

dsPIC® DSC Acoustic Echo Cancellation Library User's Guide

© 2004-2011 Microchip Technology Inc.

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

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

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

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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC^{32} logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2004-2011, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

ISBN: 978-1-61341-247-3

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.

© 2004-2011 Microchip Technology Inc.

dsPIC® DSC ACOUSTIC ECHO CANCELLATION LIBRARY USER'S GUIDE

Table of Contents

Preface 5

Chapter 1. Introduction

1.1 Acoustic Echo Cancellation Overview 13
1.2 Features 14
1.3 Host System Requirements 15

Chapter 2. Installation

2.1 Installation Procedure 17
2.2 AEC Library Files 17

Chapter 3. AEC Demonstration

3.1 AEC Demonstration for dsPIC33F Device Family 21
3.2 AEC Demonstration for dsPIC33E Device Family 27

Chapter 4. Application Programming Interface (API)

4.1 Adding the AEC Library to an Application 31
4.2 Memory Model Compile Options 32
4.3 AEC Algorithm Overview 34
4.4 Library Usage 36
4.5 Register Usage 37
4.6 Resource Requirements 38
4.7 AEC Library API Functions 39
4.8 Application Tips 61

Index 63

Worldwide Sales and Service 64

© 2004-2011 Microchip Technology Inc.

NOTES:

© 2004-2011 Microchip Technology Inc.

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 Acoustic 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 document describes how to use the dsPIC DSC Acoustic Echo Cancellation Library. The document is organized as follows:

- **Chapter 1. Introduction** – This chapter introduces the dsPIC DSC Acoustic Echo Cancellation Library and provides a brief overview of acoustic 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. AEC Demonstration** – This chapter provides a hands-on demonstration of acoustic echo cancellation in a working application.
- **Chapter 4. Application Programming Interface (API)** – This chapter outlines how the API functions provided in the dsPIC DSC Acoustic Echo Cancellation Library can be included in your application software through the API.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*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) { ... }`

© 2004-2011 Microchip Technology Inc.

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 Acoustic Echo Cancellation 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 specific 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 more 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 more 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 `Readmes` subdirectory of the MPLAB® IDE installation directory. The Readme files contain update 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

© 2004-2011 Microchip Technology Inc.

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

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

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

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 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 Microchip web site at:
http://www.microchip.com/support

DOCUMENT REVISION HISTORY

Revision A (July 2004)

This is the initial release of this document.

Revision B (April 2005)

The document was updated to reflect added sample rate conversion functions and corresponding demo changes.

Revision C (August 2008)

This revision includes the following updates for version 5.0 of the Acoustic Echo Cancellation library.

This document has been renamed from *"dsPIC30F Acoustic Echo Cancellation Library User's Guide"* to its new name of *"dsPIC® DSC Acoustic Echo Cancellation Library User's Guide"*. There has been no change to the Microchip literature number (DS70134) 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 follows:

- **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 Acoustic Echo Cancellation library files have been updated and the `inc` folder was renamed to `h`. See **Section 2.2 "AEC Library Files"**.
- **Chapter 3. AEC Demonstration** (formerly Chapter 6. Acoustic 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 4.7 "AEC Library API Functions"**.
  - The information in the chapter formerly known as Acoustic 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 4.6 "Resource Requirements"**.

© 2004-2011 Microchip Technology Inc.

Revision D (November 2008)

- Updated demonstration file names in Table
- 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 code examples in **Chapter 4. “Application Programming Interface (API)”** to reflect changes to the API functionality
- Added **Section 4.2 “Memory Model Compile Options”**

Revision E (February 2011)

This revision incorporates the following updates:

- Note:
  - Added a note in **Section Table 4-4: “Computational Speed”**
- Sections:
  - Added **Section 4.5 “Register Usage”**
- Additional minor corrections such as language and formatting updates were incorporated throughout the document

Revision F (June 2011)

This revision incorporates the following updates:

- Examples:
  - Added title to Example 4-1 and Example 4-2 in **4.4 "Library Usage"**
- Figures:
  - Removed **FIGURE 2-1** through **FIGURE 2-5** in **2.1 "Installation Procedure"**
  - Removed **FIGURE 4-1** through **FIGURE 4-4** in **4.1 "Adding the AEC Library to an Application"**
- Notes:
  - Removed the note in **2.1 "Installation Procedure"**
  - Added a note in **4.4 "Library Usage"**
- Sections:
  - Updated the Recommended Reading section in **"Preface"**
  - Updated the first paragraph in **Chapter 1."Introduction"**
  - Updated the list of attributes in **1.3 "Host System Requirements"**
  - Updated **2.1 "Installation Procedure"**
  - Updated the version number for `EC v5.0` to `EC v6.0` in **2.2 "AEC Library Files"**
  - Updated **2.2.4 "lib Folder"**
  - Updated the existing section in **Chapter 3."AEC Demonstration"** to **3.1 "AEC Demonstration for dsPIC33F Device Family"**
  - Added **3.2 "AEC Demonstration for dsPIC33E Device Family"**
  - Updated step 1 and step 2 in **3.1.2.3 "Program the dsPIC DSC Device"**
  - Updated step 2 and step 4 in **4.1 "Adding the AEC Library to an Application"**
  - Updated **4.6 "Resource Requirements"**
- Tables:
  - Updated Table 2-1 and Table 2-3 in **2.2 "AEC Library Files"**
  - Added title to Table 4-1 through Table 4-4 in **4.6 "Resource Requirements"**
  - Updated Table 4-1, Table 4-2 and Table 4-4 in **4.6 "Resource Requirements"**
- Additional minor corrections such as language and formatting updates were incorporated throughout the document

© 2004-2011 Microchip Technology Inc.

Chapter 1. Introduction

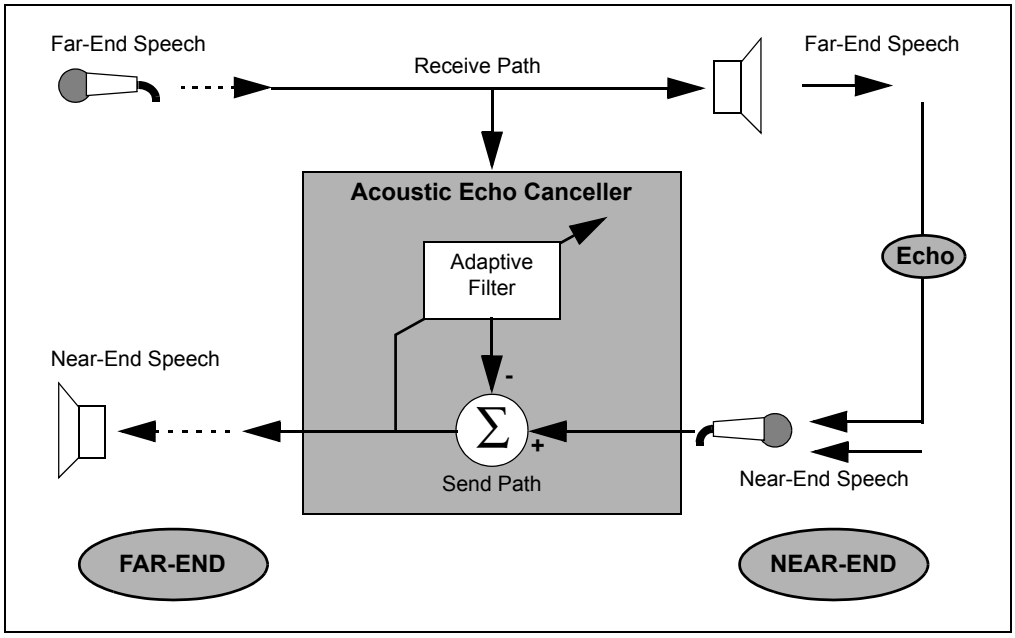
This chapter introduces the dsPIC DSC Acoustic Echo Cancellation (AEC) Library (referred to as the AEC library). This library provides functionality to suppress echo in applications that are susceptible to echo. The library supports the dsPIC33F and dsPIC33E device families. This manual provides information that you can use to incorporate the AEC capability into your embedded solution. The following topics are covered in this chapter:

- Acoustic Echo Cancellation Overview
- Features
- Host System Requirements

1.1 ACOUSTIC ECHO CANCELLATION OVERVIEW

Acoustic echo cancellation suppresses or cancels echoes generated in applications due to a feedback path or coupling between input and output terminals of a system. Figure 1-1 shows an example of a speech and telephony system, which is susceptible to Acoustic Echo. In this case, the echo is caused due to the acoustic coupling between the speaker and microphone at the near-end of the communication link, which results in a perceptible and distracting echo at the far-end. In this case, an AEC algorithm running at the near-end, will suppress the echo and improve the performance of the system.

FIGURE 1-1: AEC IN A SPEECH AND TELEPHONY APPLICATION



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

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

For hands-free phones intended to be used in automotive environment, such as inside an automobile, this library is compatible with the G.167 standard for AEC and with other relevant ITU-T standards.

The AEC 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 AEC library provides an `EC_init` function for initializing the various data structures required by the algorithm and an `EC_apply` function to remove the echo component from a 10 ms block of sampled 16-bit speech data. You can easily call both the functions through a well-documented Application Programmer's Interface (API).

The AEC algorithm is primarily a time-domain algorithm. The received far-end speech samples (typically received across a communication channel, such as a telephone line) are filtered using an adaptive Finite Impulse Response (FIR) filter and then subtracted from the near-end input speech signal. The coefficients of this filter are adapted using the Normalized Least Mean Square (NLMS) algorithm, such that the filter closely models the acoustic path between the near-end speaker and the near-end microphone (that is, the path traversed by the echo). A Nonlinear Processor (NLP) algorithm is available to eliminate the residual echo.

1.2 FEATURES

Key features of the AEC 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
- Acoustic Echo Cancellation for 16 ms, 32 ms, 64 ms or 128 ms echo delays or "tail lengths" (configurable)
- Compatible with G.167 specifications for in-car applications
- Audio Bandwidth: 0 kHz to 4 kHz at an 8 kHz sampling rate
- Convergence Rate: Up to 47 dB/sec., typically greater than 30 dB/sec
- Acoustic Echo Cancellation: Up to 50 dB, typically > 40 dB
- Can be used together with the Noise Suppression (NS) library
- Demonstration application source code is provided with the library
- NLP attenuation level can be adjusted to suit application requirements
- AEC adaptation can be force-enabled or disabled by the user application
- Run-time control of key algorithm parameters is provided

© 2004-2011 Microchip Technology Inc.

1.3 HOST SYSTEM REQUIREMENTS

The AEC 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:

© 2004-2011 Microchip Technology Inc.

Chapter 2. Installation

This chapter describes the various files in the AEC 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
- AEC Library Files

2.1 INSTALLATION PROCEDURE

To install the library, follow these steps:

1. Double-click `AEC 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 AEC library installation.

The installation process creates the folder named `EC v6.0`, which contains the files described in **2.2 "AEC Library Files"**.

2.2 AEC LIBRARY FILES

The dsPIC DSC Acoustic Echo Cancellation Library CD creates a directory named `EC v6.0`. This directory contains these folders:

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

© 2004-2011 Microchip Technology Inc.

2.2.1 demo Folder

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

TABLE 2-1: DEMONSTRATION FILES

File Name	Description
dsPIC33F EC demo2.hex	Demonstration hexadecimal file for board 2 on dsPIC33F.
dsPIC33F EC demo1.hex	Demonstration hexadecimal file for board 1 on dsPIC33F.
dsPIC33F EC demo.mcp	Demonstration MPLAB Project file for dsPIC33F.
dsPIC33E EC demo2.hex	Demonstration hexadecimal file for board 2 on dsPIC33E.
dsPIC33E EC demo1.hex	Demonstration hexadecimal file for board 1 on dsPIC33E.
dsPIC33E EC 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\ec_api.h	C header file defining the interface to the AEC 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\eclibv6_33F.a	AEC library archive file for dsPIC33F.
libs\eclibv6_33E.a	AEC library archive file for dsPIC33E.
src\dsPICDEM1_1Plus.c	C source files containing routines for the dsPICDEM1.1 Plus development board.
src\MEB.c	C source files containing routines for the MEB.
src\lcd_strings.c	C source files for LCD display driver.
src\main.c	C source files containing the main speech processing routine.
src\UART1Drv.c	C source file containing code for the UART1 peripheral.
src\UART2Drv.c	C source file containing code for the UART2 peripheral.
src\SI3000Drv.c	C source file containing the code for the Si3000 Codec.
src\WM8731CodecDrv.c	C source file containing 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.

© 2004-2011 Microchip Technology Inc.

2.2.2 `doc` Folder

This folder contains the user's guide for the AEC 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 AEC library, as listed in Table 2-2.

TABLE 2-2: INCLUDE FILE

File Name	Description
`ec_api.h`	Include file that contains the interface to the AEC 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 AEC 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
`eclibv6_33F.a`	This is the AEC library archive file for dsPIC33F. This file must be included in the application in order to use the library.
`eclibv6_33E.a`	This is the AEC library archive file for dsPIC33E. This file must be included in the application in order to use the library.

NOTES:

© 2004-2011 Microchip Technology Inc.

Chapter 3. AEC Demonstration

This chapter provides a hands-on demonstration of acoustic echo cancellation in a working application running on the dsPIC33F or dsPIC33E device.

3.1 AEC DEMONSTRATION FOR dsPIC33F DEVICE FAMILY

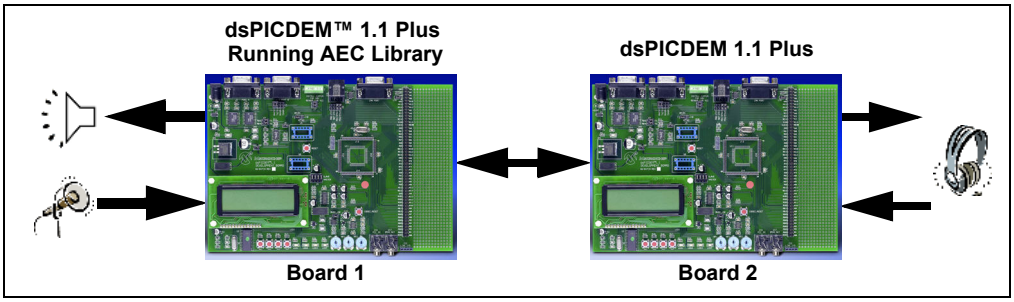
The following topics are covered in this section:

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

3.1.1 Demonstration Summary

To demonstrate the functionality of the AEC library, a sample dsPIC33F-based application emulating two speaker phones engaged in voice 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: ACOUSTIC ECHO CANCELLATION DEMONSTRATION



A speaker and a microphone are connected to dsPICDEM 1.1 Plus development board 1 and are located in proximity to each other. A headset is connected to board 2.

When a person speaks into the headset connected to board 2, 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 1.

The dsPIC DSC device on board 1 receives the signal through the on-board RS-232 transceiver and the device's UART1 module. The dsPIC DSC device then plays out the signal on the speaker through its DCI module and on-board Si3000 codec.

© 2004-2011 Microchip Technology Inc.

Due to the proximity of the speaker to the microphone, the sound from the speaker enters the microphone and is sampled by the dsPIC DSC device through the codec. The device then transmits the microphone signal to board 2 through the RS-232 interface. If a person is speaking into the microphone connected to board 1, the signal transmitted to board 2 is a combination of the near-end speech and the undesirable acoustic echo of the far-end speech. This combination of speech and echo can be heard on the headset connected to board 2. In this example, board 1 represents the near-end and board 2 represents the far-end.

When started, the program initializes with acoustic echo cancellation turned OFF, indicated by LED1 being turned OFF and OFF being written to the LCD screen. With acoustic echo cancellation OFF, the signal heard in the headset connected to board 2 contains noticeable echo.

The acoustic 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 2 becomes echo-free.

The demonstration application program invokes the `EC_apply` and `EC_applyNLP` functions from the AEC library to suppress the unwanted far-end echo mixed with the near-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.

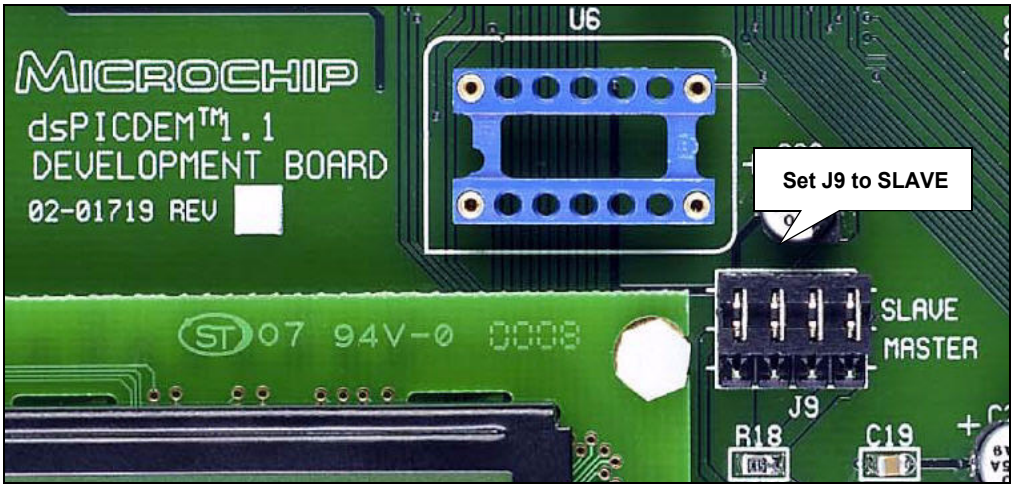
3.1.2.1 CONFIGURE dsPICDEM 1.1 PLUS DEVELOPMENT BOARDS

Before applying power, you need to configure the board:

1. Set the 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 fold-up speaker to the SPKR OUT jack (J17) on board 1.
3. Connect the microphone to the MIC IN jack (J16) on board 1. Make sure the microphone is turned on and is situated close enough to the speaker to generate feedback into the microphone.
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.

© 2004-2011 Microchip Technology Inc.

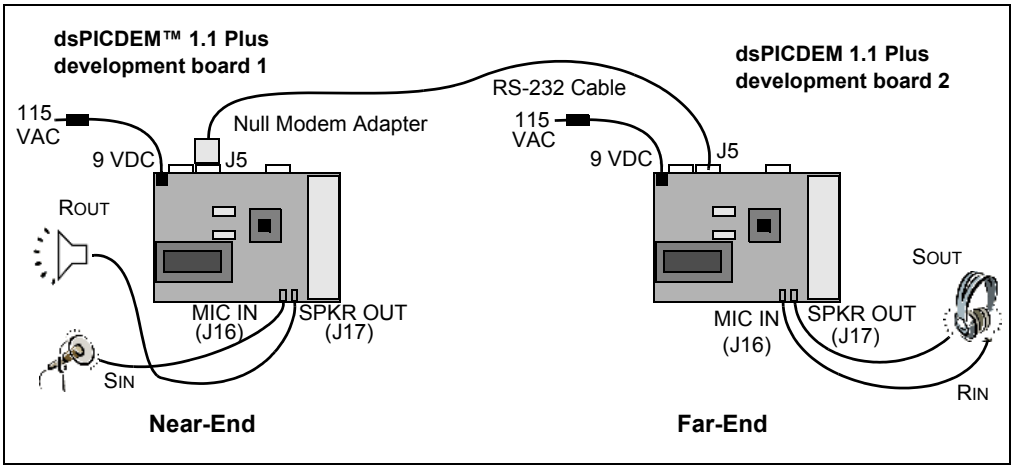
FIGURE 3-2: DEMONSTRATION BOARD SETUP



3.1.2.2 SET UP THE DEMONSTRATION

After both the boards are configured, attach the speakers and microphones 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 AEC 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 EC 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 hexadecimal file by selecting either:
   - *File>Import>dsPIC33F EC demo1.hex* for board 1 or
   - *File>Import>dsPIC33F EC demo2.hex* for board 2
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 menu>Program*. The Output window displays the download process and indicates that the programming has succeeded.
5. 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 acoustic echo cancellation information in the LCD display.
6. Repeat steps 2 through 5 for the second board.
7. 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™ In-Circuit Emulator can be used instead 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:

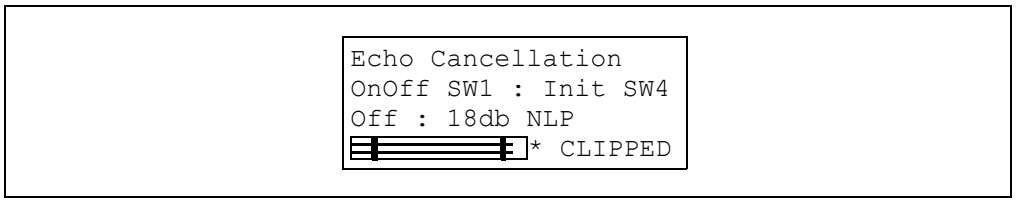
1. Press the Reset button on board 1. Wait till LED1 through LED4 on board 1 turn ON. Now Reset board 2. Wait until LED1 through LED4 on board 1 and board 2 turn OFF. This indicates that the boards are synchronized and the demo is running.
2. Put on the headset connected to dsPICDEM 1.1 Plus development board 2.
3. Begin speaking into the microphone input of the headset. You should observe the top bar of the volume unit (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 the dsPICDEM 1.1 Plus development board 1. 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 dsPICDEM 1.1 Plus development board 1. Observe that the LED1 on board 1 turns on, indicating that the AEC 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 acoustic echo cancellation during double talk, have a person simultaneously speak into the microphone connected to dsPICDEM 1.1 Plus development board 1. 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 speaking normally.

© 2004-2011 Microchip Technology Inc.

To experiment with different values of echo tail length, change the `EC_ECHOTAIL` constant defined in the `ec_api.h` include file, rebuild `EC demo.mcp`, reprogram board 1 and rerun the demo application. The demo application relays the state of operation through 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 acoustic echo cancellation ON and OFF. SW4 is used to reinitialize the acoustic echo cancellation 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 `nearEndIn` buffer, the bottom bar represents the `farEndIn` 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 acoustic echo cancellation 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 AEC library functions.

The main function calls the `EC_init()` function from the AEC library, which initializes the AEC 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, this 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.

© 2004-2011 Microchip Technology Inc.

The 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 (ISR), the corresponding interrupt flag is cleared, data is either written to U1TXREG, or read from U1RXREG and saved in a circular buffer.

The codec driver is read for a full frame of data. The contents of the coded data buffers are copied into the `nearEndIn` array and the `EC_apply()` function from the AEC library is called with `nearEndIn` as the input data frame. The `nearEndIn` data buffer, which is also the output of the `EC_apply` function after it has been executed, is transferred to the UART for transmission to board 2. The UART data is converted from µ-Law to 16-bit linear and stored in `farEndIn`, which is the reference (echo) input data frame to `EC_apply()`.

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 the acoustic echo cancellation ON or OFF, external interrupts for SW1 are enabled. In the main loop, the value of `applyAEC` is read and passed to `EC_apply()` as the enable flag. If `applyAEC` is '`0`', the acoustic echo cancellation is still called, but the input/output buffer is not changed. This enables the acoustic echo cancellation to maintain adaptation to changes in echo path, while it is not enabled.

© 2004-2011 Microchip Technology Inc.

3.2 AEC DEMONSTRATION FOR dsPIC33E DEVICE FAMILY

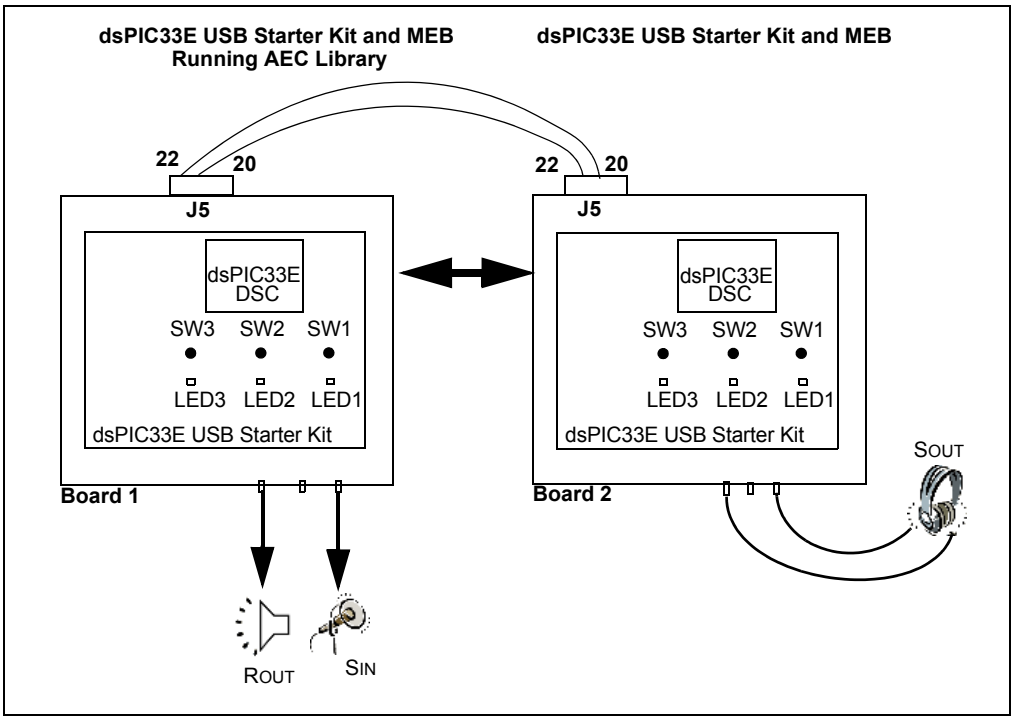
The following topics are covered in this section:

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

3.2.1 Demonstration Summary

To demonstrate the functionality of the AEC 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: ACOUSTIC ECHO CANCELLATION DEMONSTRATION



A speaker and a microphone are connected to board 1 and are located in proximity to each other. A headset is connected to board 2.

When a person speaks into the headset connected to board 2, the speech signal is sampled through the on-board WM8731 voice band codec and the 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 1.

The dsPIC DSC device on board 1 receives the signal through the on-board I/O expansion connector and the device's UART2 module. The dsPIC DSC device then plays out the signal on the speaker through its DCI module and on-board WM8731 codec.

Due to the proximity of the speaker to the microphone, the sound from the speaker enters the microphone and is sampled by the dsPIC DSC device through the codec. The device then transmits the microphone signal to board 2 through the UART2 module. If a person is speaking into the microphone connected to board 1, the signal transmitted to board 2 is a combination of the near-end speech and the undesirable acoustic echo of the far-end

speech. This combination of speech and echo can be heard on the headset connected to board 2. In this example, board 1 represents the near-end and board 2 represents the far-end.

When started, the program initializes with acoustic echo cancellation turned OFF, indicated by LED3 being turned OFF on the dsPIC33E USB Starter Kit. With the acoustic echo cancellation OFF, the signal heard in the headset connected to board 2 contains noticeable echo.

The acoustic echo cancellation 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 2 becomes echo-free.

The demonstration application program invokes the `EC_apply` and `EC_applyNLP` functions from the AEC library to suppress the unwanted far-end echo mixed with the near-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 fold-up speaker to the headphone jack (J18) on MEB 1.
3. Connect the microphone to the microphone jack (J7) on MEB 1. Make sure the microphone is turned on and is situated close enough to the speaker to generate feedback into the microphone.
4. Connect the headset microphone to the microphone jack on MEB 2.
5. Connect the headset speaker to the headphone jack on MEB 2.
6. Connect each of the two dsPIC33E starter kits to a PC using the USB A-to-mini B cable provided with the starter kit.
7. 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 SET UP THE DEMONSTRATION

After both the boards are configured, attach the speakers and microphones, and interconnect the boards.

© 2004-2011 Microchip Technology Inc.

3.2.2.3 PROGRAM THE dsPIC DSC DEVICE

Use this process to load the AEC demonstration into the dsPIC DSC device on each board.

1. On your PC, launch MPLAB IDE and open the `dsPIC33E EC 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 hexadecimal file by selecting either:
   - *File>Import>dsPIC33E EC demo1.hex* for board 1 or
   - *File>Import>dsPIC33E EC demo2.hex* for board 2
3. Select Starter Kit on Board as the Programmer, and then select *Programmer>Connect* to link to the dsPIC DSC target device on board 1. The Output window confirms that the target device is ready.
4. Select *Programmer>Program*. The Output window displays the download process and indicates that the programming has succeeded.
5. Repeat steps 2 through 4 for the second board.
6. Make sure that the two boards are not located too close to each other to avoid any acoustic coupling between the two boards.

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 2.
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 1. 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 LED3 on board 1 turns on, indicating that the AEC 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 acoustic echo cancellation during double talk, let a person simultaneously speak into the microphone connected to board 1. 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 `EC_ECHOTAIL` constant defined in the `ec_api.h` include file, rebuild `EC 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 acoustic echo cancellation 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 AEC library functions.

The main function calls the `WM8731Init` function from the AEC library, which initializes the AEC 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 read for a full frame of data. The contents of the coded data buffers are copied into the `nearEndIn` array and the `EC_apply()` function from the AEC library is called with `nearEndIn` as the input data frame. The `nearEndIn` data buffer, which is also the output of the `EC_apply` function after it has been executed, is transferred to the UART for transmission to board 2. The UART data is converted from µ-Law to 16-bit linear and stored in `farEndIn`, which is the reference (echo) input data frame to `EC_apply()`.

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

© 2004-2011 Microchip Technology Inc.

Chapter 4. Application Programming Interface (API)

This chapter describes the Application Programming Interface (API) to the AEC library. Topics covered include:

- Adding the AEC Library to an Application
- Memory Model Compile Options
- AEC Algorithm Overview
- Library Usage
- Register Usage
- AEC Library API Functions
- Application Tips

4.1 ADDING THE AEC LIBRARY TO AN APPLICATION

To use the AEC library in an application, the library archive must be added to the application project workspace, and the `ec_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 AEC 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.

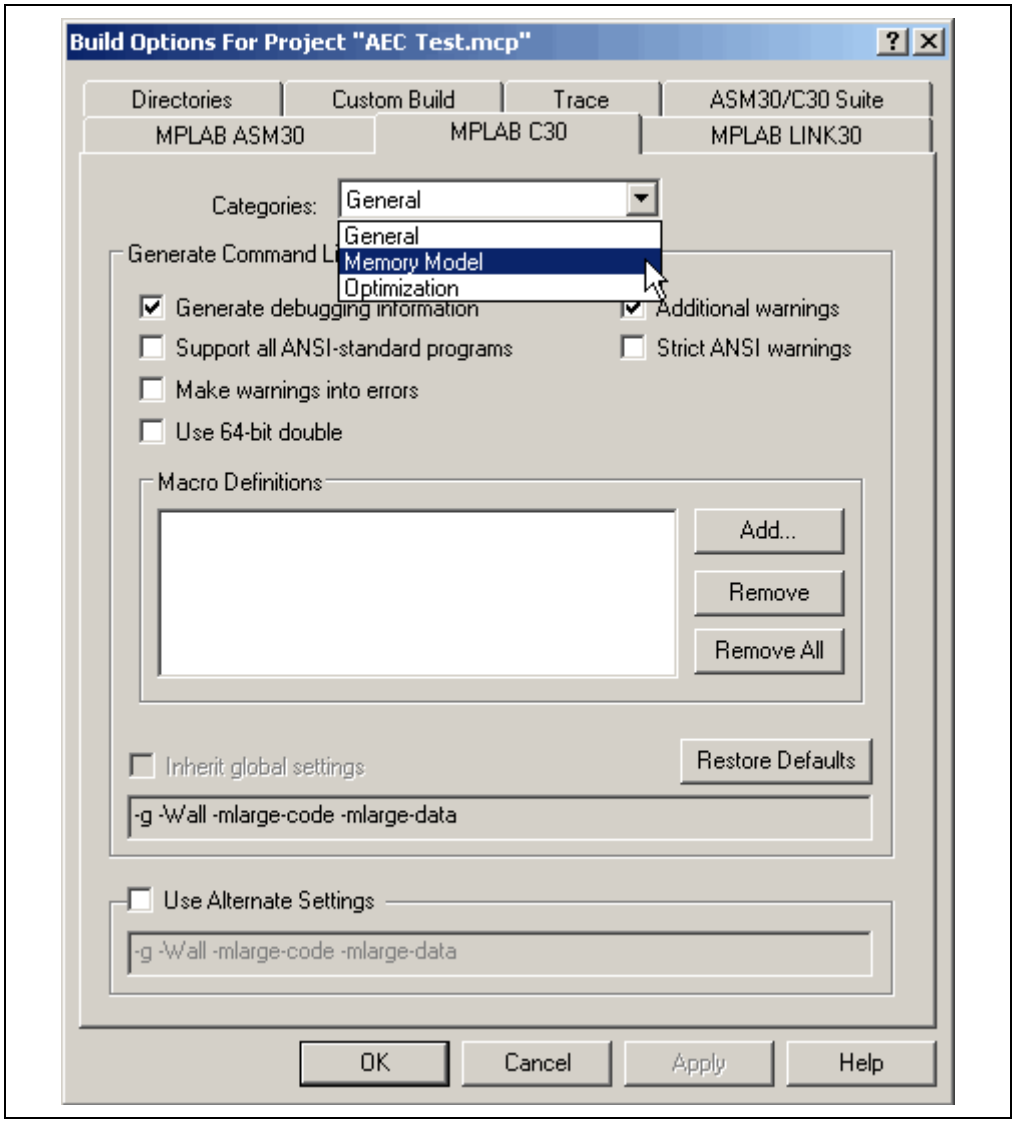
© 2004-2011 Microchip Technology Inc.

4.2 MEMORY MODEL COMPILE OPTIONS

While using the AEC library with the 128 ms tail length option 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.

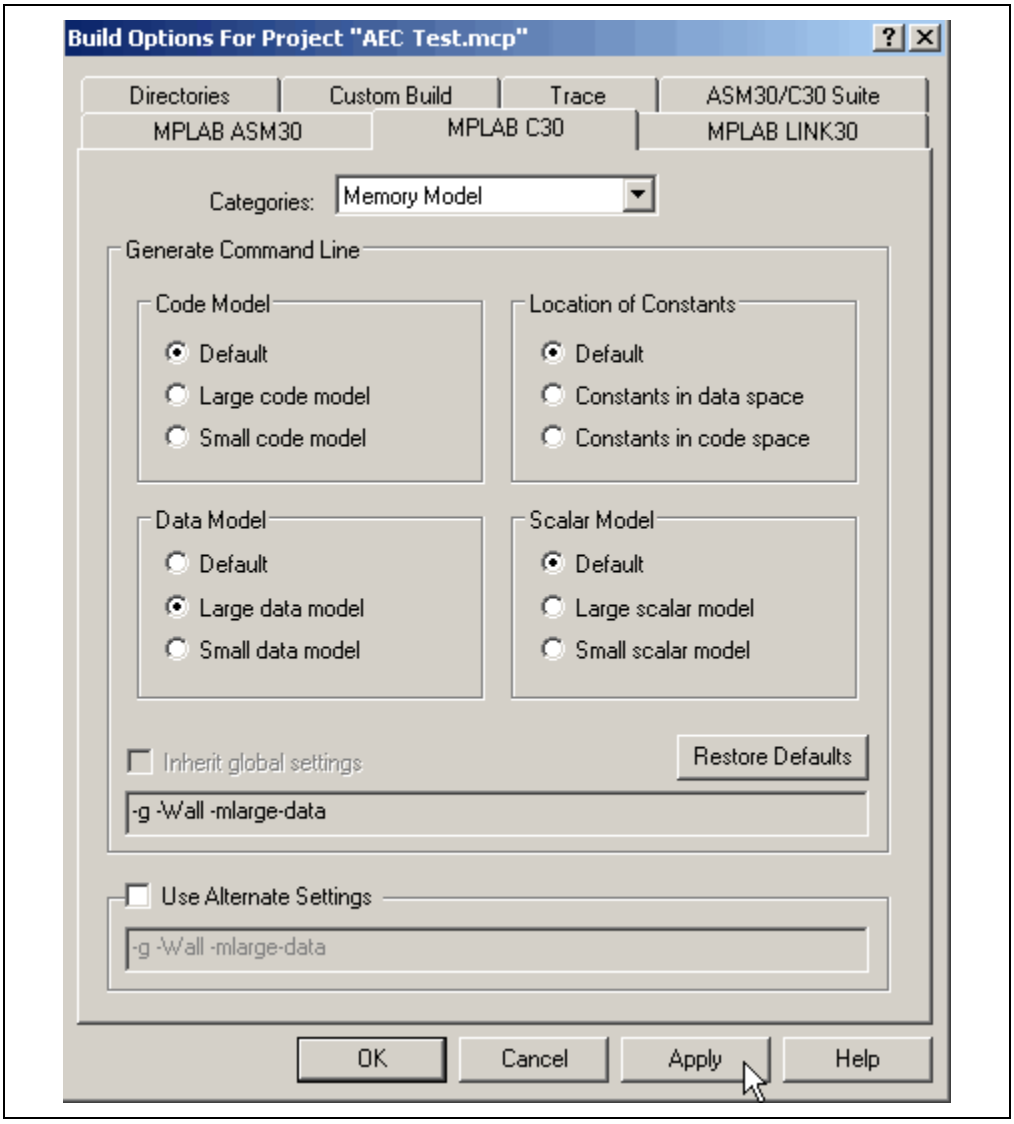
FIGURE 4-1: PROJECT BUILD OPTIONS



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

© 2004-2011 Microchip Technology Inc.

FIGURE 4-2: COMPILER MEMORY MODEL SETTINGS



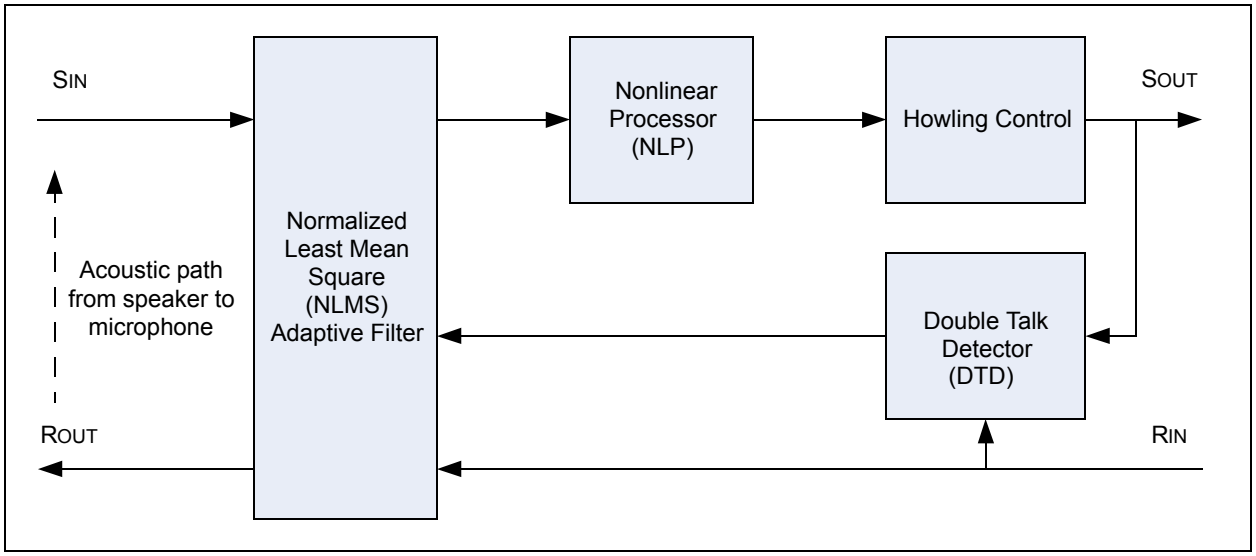
3. Click **Apply**, and then click **OK**.

This completes the procedure.

4.3 AEC ALGORITHM OVERVIEW

This section describes the AEC algorithm. A conceptual block diagram illustrating the operation of the AEC algorithm is shown in Figure 4-3.

FIGURE 4-3: AEC ALGORITHM FUNCTIONAL BLOCKS



The AEC algorithm can be divided into these functions:

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

A typical AEC system involves these 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

The systems in which the near-end speaker and microphone do not have sufficient acoustic separation, the SIN signal not only contains the microphone input (presumably spoken by a talker at the near-end), but also an undesirable echo generated by the acoustic path from the speaker to the microphone. This signal is then transmitted to the far-end through the communication channel (wired or wireless), with the result that the listener at the other end hears a perceptible echo of his/her own speech. Traditionally, this problem was avoided by allowing only one person to talk at any given time (that is, by not allowing "double talk").

An AEC algorithm consists of an adaptive filter and various associated control functions, which not only eliminate the acoustic echo but also enables double talk (that is, full-duplex operation). This algorithm operates at the same communicating node at which the echo was generated. The control functions used in the AEC algorithm, in conjunction with the NLMS adaptive filter, are Double Talk Detector, Nonlinear Processor and Howling Control.

© 2004-2011 Microchip Technology Inc.

4.3.1 Normalized Least Mean Square (NLMS) Adaptive Filter

NLMS is the fundamental adaptation algorithm used for estimating and canceling out the acoustic 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 acoustic path between the speaker and the microphone. The filter thereby produces an echo estimate of RIN, which is then subtracted from the SIN signal.

4.3.2 Nonlinear Processor (NLP)

The AEC 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. 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 send path between the output of the NLMS filter and the SOUT 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 AEC library offers two functions for using the NLP. The `EC_applyNLP()` function applies the NLP action to the input buffer. The `EC_setNLPLevel()` function varies the level of attenuation.

4.3.3 Howling Control

Howling is a typical problem in full-duplex communication. It builds up due to the acoustic feedback path. One way to reduce howling is to shift the frequency of the signal that is picked up by the microphone by 10 Hz to 20 Hz, before it is sent out over a communication channel. This shift is usually not perceived as unnatural by the human ear. The shifted signal appears at the destination loudspeakers and travels back to the originator, shifted by another 10 Hz to 20 Hz. The signal travels many times through this acoustic path and is quickly shifted out of the pass band, thereby reducing the problem of unpleasant feedback.

The AEC library offers a function, `EC_setHowlingControl()`, to enable and disable howling control.

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 acoustic echo. To prevent such a condition, a 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 near-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 AEC library offers two functions, `EC_setDoubleTalkHangover()` and `EC_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 AEC 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 `EC_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 AEC 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 AEC library:

1. **Set the Echo Tail Length:** In the file `ec_api.h`, set the `EC_ECHOTAIL` value to the desired echo tail length. The valid values are 8 ms, 16 ms, 32 ms, 64 ms and 128 ms. Note that setting an invalid value will cause the `EC_init()` function to return a `EC_ORDERERROR` value. This coding step is shown in Example 4-1.

EXAMPLE 4-1: SETTING THE ECHO TAIL LENGTH

```
.
.
.
#ifndef __EC_API_H__
#define __EC_API_H__

/* Set the desired echo tail length */
#define EC_ECHOTAIL 64 /* Step 1 */

// Nothing below this line should be changed
#define EC_FRAME    80
#define EC_FALSE    0
#define EC_TRUE     1
.
.
.
```

Steps 2 through 6 should be performed in the user application. Refer to Example 4-2 for the actual code required.

2. **Allocate the memory for the AEC algorithm state holder:** This memory is an integer array in X memory aligned at an address boundary of 2 bytes. The `EC_XSTATE_MEM_SIZE_INT` macro specifies the size of this array. Every audio channel to be processed will requires it own state holder.
3. **Allocate the memory for the AEC algorithm X and Y scratch memories:** The X scratch memory is an integer array in X memory aligned at an address boundary of 4 bytes. The Y scratch memory is an integer array in Y memory aligned at an address boundary of 2 bytes. Multiple audio channels can share the same scratch memories.

> **Note:** In some dsPIC33E devices, the Y memory is located in the EDS. In such cases, the Y scratch memory array must be tagged with the `__eds__` keyword and assigned an `eds` attribute.

4. **Initialize the AEC algorithm state for each audio channel:** Use the `EC_init()` function for initializing the acoustic echo cancellation state for each audio channel.
5. **Apply the AEC to an audio frame:** Use the `EC_apply()` function to perform echo cancellation on an audio frame. If a frame is not required to be processed by the AEC algorithm, the function should still be called with the enable parameter set to `EC_FALSE`. This will allow the AEC algorithm to continue adapting to the echo in the audio frame. The audio frame stays unaffected.
6. **Use the NLP function:** Use the `EC_applyNLP()` function to suppress residual echo.

© 2004-2011 Microchip Technology Inc.

EXAMPLE 4-2: LIBRARY USAGE EXAMPLE

```
// Channel 1 memory structures.
int ecStateMemX1 [EC_XSTATE_MEM_SIZE_INT]   _XBSS(2); /* Step 2 */

// Channel 2 memory structures
int ecStateMemX2 [EC_XSTATE_MEM_SIZE_INT]   _XBSS(2); /* Step 2 */

// Each instance can share the same X and Y scratch memory
int ecScratchX   [EC_XSCRATCH_MEM_SIZE_INT] _XBSS(2); /* Step 3 */
int ecScratchY   [EC_YSCRATCH_MEM_SIZE_INT] _YBSS(2); /* Step 3 */
.
.
.
void main()
{
   EC_init(ecStateMemX1, ecScratchX, EC_ORDER); /* Step 4 */
   EC_init(ecStateMemX2, ecScratchX, EC_ORDER); /* Step 4 */
   .
   .
   .
   While(1)
   {
      EC_apply(ecStateMemX1, ecScratchY, nearEndIn1, farEndIn1, EC_TRUE); /* Step 5 */
      EC_applyNLP(ecStateMemX1, nearEndIn1, EC_TRUE); /* Step 6 */

      EC_apply(ecStateMemX2, ecScratchY, nearEndIn2, farEndIn2, EC_TRUE); /* Step 5 */
      EC_applyNLP(ecStateMemX2, nearEndIn2, EC_TRUE); /* Step 6 */
   }
}
```

4.5 REGISTER USAGE

The dsPIC DSC Acoustic Echo Cancellation 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, especially if the interrupt has occurred while a library function is executing.

An ISR may access constants in program memory through a Program Space Visibility (PSV) read operation. In case of dsPIC33F, it is suggested that the application explicitly manage PSV operations while using the AEC library. The application should ensure that PSV access is enabled and that the PSVPAG register contains the required value. In case of dsPIC33E, the library does not use PSV.

4.6 RESOURCE REQUIREMENTS

The AEC 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	6810 (dsPIC30F/dsPIC33F) 7032 (dsPIC33E)	`.libec`
Tables in Program Memory	2154 (dsPIC30F/dsPIC33F) 0 (dsPIC33E)	`.const`
Total Program Memory	8964 (dsPIC30F/dsPIC33F) 7032 (dsPIC33E)	—

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

Function	Size (bytes)	Alignment	Section
`ecStateMemX`	2240	2	X data memory
`scratchMemX`	1504	2	X data memory
`scratchMemY`	1184	2	Y data memory
`nearEndIn`	160	2	X or Y data memory
`farEndIn`	160	2	X or Y data memory
Tables in Data Memory	0 (dsPIC30F/dsPIC33F) 1436 (dsPIC33E)	2	X data memory
Total Data Memory	5248 (dsPIC30F/dsPIC33F) 6684 (dsPIC33E)	—	—

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
`EC_init()`	All	< 0.5	Once
`EC_apply() + EC_applyNlP()`	8	3	10 ms
`EC_apply() + EC_applyNlP()`	16	5	10 ms
`EC_apply() + EC_applyNlP()`	32	8	10 ms
`EC_apply() + EC_applyNlP()`	64	14	10 ms
`EC_apply() + EC_applyNlP()`	128	27	10 ms
All other functions	All	Minimal	As required

Note: Enabling the Howling Control feature requires an additional 8 MIPS.

4.6.1 Data Format

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

© 2004-2011 Microchip Technology Inc.

4.7 AEC LIBRARY API FUNCTIONS

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

- EC_init
- EC_relocateXScratchMem
- EC_apply
- EC_applyNLP
- EC_setNLPLevel
- EC_getNLPLevel
- EC_setDoubleTalkHangover
- EC_getDoubleTalkHangover
- EC_setAdaptionHangover
- EC_getAdaptionHangover
- EC_setHowlingControl
- EC_getHowlingControl
- EC_setForceAdapt
- EC_getForcedAdapt
- EC_setInhibitAdaption
- EC_getInhibitAdaption
- EC_estimateERLE
- EC_TRUE
- EC_FALSE
- EC_FRAME
- EC_ECHOTAIL
- EC_ORDER
- EC_XSTATE_MEM_SIZE_INT
- EC_XSCRATCH_MEM_SIZE_INT
- EC_YSCRATCH_MEM_SIZE_INT
- EC_DOUBLETALKHANGOVERDEFAULT
- EC_ADAPTIONHANGOVERDEFAULT
- EC_NLPLEVELDEFAULT
- EC_HOWLINGCONTROLDEFAULT
- EC_FORCEADAPTDEFAULT
- EC_ORDERERROR
- EC_OK

EC_init

Description

Initializes the AEC algorithm.

Include

`ec_api.h`

Prototype

```
int EC_init(int* ptrStateX, int* xScratchMem, int order);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC
`xScratchMem`	a pointer to an area of scratch memory in X RAM used by the AEC algorithm
`order`	an integer representing the filter order (`EC_ORDER`)

Return Value

`EC_OK`	If the value of `EC_ORDER` is correct
`EC_ORDERERROR`	If the value of `EC_ORDER` is incorrect

Remarks

The value of order can be set to `EC_ORDER`, which is a macro in `ec_api.h` that calculates the filter order based on the echo tail length required.

Code Example

```
int ecStateMemX     [EC_XSTATE_MEM_SIZE_INT]      _XBSS(2);
int ecScratchX      [EC_XSCRATCH_MEM_SIZE_INT]    _XBSS(2);
int ecScratchY      [EC_YSCRATCH_MEM_SIZE_INT]    _YBSS(2);
.
.
.
EC_init(ecStateMemX, ecScratchX, EC_ORDER);
```


© 2004-2011 Microchip Technology Inc.

EC_relocateXScratchMem

Description

Changes the X scratch memory buffer used by the AEC algorithm.

Include

`ec_api.h`

Prototype

```
void EC_relocateXScratchMem(int* ptrStateX, int* xScratchMem);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC
`xScratchMem`	a pointer to the new X scratch memory to be used by the AEC algorithm

Return Value

None.

Remarks

After relocating the X scratch memory, the older scratch memory is not used by the AEC library.

Code Example

```
int ecStateMemX     [EC_XSTATE_MEM_SIZE_INT]      _XBSS(2);
int ecScratchX      [EC_XSCRATCH_MEM_SIZE_INT]    _XBSS(2);
int ecScratchX2     [EC_XSCRATCH_MEM_SIZE_INT]    _XBSS(2);
int ecScratchY      [EC_YSCRATCH_MEM_SIZE_INT]    _YBSS(2);
.
.
.
EC_init(ecStateMemX, ecScratchX, EC_ORDER);
```

The AEC algorithm is using `ecScratchX` for its scratch memory.

```
.
.
.
EC_relocateXScratchMem(ecStateMemX, ecScratchX2);
```

The AEC algorithm is now using `ecScratchX2` for its scratch memory.

EC_apply

Description

Applies acoustic echo cancellation to the current frame of data.

Include

`ec_api.h`

Prototype

```
void EC_apply(int* ptrStateX, int* yScratchMem, int* nearEndIn,
int* farEndIn, int enable);
```

Arguments

`ptrStateX`	a pointer to the X memory for this instance of AEC
`yScratchMem`	a pointer to an area of scratch memory in Y RAM used by the AEC algorithm
`nearEndIn`	a pointer to the near-end speech input (SIN) signal on buffer of size `EC_FRAME`. This buffer can be in X or Y memory
`farEndIn`	a pointer to the far-end speech receive input (RIN) signal on buffer of size `EC_FRAME`. This buffer can be in X or Y memory
`enable`	a flag to indicate if AEC is required for this buffer (`EC_TRUE` or `EC_FALSE`)

Return Value

None.

Remarks

The AEC algorithm is process-in-place, meaning that the output is passed back in the input buffer. Setting Enable to `EC_FALSE` returns an un-processed buffer of data, but the AEC algorithm still adapts on the data.

Code Example

```
int ecStateMemX     [EC_XSTATE_MEM_SIZE_INT]      _XBSS(2);
int ecScratchY      [EC_YSCRATCH_MEM_SIZE_INT]    _YBSS(2);
int ecScratchX      [EC_XSCRATCH_MEM_SIZE_INT]    _XBSS(2);
.
.
.
EC_init(ecStateMemX, ecScratchX, EC_ORDER);
.
.
.
EC_apply(ecStateMemX, ecScratchY, nearEndIn, farEndIn, EC_TRUE);
```


© 2004-2011 Microchip Technology Inc.

EC_applyNLP

Description

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

Include

`ec_api.h`

Prototype

```
void EC_applyNLP(int* ptrStateX, int* nearEndIn, int enable);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC
`nearEndIn`	a pointer to the input/output buffer of size `EC_FRAME`
`enable`	a flag to indicate if NLP is required for this buffer (`EC_TRUE` or `EC_FALSE`)

Return Value

None.

Remarks

None.

Code Example

```
int ecStateMemX  [EC_XSTATE_MEM_SIZE_INT]   _XBSS(2);
int ecScratchY   [EC_YSCRATCH_MEM_SIZE_INT] _YBSS(2);
int ecScratchX   [EC_XSCRATCH_MEM_SIZE_INT] _XBSS(2);
.
.
.
EC_init(ecStateMemX, ecScratchX, EC_ORDER);
.
.
.
EC_apply(ecStateMemX, ecScratchY, nearEndIn, farEndIn, EC_TRUE);
EC_applyNLP(ecStateMemX, nearEndIn, EC_TRUE);
```

EC_setNLPLevel

Description

Sets the required level of NLP.

Include

`ec_api.h`

Prototype

```
void EC_setsetNLPLevel(int* ptrStateX, int level);
```

Arguments

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

Return Value

None.

Remarks

Each value represents a 6 dB change in the NLP level. Therefore, a value of 3 translates to an NLP level of 18 dB.

Code Example

```
EC_setNLPLevel(ecStateMemX, 3);
```

Sets the desired NLP level to 3 for the instance of the algorithm `ecStateMemX`.

© 2004-2011 Microchip Technology Inc.

EC_getNLPLevel

Description

Returns the current NLP level.

Include

`ec_api.h`

Prototype

```
int EC_getNLPLevel(int* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the state memory for this instance of AEC

Return Value

The current NLP level as an integer for the instance of the algorithm `ecStateMemX`.

Remarks

None.

Code Example

```
int nlpLevel;
nlpLevel = EC_getNLPLevel(ecStateMemX);
```

`nlpLevel` contains the current NLP level for the instance of the algorithm `ecStateMemX`.

EC_setDoubleTalkHangover

Description

Sets the hangover value for the double talk detection portion of the AEC algorithm.

Include

`ec_api.h`

Prototype

```
void EC_setDoubleTalkHangover(int* ptrStateX, int frames);
```

Arguments

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

Return Value

None.

Remarks

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

Code Example

```
EC_setDoubleTalkHangover(ecStateMemX, 10);
```

Sets the desired double talk hangover to 10 frames (representing 100 ms of data) for the instance of the algorithm `ecStateMemX`.

© 2004-2011 Microchip Technology Inc.

EC_getDoubleTalkHangover

Description

Returns the current double talk detection hangover.

Include

`ec_api.h`

Prototype

```
int EC_getDoubleTalkHangover(int* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the state memory for this instance of AEC

Return Value

The current double talk detection hangover value.

Remarks

None.

Code Example

```
int doubleTalkHangover;
doubleTalkHangover = EC_getDoubleTalkHangover(ecStateMemX);
```

`doubleTalkHangover` contains the double talk detection hangover value set for the instance of the algorithm `ecStateMemX`.

EC_setAdaptionHangover

Description

Sets the hangover value for the adaptation update portion of the AEC algorithm.

Include

`ec_api.h`

Prototype

```
void EC_setAdaptionHangover(int* ptrStateX, int frames);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC
`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. Setting adaptation hangover to '`1`' disables the hysteresis.

Code Example

```
EC_setAdaptionHangover(ecStateMemX, 3);
```

Sets the desired adaptation hangover to 3 frames (representing 30 ms of data) for the instance of the algorithm `ecStateMemX`.

© 2004-2011 Microchip Technology Inc.

EC_getAdaptionHangover

Description

Returns the current adaptation hangover.

Include

`ec_api.h`

Prototype

```
int EC_getAdaptionHangover(int* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the state memory for this instance of AEC

Return Value

The current adaptation hangover value.

Remarks

None.

Code Example

```
int adaptionHangover;
adaptionHangover = EC_getAdaptionHangover(ecStateMemX);
```

`adaptionHangover` contains the adaptation hangover value set for the instance of the algorithm `ecStateMemX`.

EC_setHowlingControl

Description

Enables or disables the howling control.

Include

`ec_api.h`

Prototype

```
void EC_setHowlingControl(int* ptrStateX, int howlingControl);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC
`howlingControl`	a flag to indicate if howling control is required (`EC_TRUE` or `EC_FALSE`)

Return Value

None.

Remarks

None.

Code Example

```
EC_setHowlingControl(ecStateMemX, EC_TRUE);
```

Enables the howling control for the instance of the algorithm `ecStateMemX`.

© 2004-2011 Microchip Technology Inc.

EC_getHowlingControl

Description

Returns the current state of the howling control.

Include

`ec_api.h`

Prototype

```
int EC_getHowlingControl(int* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the state memory for this instance of AEC

Return Value

Returns the current state of howling control.

Remarks

None.

Code Example

```
int howlingControl;
howlingControl = EC_getHowlingControl(ecStateMemX);
```

`howlingControl` contains the howling control state set for the instance of the algorithm `ecStateMemX`.

EC_setForceAdapt

Description

Enables or disables the forced adaptation of the AEC algorithm.

Include

`ec_api.h`

Prototype

```
void EC_setForceAdapt(int* ptrStateX, int state);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC
`state`	either `EC_TRUE` or `EC_FALSE`

Return Value

None.

Remarks

None.

Code Example

```
EC_setForceAdapt(ecStateMemX, EC_TRUE);
```

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

© 2004-2011 Microchip Technology Inc.

EC_getForcedAdapt

Description

Returns the state of the forced adaptation.

Include

`ec_api.h`

Prototype

```
int EC_getForceAdapt(int* ptrStateX);
```

Arguments

`ptrStateX` a pointer to the state memory for this instance of AEC

Return Value

The current state of forced adaptation.

Remarks

None.

Code Example

```
int forceAdapt;
forceAdapt = EC_getForceAdapt(ecStateMemX);
```

`forceAdapt` contains the forced adaptation value set for the instance of the algorithm `ecStateMemX`, either `EC_TRUE` or `EC_FALSE`.

EC_setInhibitAdaption

Description

Sets the state of the inhibit adaptation.

Include

`ec_api.h`

Prototype

```
void EC_setInhibitAdaption(int* ptrStateX, int state);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC
`state`	either `EC_TRUE` or `EC_FALSE`

Return Value

None.

Remarks

None.

Code Example

```
EC_setInhibitAdaption(ecStateMemX, EC_TRUE);
```

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

© 2004-2011 Microchip Technology Inc.

EC_getInhibitAdaption

Description

Returns the state of the inhibit adaptation.

Include

`ec_api.h`

Prototype

```
long EC_getInhibitAdaption(int* ptrStateX);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of AEC

Return Value

`EC_TRUE`	If adaptation is inhibited
`EC_FALSE`	If adaptation is not inhibited

Remarks

None.

Code Example

```
long inhibitAdaption;
inhibitAdaption = EC_getInhibitAdaption(ecStateMemX);
```

`inhibitAdaption` contains the inhibit adaptation value set for the instance of the algorithm `ecStateMemX`, either `EC_TRUE` or `EC_FALSE`.

EC_estimateERLE

Description

Estimates the Echo Return Loss Enhancement (ERLE).

Include

`ec_api.h`

Prototype

```
int EC_estimateSNR(int* preEC, int* postEC);
```

Arguments

`preEC`	a pointer to the buffer of data before the AEC algorithm has been applied
`postEC`	a pointer to the buffer of data after the AEC algorithm has been applied

Return Value

An estimate of the ERLE.

Remarks

This is a crude estimate of the ERLE of the current frame of data.

Code Example

```
int erle;
for (i = 0; i < EC_FRAME; i++)
{
   nearEndBefore[i] = nearEndIn[i];
}
EC_apply(ecStateMemX, ecScratchY, nearEndIn, farEndIn, EC_TRUE);
EC_applyNLP(ecStateMemX, nearEndIn, EC_TRUE);
erle = EC_estimateSNR(nearEndBefore, nearEndIn);
```

`erle` contains the estimate of the ERLE (in dB).

© 2004-2011 Microchip Technology Inc.

EC_TRUE

Description

Used to indicate true to the AEC algorithm.

Value

```
1
```

EC_FALSE

Description

Used to indicate false to the AEC algorithm.

Value

```
0
```

EC_FRAME

Description

The size of the input buffer processed.

Value

```
80
```

EC_ECHOTAIL

Description

The length of the echo tail length required (8 ms, 16 ms, 32 ms, 64 ms or 128 ms).

Value

```
64
```

© 2004-2011 Microchip Technology Inc.

EC_ORDER

Description

The size of the input buffer processed.

Value

```
(EC_ECHOTAIL * 8)
```

EC_XSTATE_MEM_SIZE_INT

Description

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

Value

```
((EC_ORDER * 2) + 96)
```

EC_XSCRATCH_MEM_SIZE_INT

Description

Size in integers of the memory location required for the X-Scratch memory.

Value

```
(EC_ORDER + (EC_FRAME * 3))
```

EC_YSCRATCH_MEM_SIZE_INT

Description

Size in integers of the memory location required for the Y-Scratch memory.

Value

```
(EC_ORDER + EC_FRAME)
```


© 2004-2011 Microchip Technology Inc.

EC_DOUBLETALKHANGOVERDEFAULT

Description

Value in frames for the double talk hangover.

Value

3

EC_ADAPTIONHANGOVERDEFAULT

Description

Value in frames for the default adaptation hangover.

Value

1

EC_NLPLEVELDEFAULT

Description

Value of the default NLP attenuation (EC_NLPLEVELDEFAULT * 6 dB).

Value

3

EC_HOWLINGCONTROLDEFAULT

Description

Boolean value of the state of the Howling Control module.

Value

EC_FALSE

EC_FORCEADAPTDEFAULT

Description

Boolean value to enable forced adaptation.

Value

EC_FALSE

EC_ORDERERROR

Description

Return value from EC_init to indicate that the specified order is not valid.

Value

0x8001

EC_OK

Description

Return value from EC_init to indicate that the specified order is valid.

Value

0x0000

© 2004-2011 Microchip Technology Inc.

4.8 APPLICATION TIPS

Although the AEC algorithm makes a best effort to suppress the echo in a system, its performance can be optimized by proper selection of parameters. In general, experimentation with this library is encouraged.

Some tips for affecting the performance of the algorithm are as follows:

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, then good use is being made of the available precision; levels higher than this carry a risk of amplitude clipping.
2. Choose a NLP level that best fits the application. A very high NLP level not only suppresses the residual echo, but may also reduce the speech level to some extent.
3. The Double Talk Detection Hangover can be used to adjust the window during which the AEC algorithm will wait before applying NLP to the send signal. If the application is experiencing some leading edge speech clipping, then try increasing the DTD Hangover.
4. The Adaptation Hangover can be used to adjust the window during which an adaptation hangover 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 AEC 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.

NOTES:

© 2004-2011 Microchip Technology Inc.

Index

A

Acoustic Echo Cancellation
- Typical Applications ... 14

Algorithm
- Normalized Least Mean Square (NLMS) ... 14

API Functions ... 39
- ... 42
- EC_ADAPTIONHANGOVERDEFAULT ... 59
- EC_apply ... 42
- EC_applyNLP ... 43
- EC_DOUBLETALKHANGOVERDEFAULT ... 59
- EC_ECHOTAIL ... 57
- EC_estimateERLE ... 56
- EC_FALSE ... 57
- EC_FORCEADAPTDEFAULT ... 60
- EC_FRAME ... 57
- EC_getAdaptionHangover ... 49
- EC_getDoubleTalkHangover ... 47
- EC_getForceAdapt ... 53
- EC_getHowlingControl ... 51
- EC_getInhibitAdaption ... 55
- EC_getNLPLevel ... 45
- EC_HOWLINGCONTROLDEFAULT ... 59
- EC_init ... 40
- EC_NLPLEVELDEFAULT ... 59
- EC_OK ... 60
- EC_ORDER ... 58
- EC_ORDERERROR ... 60
- EC_relocateXScratchMem ... 41
- EC_setAdaptionHangover ... 48
- EC_setDoubleTalkHangover ... 46
- EC_setForceAdapt ... 52
- EC_setHowlingControl ... 50
- EC_setInhibitAdaption ... 54
- EC_setNLPLevel ... 44
- EC_TRUE ... 57
- EC_XSCRATCH_MEM_SIZE_INT ... 58
- EC_XSTATE_MEM_SIZE_INT ... 58
- EC_YSCRATCH_MEM_SIZE_INT ... 58

C

Customer Notification Service ... 9
Customer Support ... 9

D

Demo Code Description ... 26, 30
Demonstration Procedure ... 25, 29
Demonstration Setup ... 22, 28
Demonstration Summary ... 21, 27
Device Support ... 15
Documentation
- Conventions ... 6
- Layout ... 5

F

FIR Filter ... 14

H

Host System Requirements ... 15

I

Installation
- Installing the Library ... 17

Internet Address ... 8

L

Library Files
- demo Folder ... 18
- doc Folder ... 19
- h Folder ... 19
- lib Folder ... 19

M

Microchip Internet Web Site ... 8

N

NLMS ... 14
Normalized Least Mean Square (NLMS) Adaptive Filter ... 35
- Double Talk Detector (DTD) ... 35
- Howling Control ... 35
- Nonlinear Processor ... 35

Normalized Least Mean Square Algorithm ... 14

O

Overview
- Acoustic Echo Cancellation ... 13

R

Reading, Recommended ... 7
Resource Requirements
- Computational Speed ... 38
- Data Memory Usage ... 38
- Estimated Dynamic Memory Usage ... 38
- Program Memory Usage ... 38

W

Warranty Registration ... 7
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

© 2004-2011 Microchip Technology Inc.