

3.2.2.2 PROGRAM THE dsPIC DSC DEVICE

Use this process to load the LEC demonstration into the dsPIC DSC device on the dsPICDEM 1.1 Plus Development Board.

1. On your PC, launch MPLAB IDE and open the `dsPIC33E LEC demo.mcp` project located in the `demo` folder. For more information on using MPLAB IDE, refer to the *"MPLAB® IDE User's Guide"* (DS51025).
2. Select **Starter Kit on Board** as the programmer.
3. Import the project hexadecimal file: *File>Import>dsPIC33E LEC demo 1.hex* for board 1 or *File>Import>dsPIC33E LEC demo 2.hex* for board 2.
4. Select *Programmer>Connect* to link to the dsPIC DSC target device on board 1. The Output window confirms that the target device is ready.
5. Select *Programmer>Program*. The Output window displays the download process and indicates that the programming has succeeded.
6. Repeat steps 3 through 5 for the second board.

Note: After programming each device, unplug and reconnect the USB cable to the Starter Kit, to ensure that the WM8731 audio codec can be reconfigured.

3.2.3 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 MPLAB IDE, reset and run the program in board 1. Wait until LED1 through LED3 on board 1 turn ON. Now reset and run the board 2. Wait until LED1 through LED3 on board 1 and board 2 turn OFF. This indicates that the boards are synchronized and the demo is running.

> **Note:** If only one PC is available for running the demo, 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).

2. Put on the headset connected to board 1.
3. Begin speaking into the microphone input of the headset. On the speaker output of the headset, you should be able to hear an echo of your own speech.
4. If desired, have someone simultaneously speak into the microphone connected to board 2. In this case, you will hear the other person's speech as well as an echo of your own speech.
5. Press the switch, SW1, on the dsPIC33E USB Starter Kit. Observe that the LED1 on board 1 turns ON, indicating that the LEC algorithm is active.
6. Again, speak into the microphone of the headset connected to board 1. You should no longer hear the echo of your own speech.
7. To observe line echo cancellation during double talk, let a person simultaneously speak into the microphone connected to board 2. You will only hear the other person's speech, free of the echo of your own speech. You will be able to experience the difference in ease of conversation by switching the echo canceller ON and OFF while you and the other person are talking normally.

To experiment with different values of echo tail length, change the `LEC_ECHO_DELAY` constant defined in the `ec_api.h` include file, rebuild `LEC demo.mcp`, reprogram board 1 and rerun the demo application. The demo application relays the state of operation through the LEDs.

The amount of nonlinear processing can be increased in 6 dB steps (up to 90 dB) by pressing SW3. It can be reduced in 6 dB steps (down to 0 dB) by pressing SW2. The default level for most applications is 18 dB. The LEC can be reinitialized by pressing switch, S1, on the LCD side of the MEB.

3.2.4 Demonstration Code Description

The demonstration code runs on a dsPIC33E 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 demo application. This main function allocates all the variables and arrays in data memory that are needed for DCI data buffering, as well as the blocks of data memory that need to be allocated for the LEC library functions.

The main function calls the `LEC_Init` function from the LEC library, which initializes the LEC algorithm to its default state.

The main function also calls the `WM8731Init()` function to initialize the DCI 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, transmit slots and two receive slots are used in this demonstration.

Subsequently, the `WM8731Start()` function is used to enable the DCI module and I^2C module. The codec is configured for a sample rate of 8 kHz.

The UART initialization and data processing is performed by the `UART2_open()` function. The UART module is configured to generate an interrupt for every byte transmitted or received. The UART module is run at a baud rate of ~250000 bps, with an 8-bit, no parity, 1 Stop bit data format (8-N-1). In the UART Transmit and Receive ISR, the corresponding interrupt flag is cleared, data is either written to U2TXREG, or read from U2RXREG and saved in a circular buffer.

The codec driver is polled for a full frame of data. When the codec driver indicates that a full frame of data is available, the contents of the codec data buffers are copied into the `refIn` array and the `LEC_apply()` function from the LEC Library is called with `lecIn` as the input data frame. The output of the `LEC_apply()` function is stored in `lecOut`. The `refIn` data buffer is encoded and transferred to the UART for transmission to board 2. The `lecOut` buffer is played out through the Si3000 codec on board 1.

In the main loop, the value of `applyLEC` is read and passed to `LEC_apply()` as the enable flag. If `applyLEC` is '`0`', the line echo cancellation is still called, but the input/output buffer is not changed. This enables the line echo cancellation algorithm to maintain adaptation to changes in echo path, while it is not enabled.

NOTES:

Chapter 4. Application Programming Interface (API)

This chapter describes in detail the Application Programming Interface to the dsPIC DSC Line Echo Cancellation Library. Topics covered include:

- Adding the Line Echo Cancellation Library to an Application
- Memory Model Compile Options
- LEC Algorithm Overview
- Library Usage
- Resource Requirements
- Line Echo Cancellation Library API Functions
- Application Tips

4.1 ADDING THE LINE ECHO CANCELLATION LIBRARY TO AN APPLICATION

To use the LEC library in an application, the library archive must be added to the application project workspace, and the `lec_api.h` header file must be included in application code. This file can be copied from the h folder (located in the installation directory) to the application project folder.

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

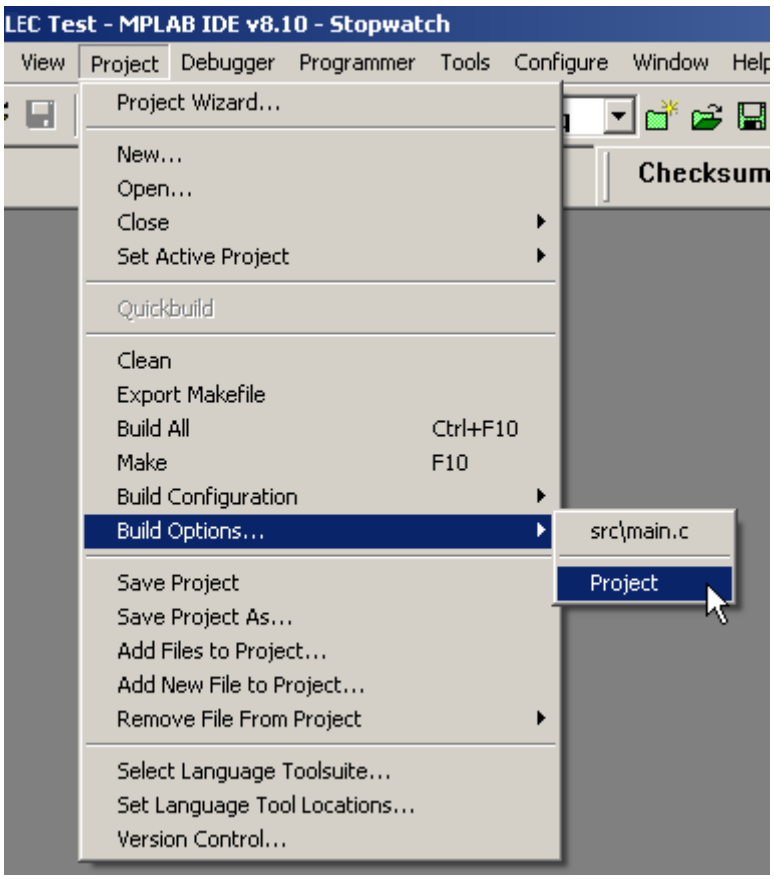
1. In the application MPLAB workspace, right-click **Library Files** in the Project Window and select **Add files**.
2. Browse to the location of the desired LEC library archive file available in the `libs` folder in the installation directory.
3. Select the desired file and click **Open**.
4. The library is now added to the application. Verify that the library archive is shown in the MPLAB project.

4.2 MEMORY MODEL COMPILE OPTIONS

While using the LEC library, it may be necessary to direct the compiler to use a large memory model, as described in the following steps.

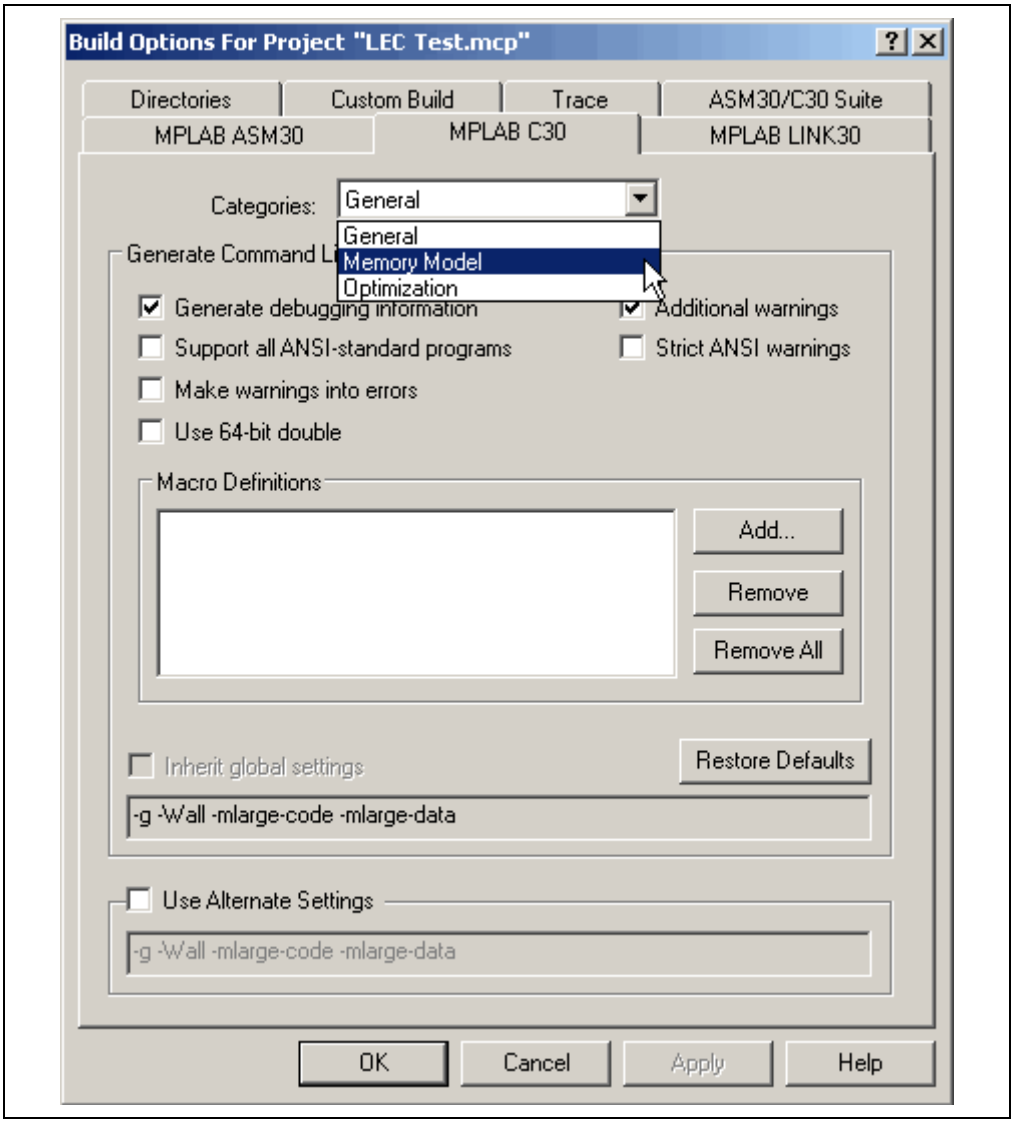
1. From the MPLAB IDE menu, select *Project>Build Options>Project* as shown in Figure 4-1.

FIGURE 4-1:



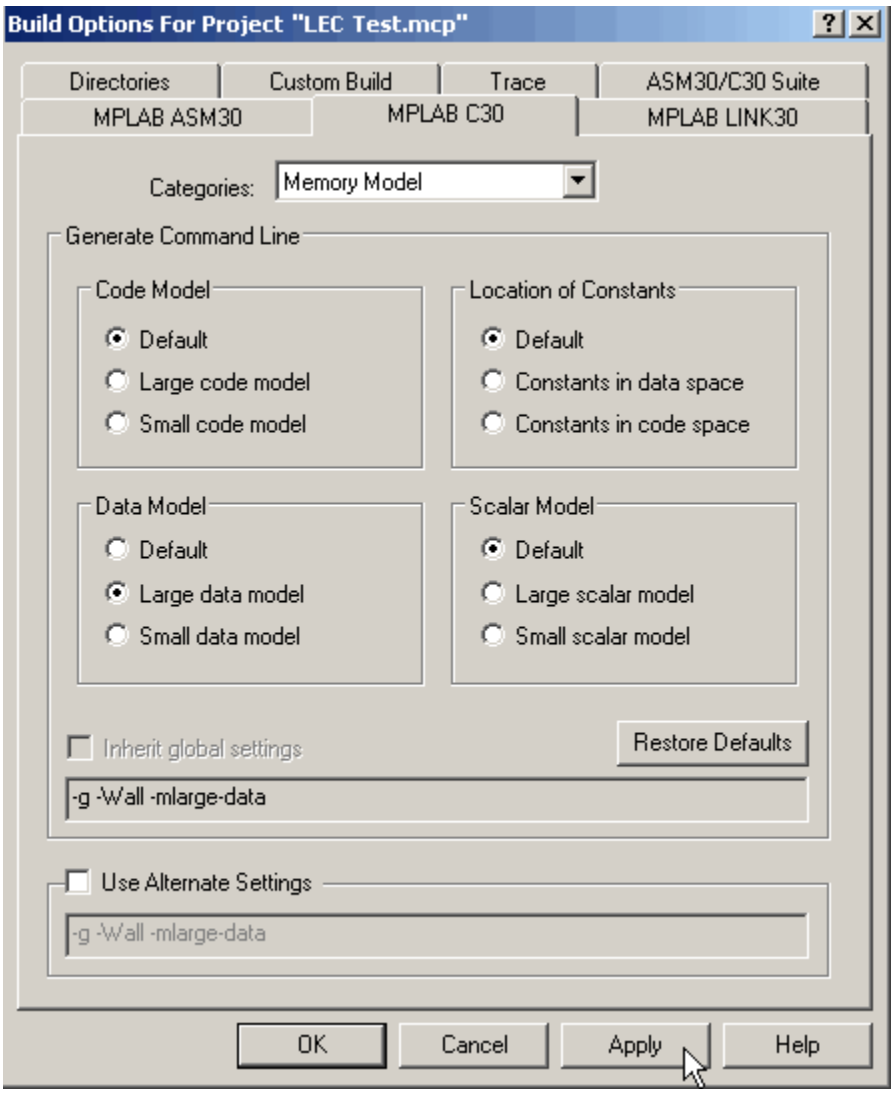
2. Click the **MPLAB C30** tab and set the following options:
   a) From the Categories drop-down list, select **Memory Model**, as shown in Figure 4-2.

FIGURE 4-2:



b) In the Data Model section, select **Large data model**, as shown in Figure 4-3.

FIGURE 4-3:



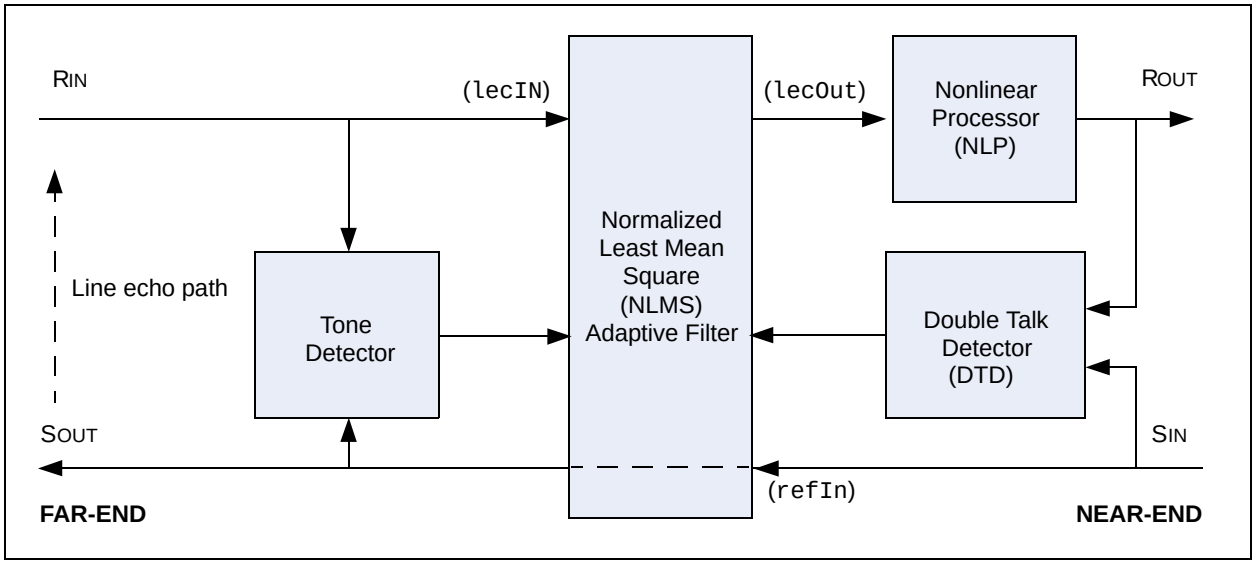
3. Click **Apply**, and then click **OK**.

This completes the procedure.

4.3 LEC ALGORITHM OVERVIEW

This section describes the Line Echo Cancellation (LEC) algorithm. A conceptual block diagram illustrating the operation of the LEC algorithm is shown in Figure 4-4.

FIGURE 4-4: LEC ALGORITHM FUNCTIONAL BLOCKS



The LEC algorithm can be divided into these functions:

- Normalized Least Mean Square (NLMS) Adaptive Filter
- Double Talk Detector (DTD)
- Nonlinear Processor (NLP)
- Tone Detector

A typical LEC system involves these four signals:

- Far-end speech receive input (RIN)
- Near-end speech send output (SOUT)
- Far-end speech output (ROUT), usually sent to a local speaker
- Near-end speech input (SIN), usually received from a local microphone

Line echo in communication systems is caused by imperfect termination of communication links. A typical example of this is the 2-wire to 4-wire hybrid used in telephone networks. This echo is independent of any acoustic coupling that may exist at the far-end. Therefore, even if the microphone and speaker at the far-end are acoustically isolated from each other, the near-end still hears an echo of the near-end microphone signal.

Figure 4-4 also shows the nomenclature that is used to denote the different signals in the LEC API. `lecIn` refers to the RIN signal and is the signal name from which echo is removed. `refIn` refers to the SIN signal and is the reference signal to the LEC algorithm. `lecOut` is the echo suppressed output of the LEC algorithm.

The LEC algorithm consists of an adaptive filter and various associated control functions, which eliminate the line echo. This algorithm typically operates at the communication node that is receiving the echo. The control functions used in the LEC algorithm, in conjunction with the NLMS adaptive filter, are Double Talk Detector, Nonlinear Processor, and Tone Detector.

4.3.1 Normalized Least Mean Square (NLMS) Adaptive Filter

NLMS is the fundamental adaptation algorithm used for estimating and canceling out the line echo. This filter tries to minimize the mean square error between the SIN and RIN signals. Under conditions where there is no double talk, this will result in a set of filter coefficients that approximate the line echo path between the speaker and the microphone. The filter thereby produces an echo estimate of SIN, which is then subtracted from the RIN signal.

4.3.2 Nonlinear Processor (NLP)

The LEC algorithm by itself may not be capable of adequately modeling echo paths that generate significant levels of nonlinear distortion. This necessitates the usage of a Nonlinear Processor (NLP). The function of the NLP is to substantially suppress the residual echo level which remains at the output of the NLMS adaptive filter, so that a very-low returned echo level can be achieved even if the echo path is nonlinear. The NLP is located in the receiver path between the output of the NLMS filter and the ROUT port of the system. The NLP basically attenuates low-level signals (which are assumed to be residual echo) and passes high-level signals (which are assumed to be desirable near-end speech).

The LEC library offers two functions for using the NLP. The `LEC_applyNLP()` function applies the NLP action to the input buffer. The `LEC_setNLPLevel()` function varies the level of attenuation.

4.3.3 Tone Detector

The tone detector listens on both channels (RIN and SIN) for specific tones (2100 Hz tone with periodic phase reversals which occur every 450 ± 25 ms) which are generated by network operators' Automatic Test and Measurement Equipment. If a tone of this type is detected on either channel, the echo canceller is disabled. It is re-enabled when the tone ceases. Low level tones are ignored. Tone detection is typically utilized in networking applications.

4.3.4 Double Talk Detector (DTD)

Double talk is the condition that occurs as a result of two talkers on both sides (RIN and SIN) talking at the same time. During double talk, the signal SIN acts like uncorrelated noise and may cause the coefficients of the NLMS adaptive filter to diverge, thereby failing to effectively cancel the line echo. To prevent such a condition, a Double Talk Detector (DTD) is used to inhibit adaptation of the filter during periods of simultaneous far-end and near-end speech. The DTD also inhibits the operation of the NLP to prevent loss of far-end speech. In this algorithm, an energy-based double talk detector is used, in which double talk is detected when Average Energy of SOUT > Average Energy of RIN.

The LEC library offers two functions, `LEC_setDoubleTalkHangover()` and `LEC_setAdaptionHangover()`, to control the amount of DTD hangover. The double talk hangover represents the number of frames after double talk has been detected for which the LEC algorithm will wait before resuming application of NLP. For example, if the hangover value is 6, the algorithm will wait for 6 frames before applying NLP again.

The adaptation hangover is controlled by `LEC_setAdaptionHangover()`. The default value for this function is '`1`', so that adaptation resumes one frame after the end of double talk.

4.4 LIBRARY USAGE

The LEC algorithm has been designed to be usable in a re-entrant environment. This enables the algorithm to process many independent channels of audio, each channel having its own setting and parameters.

The following coding steps need to be performed to enable use of the LEC library:

1. **Set the Echo Tail Length:** In the file `lec_api.h`, set the `LEC_ECHODELAY` value to the desired echo tail length. The valid values are 8, 16, 32, 64 and 128 ms. This coding step is shown in Example 4-1.

EXAMPLE 4-1:

```
/*************** user-defined constants - start ********************/
#define LEC_ECHO_DELAY 64         /* echo path length (Step 1) */
/*************** user-defined constants - end **********************/
#define LEC_ENABLE 1
#define LEC_DISABLE 0
#define LEC_TRUE 1
#define LEC_FALSE 0
#define LEC_FRAME 80              /* length of input block */
#define LEC_ORDER (LEC_ECHO_DELAY*8)
```

Steps 2 through 7 should be performed in the user application. Refer to Example 4-2 for the actual code required.

2. **Allocate memory for input, output, and reference data arrays:** These arrays should be placed in Y memory.
3. **Allocate the memory for the LEC algorithm state holder:** The LEC state holder has two components, one in X memory and the other in Y memory. The state holder in X memory is a long type array of size `LEC_X_MEM_SIZE_LONG`, starting at an address aligned at boundary of four bytes. The state holder in Y memory is a long type array of size `LEC_Y_MEM_SIZE_LONG`, starting at an address aligned at boundary of four bytes. Every audio channel to be processed will require its own state holder.
4. **Allocate the memory for the LEC algorithm X and Y scratch memories:** The X scratch memory is an integer array in X memory aligned at an address boundary of two bytes. The Y scratch memory is an integer array in Y memory aligned at an address boundary of 512 bytes. Multiple audio channels can share the same scratch memories.

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

5. **Initialize the Line Echo Cancellation algorithm state for each audio channel:** Use the `LEC_init()` function for initializing the LEC state for each audio channel.
6. **Apply LEC to an audio frame:** Use the `LEC_apply()` function to perform echo cancellation on an audio frame. If a frame is not required to be processed by the LEC algorithm, the function should still be called with the enable parameter set to `LEC_FALSE`. This will allow the LEC algorithm to continue adapting to the echo in the audio frame. The audio frame stays unaffected.
7. **Use the NLP function:** Use the `LEC_applyNLP()` function to suppress residual echo.

EXAMPLE 4-2:

```
// Data arrays for LEC Channel 1
short lecIn1[LEC_FRAME]   _YBSS(2);                              /* Step 2 */
short lecOut1[LEC_FRAME]  _YBSS(2);                              /* Step 2 */
short refIn1[LEC_FRAME]   _YBSS(2);                              /* Step 2 */

// Data arrays for LEC Channel 2
short lecIn2[LEC_FRAME]   _YBSS(2);                              /* Step 2 */
short lecOut2[LEC_FRAME]  _YBSS(2);                              /* Step 2 */
short refIn2[LEC_FRAME]   _YBSS(2);                              /* Step 2 */

// Channel 1 memory structures
long lecStateMemX1[LEC_X_MEM_SIZE_LONG]  _XBSS(4);               /* Step 3 */
long lecStateMemY1[LEC_Y_MEM_SIZE_LONG]  _YBSS(4);               /* Step 3 */

// Channel 2 memory structures.
long lecStateMemX2[LEC_X_MEM_SIZE_LONG]  _XBSS(4);               /* Step 3 */
long lecStateMemY2[LEC_Y_MEM_SIZE_LONG]  _YBSS(4);               /* Step 3 */

// Each instance can share the same X and Y scratch memory.
short lecScratchX[LEC_X_SCRATCH_MEM_SIZE_SHORT]  _XBSS(2);       /* Step 4 */
short lecScratchY[LEC_Y_SCRATCH_MEM_SIZE_SHORT]  _YBSS(512);     /* Step 4 */
.
.
.
void main()
{
    LEC_init(lecStateMemX1, lecStateMemY1, lecScratchX, lecScratchY, ENABLE_NLP1, ECHO_DELAY1);
    /* Step 5 */

    LEC_init(lecStateMemX2, lecStateMemY2, lecScratchX, lecScratchY, ENABLE_NLP1, ECHO_DELAY2);
    /* Step 5 */

    While(1)
    {
        LEC_apply(lecStateMemX1, lecStateMemY1, refIn1, lecIn1, lecOut1, LEC_FRAME, LEC_TRUE);
        /* Step 6 */

        LEC_applyNLP(lecStateMemX1, lecOut1, LEC_FRAME, LEC_NLP_SHIFT1);    /* Step 7 */

        LEC_apply(lecStateMemX2, lecStateMemY2, refIn2, lecIn2, lecOut2, LEC_FRAME, LEC_TRUE);
        /* Step 6 */

        LEC_applyNLP(lecStateMemX2, lecOut2, LEC_FRAME, LEC_NLP_SHIFT2);    /* Step 7 */
    }
}
```

4.5 RESOURCE REQUIREMENTS

The Line Echo Cancellation Library requires the following resources while running on the dsPIC DSC device.

TABLE 4-1: PROGRAM MEMORY USAGE

Type	Size (bytes)	Section
Code in Program Memory	6339 (dsPIC30F/dsPIC33F) 6324 (dsPIC33E)	`.liblec`
Tables in Program Memory	0	`.const`
Total Program Memory	6339 (dsPIC30F/dsPIC33F) 6324 (dsPIC33E)	—

TABLE 4-2: DATA MEMORY USAGE (64 ms ECHO TAIL LENGTH)

Function	Size (bytes)	Alignment	Section
`lecStateMemX`	2944	4	X data memory
`lecStateMemY`	2048	4	Y data memory
`lecScratchX`	2208	2	X data memory
`lecScratchY`	2048	512	Y data memory
`lecIn`	160	2	Y data memory
`lecOut`	160	2	Y data memory
`refIn`	160	2	Y data memory
Tables in Data Memory	28	2	X data memory
Total Data Memory	9756	—	—

TABLE 4-3: ESTIMATED DYNAMIC MEMORY USAGE

Section	Size (bytes)
Heap	0
Stack	< 300

TABLE 4-4: COMPUTATIONAL SPEED

Function	Echo Tail	MIPS	Typical Call Frequency
`LEC_init()`	All	< 0.5	Once
`LEC_apply() + LEC_applyNLP()`	8	2.4	10 ms
`LEC_apply() + LEC_applyNLP()`	16	3	10 ms
`LEC_apply() + LEC_applyNLP()`	32	4	10 ms
`LEC_apply() + LEC_applyNLP()`	64	6.2	10 ms
`LEC_apply() + LEC_applyNLP()`	128	10.6	10 ms
All other functions	All	Minimal	As required

Note: The MIPS requirements in Table 4-4 reflect Tone Detection disabled. Enabling the Tone Detection feature requires an additional 1 MIPS.

4.5.1 Data Format

The data type of `lecIn` and `refIn` can be 10-, 12- or 16-bit linear PCM data. The LEC algorithm automatically adjusts for the data format used.

4.6 LINE ECHO CANCELLATION LIBRARY API FUNCTIONS

This section lists and describes the Application Programming Interface functions that are available in the dsPIC DSC Line Echo Cancellation Library. The functions are listed below followed by their individual detailed descriptions.

- LEC_init
- LEC_relocateXScratchMem
- LEC_relocateYScratchMem
- LEC_apply
- LEC_applyNLP
- LEC_setNLPLevel
- LEC_getNLPLevel
- LEC_setNLPHangover
- LEC_getNLPHangover
- LEC_setNLPThreshold
- LEC_getNLPThreshold
- LEC_setAdaptationHangover
- LEC_getAdaptationHangover
- LEC_setToneDetect
- LEC_getToneDetect
- LEC_setDoubleTalkDetect
- LEC_getDoubleTalkDetect
- LEC_setInhibitAdaptation
- LEC_getInhibitAdaptation
- LEC_TRUE
- LEC_FALSE
- LEC_ECHO_DELAY
- LEC_FRAME
- LEC_ENABLE_DEFAULT
- LEC_ENABLE_NLP_DEFAULT
- LEC_ENABLE_DT_DETECT_DEFAULT
- LEC_ENABLE_TONE_DETECT_DEFAULT
- LEC_INHIBIT_ADAPT_DEFAULT
- LEC_NLP_SUP_THRESHOLD_DEFAULT
- LEC_NLP_HANGOVER_FRAMES_DEFAULT
- LEC_NLP_SHIFT_DEFAULT
- LEC_DT_HANGOVER_FRAMES_DEFAULT
- LEC_X_MEM_SIZE_LONG
- LEC_Y_MEM_SIZE_LONG
- LEC_X_SCRATCH_MEM_SIZE_SHORT
- LEC_Y_SCRATCH_MEM_SIZE_SHORT

LEC_init

Description

Initializes the LEC algorithm, which is called only once.

Include

`lec_api.h`

Prototype

```
void LEC_init(long* ptrStateX, long* ptrStateY, short*
scratchX, short* scratchY, int NLPflag, int echo_tail);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`ptrStateY`	a pointer to the Y memory for this instance of LEC
`scratchX`	a pointer to the scratch memory in X space
`scratchY`	a pointer to the scratch memory in Y space
`NLPflag`	a flag to enable or disable NLP
`echo_tail`	length of echo path in milliseconds, assuming 8 kHz sampling rate. This value should be equal to `LEC_ECHO_DELAY`.

Return Value

None.

Remarks

None.

Code Example

```
long lecStateMemX[LEC_X_MEM_SIZE_LONG]          _XBSS(4);
long lecStateMemY[LEC_Y_MEM_SIZE_LONG]          _YBSS(4);
short lecScratchX[LEC_X_SCRATCH_MEM_SIZE_SHORT] _XBSS(2);
short lecScratchY[LEC_Y_SCRATCH_MEM_SIZE_SHORT] _YBSS(512);

LEC_init(lecStateMemX, lecStateMemY, lecScratchX, lecScratchY,
   LEC_TRUE, LEC_ECHO_DELAY);
```

LEC_relocateXScratchMem

Description

Changes the X scratch memory that is used by the LEC algorithm.

Include

`lec_api.h`

Prototype

```
void LEC_relocateXScratchMem(long* ptrStateX, short* scratchX)
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`scratchX`	a pointer to the new X scratch memory

Return Value

None.

Remarks

None.

Code Example

```
long lecStateMemX[LEC_X_MEM_SIZE_LONG]_XBSS(4);
long lecStateMemY[LEC_Y_MEM_SIZE_LONG]_YBSS(4);
short lecScratchX[LEC_X_SCRATCH_MEM_SIZE_SHORT]_XBSS(2);
short lecScratchY[LEC_Y_SCRATCH_MEM_SIZE_SHORT]_YBSS(512);
short lecScratchX2[LEC_X_SCRATCH_MEM_SIZE_SHORT]_XBSS(2);
LEC_init(lecStateMemX, lecStateMemY, lecScratchX, lecScratchY,
    LEC_TRUE, LEC_ECHO_DELAY);
LEC_relocateXScratchMem(lecStateMemX, lecScratchX2);
/* LEC is now using lecScratchX2 for its X scratch memory */
```

LEC_relocateYScratchMem

Description

Changes the Y scratch memory that is used by the LEC algorithm.

Include

`lec_api.h`

Prototype

```
void LEC_relocateYScratchMem(long* ptrStateX, short* scratchY)
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`scratchY`	a pointer to the new Y scratch memory

Return Value

None.

Remarks

None.

Code Example

```
long lecStateMemX[LEC_X_MEM_SIZE_LONG]_XBSS(4);
long lecStateMemY[LEC_Y_MEM_SIZE_LONG]_YBSS(4);
short lecScratchX[LEC_X_SCRATCH_MEM_SIZE_SHORT]_XBSS(2);
short lecScratchY[LEC_Y_SCRATCH_MEM_SIZE_SHORT]_YBSS(512);
short lecScratchY2[LEC_Y_SCRATCH_MEM_SIZE_SHORT]_YBSS(512);
LEC_init(lecStateMemX, lecStateMemY, lecScratchX, lecScratchY,
    LEC_TRUE, LEC_ECHO_DELAY);
LEC_relocateXScratchMem(lecStateMemX, lecScratchY2);
/* LEC is now using lecScratchY2 for its Y scratch memory */
```

LEC_apply

Description

Applies line echo cancellation to the current frame of data.

Include

`lec_api.h`

Prototype

```
void LEC_apply(long* ptrStateX, long* ptrStateY, short* refIn,
   short* lecIn, short* lecOut, short frame, short lec_enable);
```

Arguments

`ptrStateX`	echo canceller memory in X space
`ptrStateY`	echo canceller memory in Y space
`refIn`	"reference in" pointer to one frame of output samples from near terminal (`refIn` in Figure 4-4). This buffer should be in Y memory.
`lecIn`	"signal in" pointer to one frame of input samples from remote terminal (`lecIn` in Figure 4-4). This buffer should be in Y memory.
`lecOut`	"signal out" pointer to one frame of samples from remote terminal after processing by LEC (`lecOut` in Figure 4-4). This buffer should be in Y memory.
`frame`	frame length in number of samples. This value should be equal to `LEC_FRAME`.
`lec_enable`	flag to enable echo cancellation; if flag is cleared, input is passed unchanged.

Return Value

None.

Remarks

Setting `lec_enable` to `LEC_FALSE` returns a buffer of un-processed data, but the LEC continues to adapt in the background.

Code Example

```
long lecStateMemX[LEC_X_MEM_SIZE_LONG]          _XBSS(4);
long lecStateMemY[LEC_Y_MEM_SIZE_LONG]          _YBSS(4);

short lecScratchX[LEC_X_SCRATCH_MEM_SIZE_SHORT] _XBSS(2);
short lecScratchY[LEC_Y_SCRATCH_MEM_SIZE_SHORT] _YBSS(512);

LEC_init(lecStateMemX, lecStateMemY, lecScratchX, lecScratchY,
   LEC_TRUE, LEC_ECHO_DELAY);

LEC_apply(lecStateMemX, lecStateMemY, refIn, lecIn, lecOut,
   LEC_FRAME, LEC_ENABLE);
```

LEC_applyNLP

Description

Applies NLP portion of the echo cancellation to the current frame of data.

Include

`lec_api.h`

Prototype

```
void LEC_applyNLP(long* ptrStateX, short* lecOut, short frame,
   short lec_enable);
```

Arguments

`ptrStateX`	echo canceller memory
`lecOut`	one frame of output samples from `LEC_apply` (`lecOut` in Figure 4-4). This buffer should be in Y memory.
`frame`	echo canceller frame size. This value should be equal to `LEC_FRAME`.
`lec_enable`	flag to enable NLP. If flag is cleared, input is passed unchanged.

Return Value

None.

Remarks

The `lec_enable` flag should be `LEC_FALSE` if the LEC has not been applied to the frame. The `LEC_apply` NLP function is applied to the output of the `LEC_apply` function.

Code Example

```
long lecStateMemX[LEC_X_MEM_SIZE_LONG]          _XBSS(4);
long lecStateMemY[LEC_Y_MEM_SIZE_LONG]          _YBSS(4);

short lecScratchX[LEC_X_SCRATCH_MEM_SIZE_SHORT] _XBSS(2);
short lecScratchY[LEC_Y_SCRATCH_MEM_SIZE_SHORT] _YBSS(512);

LEC_init(lecStateMemX, lecStateMemY, lecScratchX, lecScratchY,
   LEC_TRUE, LEC_ECHO_DELAY);

LEC_apply(lecStateMemX, lecStateMemY, refIn, lecIn, lecOut,
   LEC_FRAME, LEC_ENABLE);

LEC_applyNLP(lecStateMemX, lecOut, LEC_FRAME, LEC_NLP_SHIFT,
   LEC_ENABLE);
```

LEC_setNLPLevel

Description

Sets the required level of NLP.

Include

`lec_api.h`

Prototype

```
void LEC_setNLPLevel(long* ptrStateX, short level);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`level`	an integer value between 0 and 15 representing the desired NLP level

Return Value

None.

Remarks

The desired NLP level is the number of right shifts performed by the NLP.

Code Example

```
LEC_setNLPLevel(lecStateMemX, 5);
```

Sets the desired NLP level to 5 for the instance of the algorithm `lecStateMemX`.

LEC_getNLPLevel

Description

Returns the current NLP level.

Include

`lec_api.h`

Prototype

```
short LEC_getNLPLevel(long* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the X memory for this instance of LEC

Return Value

The current NLP level as an integer

Remarks

None.

Code Example

```
int nlpLevel;
nlpLevel = LEC_getNLPLevel(lecStateMemX);
```

`nlpLevel` contains the desired NLP level for the instance of the algorithm `lecStateMemX`.

LEC_setNLPHangover

Description

Sets the length of the hangover count for the portion of the LEC algorithm where double talk detection is used for deciding when to apply NLP.

Include

```
lec_api.h
```

Prototype

```
void LEC_setNLPHangover(long* ptrStateX, int frames);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`frames`	an integer value from 1 to 100 representing the desired double talk hangover

Return Value

None.

Remarks

The NLP hangover is used to adjust the hysteresis around the double talk detection. A larger value keeps the double talk detection active longer. Setting NLP hangover to '`1`' disables the hysteresis.

Code Example

```
LEC_setNLPHangover(lecStateMemX, 10);
```

Sets the NLP hangover to 10 frames (representing 100 ms of data) for the instance of the algorithm `lecStateMemX`.

LEC_getNLPHangover

Description

Returns the current NLP hangover, as it is related to NLP detection.

Include

`lec_api.h`

Prototype

```
short LEC_getNLPHangover(long* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the X memory for this instance of LEC

Return Value

NLP hangover value

Remarks

None.

Code Example

```
short nlpHangover;
nlpHangover = LEC_getNLPHangover(lecStateMemX);
```

`nlpHangover` contains the NLP hangover value set for the instance of the algorithm `lecStateMemX`.

LEC_setNLPThreshold

Description

Sets the suppression threshold for NLP.

Include

`lec_api.h`

Prototype

```
void LEC_setNLPThreshold(long* ptrStateX, long threshold);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`threshold`	a Q31 number representing block energy

Return Value

None.

Remarks

Use Equation 4-1 to convert the target dBm0 value (y) to its equivalent NLP Threshold Q31 number (x).

EQUATION 4-1: NLP THRESHOLD CALCULATION

$$10^{\left(\frac{y+6}{10}\right)} \times 2^{31} \approx x$$

For example, Equation 4-2 shows the results of using Equation 4-1 to convert -40 dBm0 to its equivalent NLP Threshold Q31 value.

EQUATION 4-2: NLP THRESHOLD CALCULATION EXAMPLE

$$10^{\left(\frac{-40+6}{10}\right)} \times 2^{31} \approx 0xD0B90$$

Code Example

```
LEC_setNLPThreshold(lecStateMemX, 0x00010000);
```

Sets the suppression threshold to 0x00010000 for the instance of the algorithm `lecStateMemX`.

LEC_getNLPThreshold

Description

Returns the current NLP suppression threshold.

Include

`lec_api.h`

Prototype

```
long LEC_getNLPHangover(long* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the X memory for this instance of LEC

Return Value

NLP suppression threshold level

Remarks

None.

Code Example

```
long nlpThres;
nlpThres = LEC_getNLPThreshold(lecStateMemX);
```

`nlpThres` contains the noise suppression threshold setting for the instance of the algorithm `lecStateMemX`.

LEC_setAdaptationHangover

Description

Sets the length in frames of adaptation update hangover.

Include

`lec_api.h`

Prototype

```
void LEC_setAdaptationHangover(long* ptrStateX, short frames);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`frames`	an integer value from 1 to 100 representing the adaptation hangover

Return Value

None.

Remarks

The adaptation hangover is used to adjust the hysteresis around the adaptation update. A larger value keeps the adaptation update active longer.

Code Example

```
LEC_setAdaptationHangover(lecStateMemX, 3);
```

Sets the desired adaptation hangover to three frames (representing 30 ms of data) for the instance of the algorithm `lecStateMemX`.

LEC_getAdaptationHangover

Description

Returns the current adaptation hangover.

Include

`lec_api.h`

Prototype

```
short LEC_getAdaptationHangover(long* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the X memory for this instance of LEC

Return Value

The adaptation hangover value

Remarks

None.

Code Example

```
short adaptationHangover;
adaptationHangover = LEC_getAdaptationHangover(lecStateMemX);
```

`adaptationHangover` contains the adaptation hangover value set for the instance of the algorithm `lecStateMemX`.

LEC_setToneDetect

Description

Enables or disables the Tone Detection module.

Include

`lec_api.h`

Prototype

```
void LEC_setToneDetect(long* ptrStateX, short toneDetect);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`toneDetect`	a flag to indicate if tone detection is required, either `LEC_TRUE` or `LEC_FALSE`

Return Value

None.

Remarks

None.

Code Example

```
LEC_setToneDetect(lecStateMemX, EC_TRUE);
```

Enables tone detection for the instance of the algorithm `lecStateMemX`.

LEC_getToneDetect

Description

Returns the current state of the Tone Detection module.

Include

`lec_api.h`

Prototype

```
short LEC_getToneDetect(long* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the X memory for this instance of EC

Return Value

The tone detection flag

Remarks

None.

Code Example

```
short toneDetect;
toneDetect = LEC_getToneDetect(lecStateMemX);
```

`toneDetect` contains the tone detection state set for the instance of the algorithm `lecStateMemX`, either `LEC_TRUE` or `LEC_FALSE`.

LEC_setDoubleTalkDetect

Description

Enables or disables the double talk detection of the LEC algorithm.

Include

`lec_api.h`

Prototype

```
void LEC_setDoubleTalkDetect(long* ptrStateX, short state);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`state`	either `LEC_TRUE` or `LEC_FALSE`

Return Value

None.

Remarks

Disabling double talk detection may improve LEC performance in noisy conditions by forcing adaptation to take place at every frame. Use `LEC_FALSE` to disable the double talk detection, which makes the algorithm to adapt every frame.

Code Example

```
void LEC_setDoubleTalkDetect(lecStateMemX, LEC_FALSE);
```

Sets forced adaptation for the instance of the algorithm `lecStateMemX`.

LEC_getDoubleTalkDetect

Description

Returns the state of double talk detection.

Include

`lec_api.h`

Prototype

```
short LEC_getDoubleTalkDetect(long* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the X memory for this instance of LEC

Return Value

The double talk flag

Remarks

None.

Code Example

```
int DT_detect;
DT_detect = LEC_getForcedadaptation(lecStateMemX);
```

`DT_detect` contains the double talk detection value for the instance of the algorithm `lecStateMemX`, either `LEC_TRUE` or `LEC_FALSE`.

LEC_setInhibitAdaptation

Description

Sets the state of the inhibit adaptation.

Include

`lec_api.h`

Prototype

```
void LEC_setInhibitAdaptation(long* ptrStateX, short state);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of LEC
`state`	either `LEC_TRUE` or `LEC_FALSE`

Return Value

None.

Remarks

None.

Code Example

```
LEC_setInhibitAdaptation(lecStateMemX, EC_TRUE);
```

Prevents adaptation of the instance of the algorithm `lecStateMemX`.

LEC_getInhibitAdaptation

Description

Returns the state of the inhibit adaptation.

Include

`lec_api.h`

Prototype

```
short LEC_getInhibitadaptation(long* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the X memory for this instance of LEC

Return Value

The inhibit adaptation flag

Remarks

None.

Code Example

```
short inhibitAdaptation;
inhibitAdaptation = LEC_getInhibitAdaptation(lecStateMemX);
```

`inhibitadaptation` contains the inhibit adaptation value set for the instance of the algorithm `lecStateMemX`, either `LEC_TRUE` or `LEC_FALSE`.

LEC_TRUE

Description

Used to indicate 'true' to the LEC algorithm.

Value

1

LEC_FALSE

Description

Used to indicate 'false' to the LEC algorithm.

Value

0

LEC_ECHO_DELAY

Description

Echo path length in ms, set in `lec_api.h`.

Value

16, 32, 64 or 128

LEC_FRAME

Description

Size of the input and output buffers.

Value

80 (default), 40, 16 or 8

LEC_ENABLE_DEFAULT

Description

Flag to enable line echo cancellation.

Value

1

LEC_ENABLE_NLP_DEFAULT

Description

Flag to enable NLP operation.

Value

1

LEC_ENABLE_DT_DETECT_DEFAULT

Description

Flag to enable double talk detection (to inhibit adaptation on DT frames).

Value

1

LEC_ENABLE_TONE_DETECT_DEFAULT

Description

Flag to enable detection of test tones as specified by ITU-T G.168.

Value

1

LEC_INHIBIT_ADAPT_DEFAULT

Description

Flag to inhibit adaptation (useful in some kinds of testing).

Value

0

LEC_NLP_SUP_THRESHOLD_DEFAULT

Description

Energy threshold for applying NLP.

Value

0x00200000

LEC_NLP_HANGOVER_FRAMES_DEFAULT

Description

Number of frames after DT detected during which NLP is inhibited.

Value

25

LEC_NLP_SHIFT_DEFAULT

Description

Right shift applied by NLP (multiply by 6 to get approximate dB).

Value

5

LEC_DT_HANGOVER_FRAMES_DEFAULT

Description

Number of hangover frames to inhibit adaptation during double talk.

Value

1

LEC_X_MEM_SIZE_LONG

Description

Size in long integers of the memory location required for the X State memory.

Value

```
((LEC_ECHO_DELAY_MAX*4) + (3*LEC_FRAME_MAX/2) + 104)
```

LEC_Y_MEM_SIZE_LONG

Description

Size in long integers of the memory location required for the Y State memory.

Value

```
(LEC_ECHO_DELAY_MAX*4)
```

LEC_X_SCRATCH_MEM_SIZE_SHORT

Description

Size in short integers of the memory location required for the X scratch memory.

Value

```
(LEC_ORDER_MAX + LEC_FRAME_MAX)
```

LEC_Y_SCRATCH_MEM_SIZE_SHORT

Description

Size in short integers of the memory location required for the Y scratch memory.

Value

```
(LEC_ORDER_MAX)
```

4.7 APPLICATION TIPS

The LEC algorithm is designed to work well under a range of conditions. It may be optimized to meet your own requirements by adjusting the parameters that are described in the API.

Following are some tips for affecting the performance of the algorithm:

1. The optimum input signal levels for testing audio and communication systems are generally considered to lie 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. Choose an NLP level that best fits the application. A very high NLP level not only suppresses the residual echo, but may also clip the speech level to some extent. Do not apply NLP if the frame has not been processed by the LEC algorithm.
3. The Double Talk Detection Hangover can be used to adjust the window during which the LEC algorithm will wait before applying NLP to the send signal. If the application is experiencing some leading edge speech clipping, try increasing the DTD Hangover.
4. The Adaptation Hangover can be used to adjust the window during which an adaptation update stays active. This parameter can be adjusted in situations where the speech level is low and double talk may not be detected.
5. In cases where the near-end ambient acoustic environment is noisy (such as cell phones and automotive hands-free units), the noise level may cause the double talk detection to be activated, thereby inhibiting adaptation. In such cases, the LEC algorithm can be forced to adapt all the time, which may improve the overall system performance.
6. The user application should never inhibit adaptation. This feature is only provided for test and compliance checking purposes.
7. Tone detection should be enabled in a networking application where a standard 6.168 disabling tone may be used (2100 Hz with phase reversals every 450 ± 25 ms). Otherwise, it may be disabled.
8. The NLP Threshold value can be adjusted to avoid unwanted attenuation of speech. If the threshold is raised, more input date is likely to be treated as echo and suppressed. If the threshold is lowered, more unwanted echo is likely to get through.

Index

A

API 5
API Functions 38
- LEC_apply 42
- LEC_applyNLP 43
- LEC_DT_HANGOVER_FRAMES_DEFAULT 61
- LEC_ECHO_DELAY 58
- LEC_ENABLE_DEFAULT 59
- LEC_ENABLE_DT_DETECT_DEFAULT 59
- LEC_ENABLE_NLP_DEFAULT 59
- LEC_ENABLE_TONE_DETECT_DEFAULT 59
- LEC_FALSE 58
- LEC_FRAME 58
- LEC_getAdaptationHangover 51
- LEC_getDoubleTalkDetect 55
- LEC_getInhibitAdaptation 57
- LEC_getNLPHangover 47
- LEC_getNLPLevel 45
- LEC_getNLPThreshold 49
- LEC_getToneDetect 53
- LEC_INHIBIT_ADAPT_DEFAULT 60
- LEC_init 39
- LEC_NLP_HANGOVER_FRAMES_DEFAULT 60
- LEC_NLP_SHIFT_DEFAULT 60
- LEC_NLP_SUP_THRESHOLD_DEFAULT 60
- LEC_relocateXScratchMem 40
- LEC_relocateYScratchMem 41
- LEC_setAdaptationHangover 50
- LEC_setDoubleTalkDetect 54
- LEC_setInhibitAdaptation 56
- LEC_setNLPHangover 46
- LEC_setNLPLevel 44
- LEC_setNLPThreshold 48
- LEC_setToneDetect 52
- LEC_TRUE 58
- LEC_X_MEM_SIZE_LONG 61
- LEC_X_SCRATCH_MEM_SIZE_SHORT 61
- LEC_Y_MEM_SIZE_LONG 61
- LEC_Y_SCRATCH_MEM_SIZE_SHORT 61

C

Customer Notification Service 8
Customer Support 8

D

Demo Code Description 23, 27
Demonstration Procedure 22, 26
Demonstration Setup 20, 25
Demonstration Summary 19, 24
Documentation
- Conventions 6

Double Talk Detector (DTD) 34
DTD 34

F

Finite Impulse Response (FIR) 12

H

Host System Requirements 13

I

Installation
- Installing the Library 15

Internet Address 8

L

Library Files
- demo Folder 16
- doc Folder 17
- h Folder 17
- lib Folder 17

Line Echo Cancellation
- Typical Applications 12

Line Echo Cancellation Algorithm 12, 33

M

Microchip Internet Web Site 8
MPLAB IDE User's Guide 7

N

NLMS 34
NLP 34
Nonlinear Processor 34
Normalized Least Mean Square (NLMS) 12
Normalized Least Mean Square (NLMS)
- Adaptive Filter 34

O

Overview
- Line Echo Cancellation 11

R

Reading, Recommended 7
Resource Requirements
- Computational Speed 37
- Data Memory Usage 37
- Estimated Dynamic Memory Usage 37

T

Tone Detector 34

W

Warranty Registration 6
WWW Address 8

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