



MICROCHIP

**dsPIC[®] DSC Automatic Gain
Control (AGC) Library
User's Guide**

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

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

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

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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC³² 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.

© 2009-2011, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-61341-371-5

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.

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



dsPIC[®] DSC AUTOMATIC GAIN CONTROL (AGC) LIBRARY USER'S GUIDE

Table of Contents

Preface	5
Chapter 1. Introduction	
1.1 Automatic Gain Control Library Overview	11
1.2 Features	12
1.3 Host System Requirements	12
Chapter 2. Installation	
2.1 Installation Procedure	13
2.2 AGC Library Files	14
Chapter 3. Quick Start Demonstration	
3.1 Quick Start Demonstration for the dsPIC33F Device Family	17
3.2 Quick Start Demonstration for dsPIC33E Device Family	22
Chapter 4. Application Programming Interface	
4.1 Adding the AGC Library to an Application	25
4.2 AGC Operation and Parameters	26
4.3 External Gain Control	27
4.4 Using the AGC Library	27
4.5 Resource Requirements	29
4.6 AGC Library API Functions	30
4.7 Application Tips	59
Index	61
Worldwide Sales and Service	64

NOTES:



dsPIC[®] DSC AUTOMATIC GAIN CONTROL (AGC) LIBRARY USER'S GUIDE

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 “DSXXXXA”, where “XXXX” 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 preface contains general information that will be useful to know before using the dsPIC[®] DSC Automatic Gain Control (AGC) 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 AGC library. The document is organized as follows:

- **Chapter 1. “Introduction”** – This chapter introduces the AGC library and provides a brief overview of noise suppression and the library features. It also outlines requirements for a host PC.
- **Chapter 2. “Installation”** – This chapter provides detailed information needed to install the AGC library demonstration on a PC.
- **Chapter 3. “Quick Start Demonstration”** – This chapter provides a hands-on demonstration of automatic gain control in a working application.
- **Chapter 4. “Application Programming Interface”** – This chapter outlines how the API functions provided in the AGC library can be included in your application software via the Application Programming Interface (API).

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	<i>MPLAB[®] IDE User's Guide</i>
	Emphasized text	...is the <i>only</i> compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	<u><i>File>Save</i></u>
Bold characters	A dialog button	Click OK
	A tab	Click the Power tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	#define START
	Filenames	autoexec.bat
	File paths	c:\mcc18\h
	Keywords	_asm, _endasm, static
	Command-line options	-Opa+, -Opa-
	Bit values	0, 1
	Constants	0xFF, 'A'
Italic Courier New	A variable argument	<i>file.o</i> , where <i>file</i> can be any valid filename
Square brackets []	Optional arguments	mcc18 [options] <i>file</i> [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 dsPICDSC Automatic Gain Control (AGC) 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 MCU 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 MCU family architecture and peripheral modules, but do not cover the specifics of each device. Refer to the specific device data sheet for 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 MCU family architecture and peripheral modules, but do not cover the specifics of each device. Refer to the specific device data sheet for information.

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

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

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 <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 Microchip MPLAB REAL ICE[™] in-circuit emulator.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debuggers, 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
- 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 (August 2009)

This is the initial released version of this document.

Revision B (July 2011)

This revision includes the following updates:

- Figures:
 - Removed Figure 2-1 through Figure 2-3 in **2.1 “Installation Procedure”**
 - Updated the title to **Figure 3-1**
 - Removed Figure 3-3 and Figure 3-4 in **3.1.2.2 “Program the dsPIC DSC Device”**
 - Removed Figure 4-1 in **4.1 “Adding the AGC Library to an Application”**
- Notes:
 - Added a note in **3.1.2.1 “Configure the dsPICDEM 1.1 Plus Development Board”**
- Sections:
 - Updated **“Preface”**
 - Updated the first paragraph in **Chapter 1. “Introduction”**
 - Updated the AGC Library features in **1.2 “Features”**
 - Updated **2.1 “Installation Procedure”**
 - Updated **2.2.4 “lib Folder”**
 - Updated the first paragraph in **Chapter 3. “Quick Start Demonstration”**
 - Updated the existing section in **Chapter 3. “Quick Start Demonstration”** to **3.1 “Quick Start Demonstration for the dsPIC33F Device Family”**
 - Added **3.2 “Quick Start Demonstration for the dsPIC33E Device Family”**
 - Updated **3.1.2.2 “Program the dsPIC DSC Device”**
 - Updated step 2 in **4.1 “Adding the AGC Library to an Application”**
 - Updated the file names in the first paragraph, in **4.4 “Using the AGC Library”**
- Tables:
 - Updated **Table 2-1** in **2.2.1 “demo Folder”**
 - Removed **Table 2-2** in **2.2.2 “docs Folder”**
 - Updated **Table 2-4** in **2.2.4 “lib Folder”**
- Additional minor corrections such as language and formatting updates were incorporated throughout the document

NOTES:

Chapter 1. Introduction

The dsPIC® DSC Automatic Gain Control (AGC) Library, referred to as the AGC library, supports the dsPIC33F and dsPIC33E device families and provides an algorithm to automatically control the level of a signal. This chapter provides an overview of the AGC library.

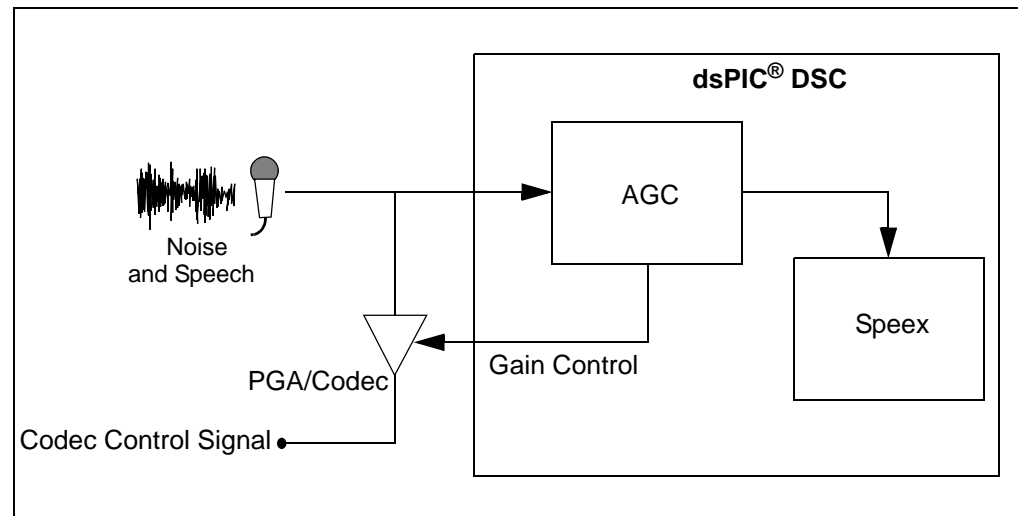
The following topics are covered in this chapter:

- [Automatic Gain Control Library Overview](#)
- [Features](#)
- [Host System Requirements](#)

1.1 AUTOMATIC GAIN CONTROL LIBRARY OVERVIEW

The AGC library is useful in speech and audio applications where the distance varies between the system user and the microphone. The main function of the AGC algorithm is to estimate the short-term peak amplitude of the input speech, and then apply a gain factor to bring it to a desired level set by the user. If no speech is detected, the gain will gradually “leak” back down to a preset default level. It is assumed that a system that utilizes AGC has been correctly set up for normal use with close proximity to the microphone. Therefore, the AGC library maintains the input signal level to the subsequent processing blocks in the signal processing chain. The rate at which the gain changes are applied to the input signal can be controlled. [Figure 1-1](#) shows an example of how the AGC can be used.

FIGURE 1-1: AGC IN AN APPLICATION



While the AGC library is implemented entirely in software, it can also be used to control the gain of an external codec and a software function is provided for this. The library will also flag amplitude clipping on the input.

1.2 FEATURES

The AGC library has the following features:

- Simple user interface – only one library file and one header file
- All functions are called from a C application program
- Full compliance with the Microchip C30 Compiler, Assembler and Linker
- Highly optimized assembly code that uses DSP instructions and advanced addressing modes
- Comprehensive Application Programming Interface (API) provides parametric control of the AGC engine
- Customized for speech applications
- Input signal clip detection
- Software functions to control the gain of the external codec
- Gain attack, release and leakage rate controls
- Audio bandwidth: 8 kHz to 48 kHz sampling rate
- Library available for download from the Microchip web site, which also includes:
 - Sample demonstration application
 - User's guide
 - Sample WAVE (.wav) files

1.3 HOST SYSTEM REQUIREMENTS

The AGC library requires a PC-compatible host system with these minimum characteristics:

- 1 GHz or higher processor
- HTML browser
- 16 MB RAM
- 40 MB available hard drive space
- Microsoft[®] Windows[®] 98, Windows 2000, Windows NT, Windows XP, Windows Vista or Windows 7



dsPIC[®] DSC AUTOMATIC GAIN CONTROL (AGC) LIBRARY USER'S GUIDE

Chapter 2. Installation

This chapter describes the various files in the AGC library and includes instructions for installing the library on your laptop or desktop PC. After installation, the library files can then be included into the target application.

The following topics are covered in this chapter:

- [Installation Procedure](#)
- [AGC Library Files](#)

2.1 INSTALLATION PROCEDURE

To install the library, follow these steps:

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

The installation creates the folder, `AGC v2.0`, which contains the files described in [2.2 "AGC Library Files"](#).

2.2 AGC LIBRARY FILES

The AGC library installer creates a directory, AGC v2.0. This directory contains these folders:

- demo
- docs
- h
- lib
- wavefiles

2.2.1 demo Folder

The demo folder contains files that are required by the AGC library quick start demonstration. This folder contains these subfolders:

- h
- libs
- src

[Table 2-1](#) describes the files in these subfolders.

TABLE 2-1: DEMONSTRATION FILES

File Name	Description
dsPIC33F AGC demo.hex	Demonstration hex file for dsPIC33F.
dsPIC33E AGC demo.hex	Demonstration hex file for dsPIC33E.
dsPIC33F AGC demo.mcp	Demonstration MPLAB [®] project file for dsPIC33F.
dsPIC33E AGC demo.mcp	Demonstration MPLAB project file for dsPIC33E.
cleanup.bat	Batch file script to clean intermediate build files.
h\dsPICDEM1_1Plus.h	C header file containing dsPICDEM [™] 1.1 Plus Development Board routines.
h\MEB.h	C header file containing Multimedia Expansion Board (MEB) routines.
h\lcd.h	C header file defining the interface to the LCD driver.
h\agc_api.h	C header file defining the interface to the AGC library.
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\agclibv2_33F.a	AGC library archive file for dsPIC33F.
libs\agclibv2_33E.a	AGC library archive file for dsPIC33E.
src\dsPICDEM1_1Plus.c	C source file containing the routines for the dsPICDEM 1.1 Plus Development Board.
src\MEB.c	C source file containing the routines for the MEB.
src\lcd.s	Assembly source code for communicating with the LCD controller.
src\main.c	C source file containing the main speech processing routine.
src\lcd_strings.c	C source file for LCD display.
src\Si3000Drv.c	C source file containing the implementation of the Si3000 codec driver.
src\WM8731Codecdrv.c	C source file containing the implementation of the WM8731 codec driver.

2.2.2 docs Folder

The `docs` folder contains the user's guide for the AGC 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

The `h` folder contains an include file for the AGC library, as listed in [Table 2-2](#).

TABLE 2-2: INCLUDE FILE

File Name	Description
<code>agc_api.h</code>	Include file that contains the interface to the AGC library. This file must be included in the application to use the AGC library.

2.2.4 lib Folder

The `lib` folder contains the library archive files for the AGC library. [Table 2-3](#) describes the files in this folder. The archive names are suffixed with the names of the target device families, `dsPIC33F` or `dsPIC33E`.

TABLE 2-3: LIBRARY FILE

File Name	Description
<code>agclibv2_33F.a</code>	AGC library archive file for <code>dsPIC33F</code> . This file must be included in the application in order to use the AGC library.
<code>agclibv2_33E.a</code>	AGC library archive file for <code>dsPIC33E</code> . This file must be included in the application in order to use the AGC library.

2.2.5 wavfiles Folder

The `wavfiles` folder contains two `.wav` files, which can be used with the quick start demonstration. The `.wav` files can be played back on the PC using a media player with the repeat function ON, to make the `.wav` file run continuously. [Table 2-4](#) describes the files in this folder.

TABLE 2-4: wavfiles

File Name	Description
<code>microchip_sas_fem_8K.wav</code>	A <code>.wav</code> file of a female voice sampled at 8 kHz.
<code>microchip_sas_male_8K.wav</code>	A <code>.wav</code> file of a male voice sampled at 8 kHz.

NOTES:



dsPIC[®] DSC AUTOMATIC GAIN CONTROL (AGC) LIBRARY USER'S GUIDE

Chapter 3. Quick Start Demonstration

This chapter describes the AGC library quick start demonstration for the dsPIC33F and dsPIC33E device families.

3.1 QUICK START DEMONSTRATION FOR THE 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

A demonstration application program included with the AGC library demonstrates the functionality of the library on the dsPIC33F DSC. In the demonstration setup (see [Figure 3-1](#)), the dsPICDEM[™] 1.1 Plus Development Board is configured to receive an 8 kHz audio signal from the PC through its microphone input port. The AGC library algorithm will process the signal and will output the signal through the speaker output port. The on-board Si3000 codec is used as the microphone and speaker interface.

A PC is used to drive sample audio signals through an audio cable from the PC's Speaker Out port to J16 (MIC IN) on the dsPICDEM 1.1 Plus Development Board. A headset or speaker is connected to the J17 (SPKR OUT) on the dsPICDEM 1.1 Plus Development Board. Changing the level (within a significant range) of the audio signal from the PC will not change the output level of the signal heard on the headset/speaker. This is due to the AGC library controlling the level of the signal.

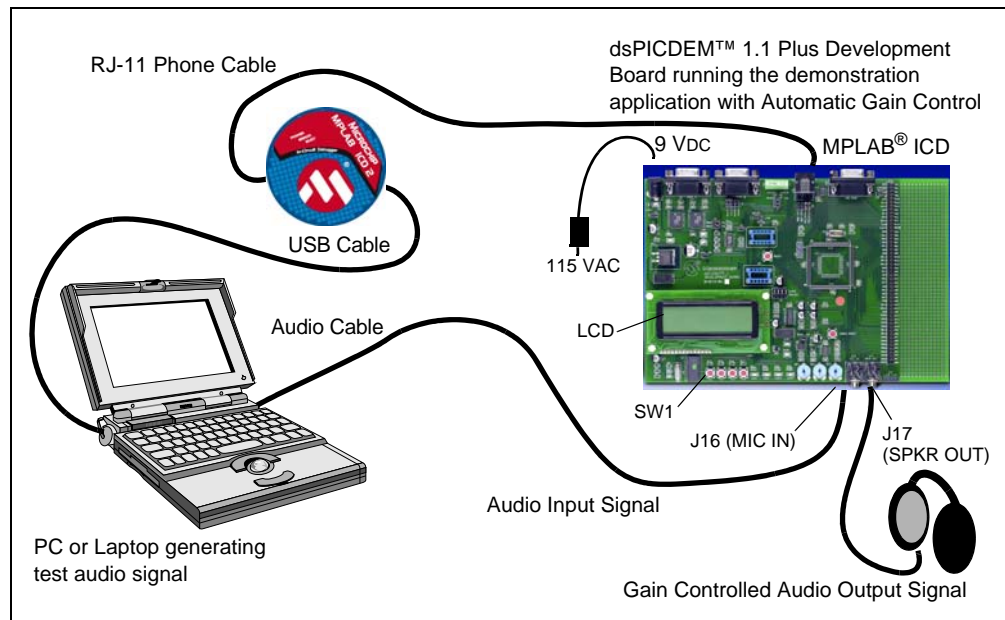
You can use the `.wav` files provided with the demonstration (in the `wavefiles` folder of the installation directory) as 8 kHz sampled speech signals, or you can provide your own signals. The input signal is captured by the on-board Si3000 voice band codec and the Data Converter Interface (DCI) module on the dsPIC DSC device. The dsPIC DSC device then outputs the gain-controlled signal through the device's DCI module and the on-board Si3000 codec.

When started, the program initializes with automatic gain control turned OFF, indicated by LED1 switched OFF, and OFF displayed on the LCD. With automatic gain control OFF, the level of the signal heard in the headset changes if the input level is changed from the PC.

Automatic gain control is enabled by pressing the switch, SW1. LED1 is now switched ON, and ON is displayed on the LCD. The speech signal heard on the headset is now gain controlled. Changing the level of the input signal will not change the level of the output. If the input signal level is below a peak level threshold, the gain is set to unity. If the input signal level is high and above the clip threshold, LED2 switches ON.

Turn on the repeat function in your PC's media player, to allow the `.wav` file to run continuously. Then, change the level in the media player to evaluate the performance of the AGC library.

FIGURE 3-1: SETUP FOR dsPIC33F AGC LIBRARY DEMO



Note: Some media players insert a break before each repeat of the .wav file. To avoid this, a sound editor program such as Audacity, can provide for continuous looping. Audacity, which is a free, cross-platform sound editor, is available from <http://audacity.sourceforge.net/>.

3.1.2 Demonstration Setup

The demonstration application is 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.

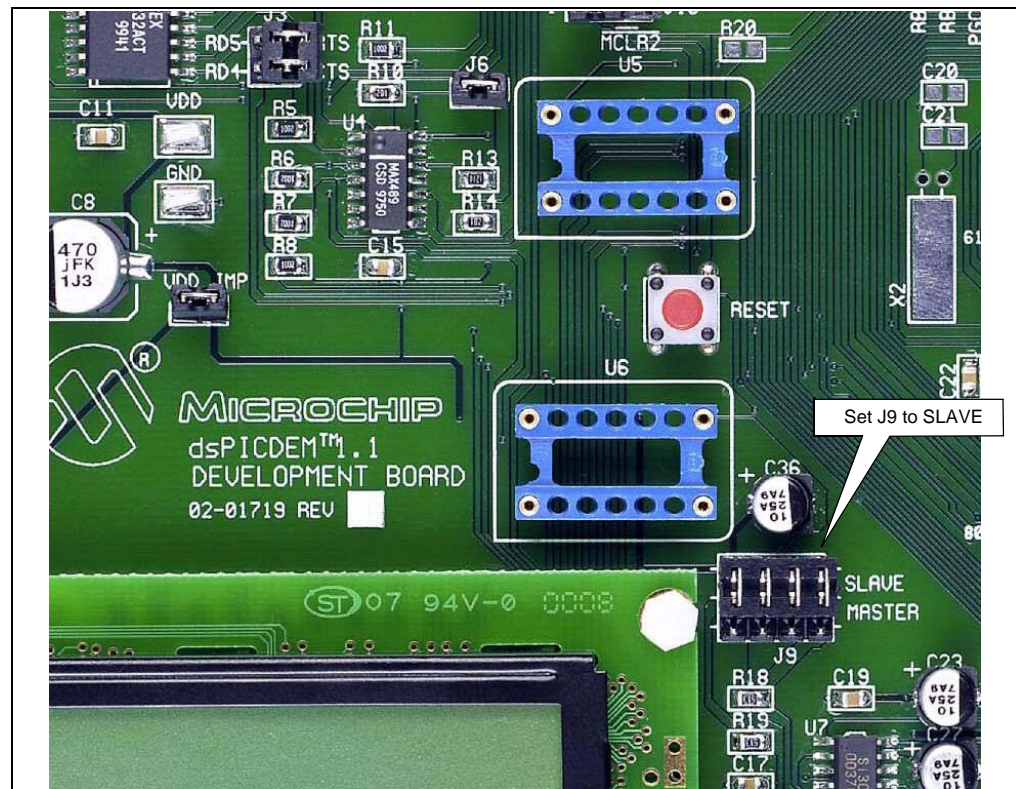
3.1.2.1 CONFIGURE THE dsPICDEM 1.1 PLUS DEVELOPMENT BOARD

Before applying power, you need to configure the board:

1. Set jumper J9 (adjacent to the oscillator socket) to the **SLAVE** position. This setting allows the on-board Si3000 codec chip to function as a serial clock slave.
2. Connect the audio cable between the Speaker Out port on the PC and the MIC IN jack (J16) on the dsPICDEM 1.1 Plus Development Board.
3. Connect the headset or speaker to the SPKR OUT jack (J17).
4. Connect the MPLAB ICD 3 between the PC (USB cable) and dsPICDEM 1.1 Plus Development Board (RJ-11 phone cable).
5. Connect the 9V power supply to power-up the dsPICDEM 1.1 Plus Development Board.

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

FIGURE 3-2: DEVELOPMENT BOARD SETUP



3.1.2.2 PROGRAM THE dsPIC DSC DEVICE

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

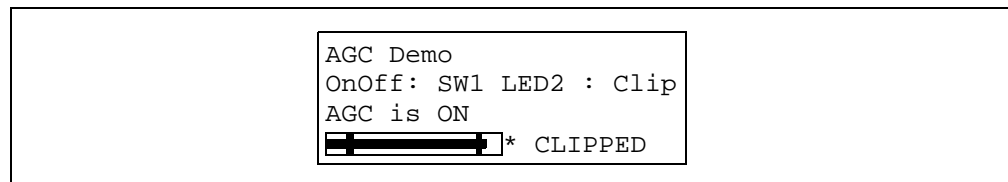
1. On your PC, launch MPLAB IDE and open the `dsPIC33F AGC demo.mcp` project located in the `demo` folder. For more information on using MPLAB IDE, refer to the "MPLAB[®] IDE, Simulator, Editor User's Guide" (DS51025).
2. Select *File > Import > dsPIC33F AGC demo.hex* to import the project hex file.
3. Select *Programmer > Connect* to link to the MPLAB ICD 3. The Output window confirms that the MPLAB ICD 3 is ready.
4. Connect the MPLAB ICD 3 to the dsPICDEM 1.1 Plus Development Board.
5. Program the dsPIC DSC device on the board.
6. Select *Programmer > Program*. The Output window displays the download progress and indicates that the programming has succeeded.
7. When the program is loaded, disconnect the MPLAB ICD 3 from the board.

3.1.3 Demonstration Procedure

With the demonstration application programmed into the device, the demonstration is ready to run. You can use the provided `.wav` files as input speech signals, which are located in the `wavfiles` folder of the installation directory, or you can provide your own signals. The input signal is sampled through the on-board Si3000 voice band codec and the DCI module of the dsPIC DSC device. The dsPIC DSC device then outputs the processed (gain controlled) signal through the device's DCI module and the on-board Si3000 codec.

The demonstration application relays the state of operation through the LED and the LCD. While the application is loading and initializing the on-chip and off-chip peripherals, a boot screen appears, which then switches automatically to the run-time screen as shown in [Figure 3-3](#).

FIGURE 3-3: DEMONSTRATION RUN-TIME LCD SCREEN



The run-time screen displays the following:

1. The name of the demonstration.
2. The switch SW1, which is used to switch AGC ON and OFF. LED2 serves as a CLIP indicator.
3. The current state of the algorithm.
4. A volume unit (VU) meter showing the input level. The bands shows an acceptable input range. The word CLIPPED is displayed when the input signal is too large.

When started, the program initializes with AGC turned OFF, indicated by LED1 turned OFF, and OFF displayed on the LCD. With the AGC OFF, the level of the signal heard on the headset changes with the level change of the input signal. This can be observed by changing the input level.

The AGC is enabled by pressing switch SW1. LED1 is now switched ON, and ON is displayed on the LCD. The level of the signal in the headset stays fairly constant as the AGC attempts to maintain the specified level of the input signal. This can be observed by changing the input level.

If the input signal amplitude is large and above the clip threshold, LED2 will switch ON until the clipping condition persists.

3.1.4 Demonstration Code Description

The demonstration code runs on a dsPIC33F device, using the Primary Oscillator as the clock source with the PLL configured for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This file allocates all of 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 AGC library functions.

The main function calls the `AGC_init()` function from the AGC library, which initializes the AGC 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. Only one transmit slot and one receive slot are used in this demonstration.

The `SI3000_open()` function also 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 the RF6 low for 100 cycles and then bringing it high. The codec is configured for a sample rate of 8 kHz. The Microphone 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.

The codec driver is polled for a full frame of data. When the codec driver indicates a full frame of data is available, the contents of the codec data buffers are copied into the `sin` array and the `AGC_apply()` function from the AGC library is called with `sin` as the input data frame. The `sin` data buffer, which is also the output of the `AGC_apply()` function after it has been executed, is output on the speaker of the headset.

The output on the LCD is made possible by initialization of the Serial Peripheral Interface (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.

The AGC library in the demonstration is configured to the following default settings:

- Target amplitude = 50% of full scale
- Clip threshold = 97%
- AGC headroom = 75%

To toggle the AGC ON and OFF, switch SW1 is polled. In the main loop, the value of `agcEnable` is read and passed to `AGC_apply` as the enable flag. If `agcEnable` is '0', the AGC library is still called, but the input/output buffer is not changed. This enables the AGC library to track the level changes in the signal. The `AGC_detectInputClip()` function is called after the `AGC_apply()` function to detect if the input signal has crossed the clip level. A clip condition will switch on LED2.

3.2 QUICK START DEMONSTRATION FOR THE dsPIC33E DEVICE FAMILY

The following topics are covered in this section:

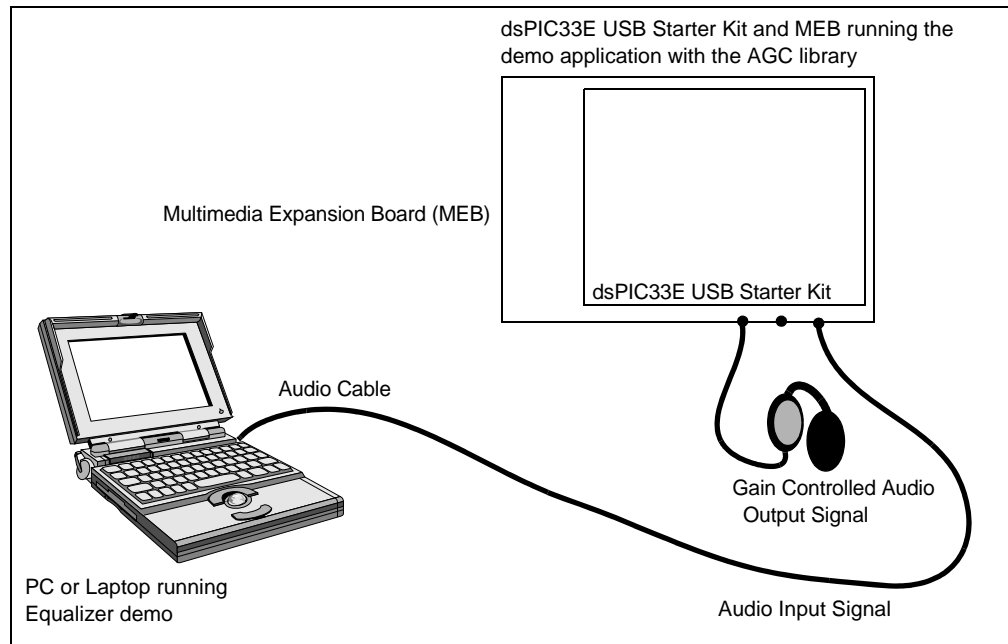
- [Demonstration Summary](#)
- [Demonstration Setup](#)
- [Demonstration Procedure](#)
- [Demonstration Code Description](#)

3.2.1 Demonstration Summary

A demonstration application program included with the AGC library demonstrates the functionality of the library on the dsPIC33E DSC. In the demonstration setup (illustrated in [Figure 3-4](#)), the Multimedia Expansion Board (MEB) in conjunction with a dsPIC33E USB Starter Kit is configured to receive an 8 kHz audio signal from the PC through its microphone input port. The AGC library algorithm will process the signal and will output the signal through the speaker output port. The on-board WM8731 codec is used as the microphone and speaker interface.

A PC is used to drive sample audio signals through an audio cable from the PC's Speaker Out port to the microphone input of the MEB. A headset or speaker is connected to the headphone output of the MEB. Changing the level (within a significant range) of the audio signal from the PC will not change the output level of the signal heard on the headset or speaker. This is due to the AGC library controlling the level of the signal.

FIGURE 3-4: SETUP FOR dsPIC33E AGC LIBRARY DEMO



You can use the .wav files provided with the demonstration (in the *wavefiles* folder of the installation directory) as 8 kHz sampled speech signals, or you can provide your own signals. The input signal is captured by the on-board WM8731 audio codec and the Data Converter Interface (DCI) module on the dsPIC DSC device. The dsPIC DSC device then outputs the gain-controlled signal through the device's DCI module and the on-board WM8731 codec.

When started, the program initializes with automatic gain control turned OFF, indicated by LED1 switched OFF on the MEB. With automatic gain control OFF, the level of the signal heard in the headset changes if the input level is changed from the PC.

Quick Start Demonstration

Automatic gain control is enabled by pressing switch S1 on the MEB. LED1 is switched ON and the speech signal heard on the headset is gain controlled. Changing the level of the input signal will not change the level of the output. If the input signal level is below a peak level threshold, the gain is set to unity. If the input signal level is high and above the clip threshold, LED2 switches ON.

Turn on the repeat function in your PC's media player, to allow the .wav file to run continuously. Then, change the level in the media player to evaluate the performance of the AGC library.

Note: Some media players insert a break before each repeat of the .wav file. To avoid this, a sound editor program, such as Audacity, can provide for continuous looping. Audacity, which is a free, cross-platform sound editor, is available from <http://audacity.sourceforge.net/>.

3.2.2 Demonstration Setup

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

3.2.2.1 CONFIGURE THE 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 the MEB.
2. Connect the audio cable between the Speaker Out port on the PC and the microphone jack (J7) on the MEB.
3. Connect a headset or speaker to the headphone jack (J8) of the MEB.
4. Connect the dsPIC33E USB Starter Kit to a PC using the USB A-to-mini-B cable provided with the Starter Kit.

3.2.2.2 PROGRAM THE dsPIC DSC DEVICE

Use the following process to load the AGC library demonstration into the dsPIC DSC device:

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

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

3.2.3 Demonstration Procedure

With the demonstration application programmed into the device, the demonstration is ready to run. You can use the provided `.wav` files, which are located in the `wavefiles` folder of the installation directory input speech signals, or you can provide your own signals. The input signal is sampled through the on-board WM8731 audio codec and the DCI module of the dsPIC DSC device. The dsPIC DSC device then outputs the processed (gain controlled) signal through the device's DCI module and the on-board WM8731 codec.

The demonstration application relays the state of operation through the LEDs. When started, the program initializes with AGC turned OFF, indicated by LED1 switching OFF on the MEB. With the AGC OFF, the level of the signal heard on the headset changes with level change in the input signal. This can be observed by changing the input level.

The AGC is enabled by pressing switch S1 on the LCD side of the MEB. LED1 is now switched ON. The level of the signal in the headset stays fairly constant as the AGC attempts to maintain the specified level of the input signal. This can be observed by changing the input level.

If the input signal amplitude is large and above the clip threshold, LED2 on the MEB will switch ON until the clipping condition persists.

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 configured for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This file allocates all of 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 AGC library functions.

The main function calls the `AGC_init()` function from the AGC library, which initializes the AGC 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 (one transmit slot and one receive slot are used in this demonstration).

Subsequently, the `WM8731Start()` function is used to enable the DCI module and I²C module. The codec is configured for a sample rate of 8 kHz (note that the library itself supports sample rates up to 48 kHz).

The codec driver is polled for a full frame of data. When the codec driver indicates a full frame of data is available, the contents of the codec data buffers are copied into the `sin` array and the `AGC_apply()` function from the AGC library is called with `sin` as the input data frame. The `sin` data buffer, which is also the output of the `AGC_apply()` function after it has been executed, is played out on the speaker output of the headset.

The AGC library in the demonstration is configured to the following default settings:

- Target amplitude = 50% of full scale
- Clip threshold = 97%
- AGC headroom = 75%

To toggle AGC ON and OFF, switch S1 is polled. In the main loop, the value of `agcEnable` is read and passed to `AGC_apply` as the enable flag. If `agcEnable` is '0', the AGC library is still called, but the input/output buffer is not changed. This enables the AGC library to track the level changes in the signal. The `AGC_detectInputClip()` function is called after the `AGC_apply()` function to detect if the input signal has crossed the clip level. A clip condition will switch on LED2.



dsPIC® DSC AUTOMATIC GAIN CONTROL (AGC) LIBRARY USER'S GUIDE

Chapter 4. Application Programming Interface

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

The following topics are covered in this chapter:

- [Adding the AGC Library to an Application](#)
- [AGC Operation and Parameters](#)
- [External Gain Control](#)
- [Using the AGC Library](#)
- [Resource Requirements](#)
- [AGC Library API Functions](#)
- [Application Tips](#)

4.1 ADDING THE AGC LIBRARY TO AN APPLICATION

To use the AGC library in an application, the library archive must be added to the application project workspace.

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

1. In the MPLAB workspace, right-click **Library Files** in the Project Window and select **Add Files**.
2. Browse to the location of either the `agclibv2_33F.a` file or the `agclibv2_33E.a` file, both of which are available in the `lib` folder in the installation directory.
3. Select the desired file, and then click **Open**.
4. The library is now added to the application.

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

4.2 AGC OPERATION AND PARAMETERS

This section explains the operation of the AGC algorithm. The parameters that will be required while using the library are also explained.

4.2.1 AGC Operation

The AGC algorithm operates by monitoring the maximum amplitude of a 10 ms speech frame. If the frame is classified as a part of a speech burst, the level is compared with the desired amplitude, which is called the amplitude target. If the amplitude of the frame needs to be increased, the overall amplitude of the signal is increased at a specified rate, which is called the Attack Rate. If the amplitude of the frame needs to be decreased, the overall amplitude of the signal is decreased at a specified rate, which is called the Release Rate.

If the amplitude is above the headroom level, this implies that the amplitude is nearing full scale and needs to be decreased at a more rapid rate, which is known as the recovery rate. If the frame is classified as a part of a silence frame (not a part of a speech burst), the signal level is decreased by the leakage factor rate, which is optional.

4.2.2 AGC Algorithm Parameters

The operation of the AGC algorithm is controlled by changing its parameters. These parameters are described below:

- **Amplitude Target:** This parameter defines the target amplitude that the input signal level is compared against in order to compute the gain. The AGC will try to maintain the average output amplitude at this level. This parameter is set using the `AGC_setAmplitudeTarget()` function.
- **Input Clip Threshold:** This parameter identifies the amplitude above which the input signal is defined to be saturated or clipped. The AGC algorithm provides an indication of this condition. The clipping condition by itself does not affect the operation of the AGC algorithm. This parameter is set by using the `AGC_setInputClipThreshold()` function. The `AGC_detectInputClip()` function can be used to determine whether the input signal has clipped.
- **Peak Detect Ratio:** This ratio is used by the AGC algorithm to determine whether the current frame is to be treated as a part of the speech burst or as silence. The lowest amplitude in the AGC History Buffer determines the silence level. If the silence frame and the ratio of the maximum amplitude of the current frame is smaller than the silence level by the Peak Detect factor, the frame is treated as if it were speech. Using a low value will make the algorithm less sensitive and slow to respond to signal changes from silence to speech or during inter-syllable durations. Choosing a higher value will cause the AGC algorithm to respond too quickly, giving all syllables a uniformly high stress (as if the person speaking is shouting). This parameter is set by using the `AGC_setPeakDetect()` function.
- **Maximum Gain:** This parameter places an upper limit on the gain that the AGC algorithm can apply to the input signal. This parameter is set by using the `AGC_setMaxGain()` function.
- **Minimum Gain:** This parameter places a lower limit on the gain that the AGC algorithm can apply to the input. If the input to the AGC is above the headroom level, then minimum gain level will be applied to it. This parameter is set by using the `AGC_setMinGain()` function.

- **Headroom:** The AGC algorithm can be directed to reduce the gain, if the predicted output rises above a critical level. This level is specified by the headroom. The headroom is always more than the Amplitude Target, but less than the Input Clip Threshold. The headroom is set by using the `AGC_setHeadroom()` function.
- **Attack Rate:** The attack rate determines the rate at which the gain is increased, if the current input amplitude is less than the set amplitude target. The attack rate is set using the `AGC_setAttack()` function.
- **Release Rate:** The release rate determines the rate at which the gain is decreased when the current output amplitude is greater than the set amplitude target. The release rate is set using the `AGC_setRelease()` function.
- **Leakage Rate:** The leakage rate determines the rate at which the gain is decreased, if the current frame is detected as silence (or close to silence) frame. That is, if the maximum amplitude is less than the current silence level by the peak detect factor, the gain is decreased at leakage rate. In an extended silence, the gain will gradually decrease to Minimum Gain. If it is desired to maintain a constant gain level during silences, this parameter should be set to zero. The leakage rate is set using the `AGC_setLeakageFactor()` function.
- **Recovery Rate:** The recovery rate determines the rate at which the gain is decreased, if the current output amplitude is greater than the headroom level. Typically, if the signal is above the headroom level, this indicates the signal is headed towards a clip condition. The attack rate is set using the `AGC_setRecovery()` function.

4.3 EXTERNAL GAIN CONTROL

The AGC library provides functions that can be used by the application to control the gain of an external codec or a Programmable Gain Amplifier, which are useful in applications that feature such components. The `AGC_getDesiredGain()` function provides the gain (in dB) required to maintain the signal level at the set Amplitude Target. This gain can then be provided to an external codec or a Programmable Gain Amplifier, which can be used to control the signal level.

4.4 USING THE AGC LIBRARY

The AGC library can process many independent channels of audio with each channel having its own settings and parameters. The application must include the `agclibv2_33F.a` or `agclibv2_33E.a`, and the `agc_api.h` files. The minimum coding steps required to use the AGC library are:

1. Select the sampling rate. Specify the sampling rate of the input signal by specifying the value of `AGC_PROC_FRAME` macro in `agc_api.h`.
Steps 2 through 5 need to be coded in the application. Refer to [Example 4-1](#) for the actual code.
2. Allocate memory for the AGC State memory. This is an integer array for size `AGC_XSTATE_MEM_SIZE_INT`. Every audio channel must have its own state memory.
3. Initialize the AGC State memory. Use the `AGC_init()` function to initialize the AGC State memory for every audio channel.
4. Set the amplitude target. Set the amplitude target for AGC output for each channel by using the `AGC_setAmplitudeTarget()` function. Note that if this step is skipped, the AGC will set the amplitude target to default (that is, 50%).
5. Apply AGC to the audio channel. Apply AGC to the audio channel by calling the `AGC_apply()` function. [Example 4-1](#) provides the code for calling this function.

EXAMPLE 4-1: CALLING THE `AGC_apply()` FUNCTION

```
#include "agc_api.h"

int agcStateX1[AGC_XSTATE_MEM_SIZE_INT];           /* Step 2 */
int agcStateX2[AGC_XSTATE_MEM_SIZE_INT];           /* Step 2 */

int main(void)
{
    AGC_init(agcStateX1);                           /* Step 3 */
    AGC_init(agcStateX2);                           /* Step 3 */

    AGC_setAmplitudeTarget(agcStateX1,Q15(0.65f));  /* Step 4 */
    AGC_setAmplitudeTarget(agcStateX2,Q15(0.60f));  /* Step 4 */
    .
    .
    .

    while(1)
    {
        AGC_apply(agcStateX1,input1,AGC_TRUE,AGC_PROC_FRAME); /* Step 5 */
        AGC_apply(agcStateX2,input2,AGC_TRUE,AGC_PROC_FRAME); /* Step 5 */
    }
}
```

Some of the AGC functions require input parameters to be specified as Q15 fractional value in hexadecimal format. The library provides a `Q15()` macro, which converts a floating point number to a Q15 fractional value in hexadecimal format.

Application Programming Interface

4.5 RESOURCE REQUIREMENTS

The AGC library requires the following resources while running on a dsPIC DSC device.

4.5.1 Program Memory, Data Memory and MIPS Usage

TABLE 4-1: PROGRAM MEMORY USAGE

	Size (bytes)	Section
Code in Program Memory	1146 (dsPIC33F) 1404 (dsPIC33E)	.libagc
Tables in Program Memory	306 (dsPIC33F) 0 (dsPIC33E)	.const
Total Program Memory	1452 (dsPIC33F) 1404 (dsPIC33E)	—

TABLE 4-2: DATA MEMORY USAGE (8 kHz SAMPLING RATE)

	Size (bytes)	Alignment	Section
agc_state_mem_x	596	2	X data memory
sin	160	2	X data memory
Tables in Data Memory	0 (dsPIC33F) 204 (dsPIC33E)	2	X data memory
Total Data Memory	756 (dsPIC33F) 960 (dsPIC33E)	—	—

The size of `agc_state_mem_x` depends on the lookback `AGC_AMP_HIST`, and the size of `sin` (Send in), assuming the default input block length of 10 ms, is proportional to the sampling rate. Sampling rates up to 48 kHz are supported.

TABLE 4-3: ESTIMATED DYNAMIC MEMORY USAGE

Section	Size (bytes)
Heap	0
Stack	< 320

TABLE 4-4: COMPUTATIONAL SPEED

Function	MIPS	Typical Call Frequency
AGC_init()	< 0.5	Once
AGC_apply()	0.5 (8 kHz) to 1.5 (48 kHz)	10 ms
All other functions	Minimal	As required

4.5.2 Data Format

`sin` can be 10-bit, 12-bit or 16-bit linear Pulse Code Modulation (PCM) data. The AGC algorithm automatically adjusts for the data format used.

4.6 AGC LIBRARY API FUNCTIONS

This section provides detailed information for the following API functions:

- `AGC_init`
- `AGC_apply`
- `AGC_detectInputClip`
- `AGC_getDesiredGain`
- `AGC_setInputClipThreshold`
- `AGC_getInputClipThreshold`
- `AGC_setPeakDetect`
- `AGC_getPeakDetect`
- `AGC_setMaxGain`
- `AGC_getMaxGain`
- `AGC_setMinGain`
- `AGC_getMinGain`
- `AGC_setAmplitudeTarget`
- `AGC_getAmplitudeTarget`
- `AGC_setHeadroom`
- `AGC_getHeadroom`
- `AGC_setAttack`
- `AGC_getAttack`
- `AGC_setRelease`
- `AGC_getRelease`
- `AGC_setLeakageFactor`
- `AGC_getLeakageFactor`
- `AGC_setRecovery`
- `AGC_getRecovery`
- `AGC_TRUE`
- `AGC_FALSE`
- `AGC_SAMPLE_RATE`
- `AGC_PROC_FRAME`
- `AGC_AMP_HIST`
- `AGC_XSTATE_MEM_SIZE_INT`
- `AGC_CLIP_THRES_DEFAULT`
- `AGC_PEAK_DETECT_DEFAULT`
- `AGC_MAX_GAIN_DEFAULT`
- `AGC_MIN_GAIN_DEFAULT`
- `AGC_INIT_GAIN_DEFAULT`
- `AGC_AMPLITUDE_TARGET_DEFAULT`
- `AGC_HEADROOM_DEFAULT`
- `AGC_ATTACK_DEFAULT`
- `AGC_RELEASE_DEFAULT`
- `AGC_LEAKAGE_FACTOR_DEFAULT`
- `AGC_RECOVERY_DEFAULT`

AGC_init

Description

Initializes the AGC algorithm.

Include

agc_api.h

Prototype

```
void AGCinit(int* ptrStateX);
```

Arguments

ptrStateX A pointer to the X memory for this instance of AGC.

Return Value

None.

Remarks

None.

Code Example

```
int agc_state_mem_x     [AGC_XSTATE_MEM_SIZE_INT]   _XBSS(2);  
...  
AGC_init(agc_state_mem_x);
```

AGC_apply

Description:

This function applies AGC to the current frame of data. It also calculates the desired AGC gain (in dB) which is available for use by suitable hardware (if software gain application is disabled, the program continues to run in the background). It also updates the amplitude clip detect flag.

Include

```
agc_api.h
```

Prototype

```
void AGC_apply(int* ptrStateX, int* Sin, int enable, int agcProcSize);
```

Arguments

<code>ptrStateX</code>	A pointer to the X memory for this instance of AGC.
<code>Sin</code>	A pointer to the input/output buffer of size <code>AGC_PROC_FRAME</code> .
<code>Enable</code>	A flag to indicate if AGC is required for this buffer (<code>AGC_TRUE/AGC_FALSE</code>).
<code>agcProcSize</code>	The length of the input/output buffer <code>Sin</code> .

Return Value

None.

Remarks

The AGC algorithm is process-in-place meaning that the output is passed back in the input buffer. Setting `Enable` to `AGC_FALSE` returns an unprocessed buffer of data, but the AGC algorithm still runs in the background and all state variables are updated.

Code Example

```
int agc_state_mem_x    [AGCG_XSTATE_MEM_SIZE_INT]    _XBSS(2);
int Sin                [AGC_PROC_FRAME]             _XBSS(2);
...
AGC_init(agc_state_mem_x);
...
AGC_apply(agc_state_mem_x, Sin, AGC_TRUE, AGC_PROC_FRAME);
```

AGC_detectInputClip

Description

Returns a flag to indicate whether input audio is close to clipping.

Include

```
agc_api.h
```

Prototype

```
int AGC_detectInputClip(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Values

`AGC_TRUE` The current input frame is above the clip threshold.

`AGC_FALSE` The current input frame is below the clip threshold.

Remarks

Clip threshold parameter can be selected by the user (see function `AGC_setInputClipThreshold`).

Code Example

```
int clipDetected;  
...  
clipDetected = AGC_detectInputClip(ptrStateX);
```

AGC_getDesiredGain

Description

Returns the gain that an external codec could apply to the signal to achieve the target amplitude.

Include

```
agc_api.h
```

Prototype

```
int AGC_getDesiredGain(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit positive integer representing gain in dB.

Remarks

This output is available whether or not AGC is actually applied: for instance, it could be used as input to an external gain control (for example, on the audio codec).

Code Example

```
int gainDesired;  
...  
gainDesired = AGC_getDesiredGain(ptrStateX);
```

AGC_setInputClipThreshold

Description

Sets the input clip threshold.

Include

agc_api.h

Prototype

```
void AGC_setInputClipThreshold(int* ptrStateX, int value);
```

Arguments

ptrStateX	A pointer to the X memory for this instance of AGC.
value	A 16-bit positive integer representing the threshold as a fraction.

Return Value

None.

Remarks

None.

Code Example

```
AGC_setInputClipThreshold(ptrStateX, Q15(0.97f));
```

This sets the threshold to 97% of full scale.

AGC_getInputClipThreshold

Description

This function will get the input clip threshold for the specified instance of AGC.

Include:

```
agc_api.h
```

Prototype

```
int AGC_getInputClipThreshold(int* ptrStateX, int value);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit positive integer representing the parameter as a fraction.

Remarks

None.

Code Example

```
int clipLevel;  
  
clipLevel = AGC_getInputClipThreshold(ptrStateX);
```

AGC_setPeakDetect

Description

Sets the amplitude ratio for detecting input frame as a speech frame. Presence of speech frame inhibits AGC gain from leaking to initial gain.

Include

```
agc_api.h
```

Prototype

```
void AGC_setPeakDetect(int* ptrStateX, int value);
```

Arguments

<code>ptrStateX</code>	A pointer to the X memory for this instance of AGC.
<code>value</code>	A 16-bit positive integer representing the parameter as a fraction.

Return Value

None.

Remarks

None.

Code Example

```
AGC_setPeakDetect(ptrStateX, Q15(0.25F));
```

This sets the peak detect ratio to 25%.

AGC_getPeakDetect

Description

Returns the current value of the amplitude ratio for detecting input peak amplitudes.

Include

```
agc_api.h
```

Prototype

```
int AGC_getPeakDetect(int* ptrStateX, int value);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit positive integer representing the parameter as a fraction.

Remarks

None.

Code Example

```
int peakRatio;  
...  
peakRatio = AGC_getPeakDetect(ptrStateX);
```

AGC_setMaxGain

Description

Sets the maximum permitted value that the AGC can apply to the signal.

Include

agc_api.h

Prototype

```
void AGC_setMaxGain(int* ptrStateX, int value);
```

Arguments

ptrStateX	A pointer to the X memory for this instance of AGC.
value	A 16-bit positive integer representing a gain level in dB.

Return Value

None.

Remarks

None.

Code Example

```
AGC_setMaxGain(ptrStateX, 32);
```

This limits the output gain to no more than 32 dB.

AGC_getMaxGain

Description

Returns the maximum gain calculated that the AGC will apply to the signal.

Include

```
agc_api.h
```

Prototype

```
int AGC_getMaxGain(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

16-bit integer representing the maximum gain in dB.

Remarks

None.

Code Example

```
int gainLevel;  
  
gainLevel = AGC_getMaxGain(ptrStateX);
```

AGC_setMinGain

Description

Sets the minimum gain that the AGC will apply to the signal.

Include

```
agc_api.h
```

Prototype

```
void AGC_setMinGain(int* ptrStateX, int value);
```

Arguments

<code>ptrStateX</code>	A pointer to the X memory for this instance of AGC.
<code>value</code>	A 16-bit positive integer representing a gain level in dB.

Return Value

None.

Remarks

It is expected that this value will be unity in most instances, but a higher value may be required for a particular setup.

Code Example

```
AGC_setMinGain(ptrStateX, 2);
```

This prevents the output gain from falling below 2 dB.

AGC_getMinGain

Description

Returns the minimum gain that the AGC will apply to the signal.

Include

```
agc_api.h
```

Prototype

```
int AGC_getMinGain(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

16-bit integer representing the minimum gain in dB.

Remarks

None.

Code Example

```
int gainLevel;  
  
gainLevel = AGC_getMinGain(ptrStateX);
```

AGC_setAmplitudeTarget

Description

Sets the target peak amplitude for the output speech, as a fraction of full scale.

Include

```
agc_api.h
```

Prototype

```
void AGC_setAmplitudeTarget(int* ptrStateX, int value);
```

Arguments

<code>ptrStateX</code>	A pointer to the X memory for this instance of AGC.
<code>value</code>	A 16-bit positive integer representing the parameter as a fraction.

Return Value

None.

Remarks

The target amplitude is specified as Q15 number. Alternatively the Q15 macro could be used to convert a float value to a Q15 number.

Code Example

```
AGC_setAmplitudeTarget(ptrStateX, Q15(0.65f));
```

This sets target level as 65% of full scale.

AGC_getAmplitudeTarget

Description

Returns the current value of target amplitude.

Include

```
agc_api.h
```

Prototype

```
int AGC_getAmplitudeTarget(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit value representing the parameter as a fraction.

Remarks

None.

Code Example

```
int ampLevel;  
  
ampLevel = AGC_getAmplitudeTarget(ptrStateX);
```

AGC_setHeadroom

Description

Sets the AGC headroom as a fraction of full scale.

Include

agc_api.h

Prototype

```
void AGC_setHeadroom(int* ptrStateX, int value);
```

Arguments

ptrStateX	A pointer to the X memory for this instance of AGC.
value	A 16-bit positive integer representing the parameter as a fraction.

Return Value

None.

Remarks

It is expected that it will always be greater than the value chosen for AGC amplitude target.

Code Example

```
AGC_setHeadroom(ptrStateX, Q15(0.75f));
```

This sets target level as 75% of full scale.

AGC_getHeadroom

Description

Returns the current value of AGC Headroom level .

Include

```
agc_api.h
```

Prototype

```
int AGC_getHeadroom(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit value representing the parameter as a fraction.

Remarks

None.

Code Example

```
int headRoomValue;  
  
headRoomValue = AGC_getHeadroom(ptrStateX);
```

AGC_setAttack

Description

Sets the AGC attack smoothing factor (factor for increasing gain) as a fraction.

Include

agc_api.h

Prototype

```
void AGC_setAttack(int* ptrStateX, int value);
```

Arguments

ptrStateX	A pointer to the X memory for this instance of AGC.
value	A 16-bit positive integer representing the parameter as a fraction.

Return Value

None.

Remarks

None.

Code Example

```
AGC_setAttack(ptrStateX, Q15(0.05f));
```

This sets the attack rate at 5%.

AGC_getAttack

Description

Returns the current value of the AGC Attack smoothing factor.

Include

agc_api.h

Prototype

```
int AGC_getAttack(int* ptrStateX);
```

Arguments

ptrStateX A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit positive integer representing the parameter as a fraction.

Remarks

None.

Code Example

```
int attackValue;  
  
attackValue = AGC_getAttack(ptrStateX);
```

AGC_setRelease

Description

Sets the AGC release smoothing factor (for decreasing gain).

Include

agc_api.h

Prototype

```
void AGC_setRelease(int* ptrStateX, int value);
```

Arguments

ptrStateX	A pointer to the X memory for this instance of AGC.
value	A 16-bit positive integer representing the parameter as a fraction.

Return Value

None.

Remarks

None.

Code Example

```
AGC_setRelease(ptrStateX, Q15(0.02f));
```

This sets the release smoothing factor to 2%.

AGC_getRelease

Description

Returns the current value of AGC Release Smoothing factor.

Include

```
agc_api.h
```

Prototype

```
int AGC_getRelease(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit positive integer value representing the parameter as a fraction.

Remarks

None.

Code Example

```
int releaseValue;  
  
releaseValue = AGC_getRelease(ptrStateX);
```

AGC_setLeakageFactor

Description

Sets the AGC leakage smoothing factor (factor for decreasing gain when speech is not present).

Include

agc_api.h

Prototype

```
void AGC_setLeakageFactor(int* ptrStateX, int value);
```

Arguments

ptrStateX	A pointer to the X memory for this instance of AGC.
value	A 16-bit positive integer representing the parameter as a fraction.

Return Value

None.

Remarks

None.

Code Example

```
AGC_setLeakageFactor(ptrStateX, Q15(0.001f));
```

This sets the leakage factor to 0.1%.

AGC_getLeakageFactor

Description

Returns the current value of the AGC Leakage Smoothing Factor.

Include

```
agc_api.h
```

Prototype

```
int AGC_getLeakageFactor(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit positive integer representing the parameter as a fraction.

Remarks

None.

Code Example

```
int leakValue;  
  
leakValue = AGC_getRelease(ptrStateX);
```

AGC_setRecovery

Description

Sets the AGC Recover smoothing factor (rate at which the signal is decreased after climbing above clip threshold).

Include

agc_api.h

Prototype

```
void AGC_setRecovery(int* ptrStateX, int value);
```

Arguments

ptrStateX	A pointer to the X memory for this instance of AGC.
value	A 16-bit positive integer representing the parameter as a fraction.

Return Value

None.

Remarks

None.

Code Example

```
AGC_setRecovery(ptrStateX, Q15(0.1f));
```

Sets the recovery rate to 10%.

AGC_getRecovery

Description

Returns the current value of AGC Recovery smoothing factor.

Include

```
agc_api.h
```

Prototype

```
int AGC_getRecovery(int* ptrStateX);
```

Arguments

`ptrStateX` A pointer to the X memory for this instance of AGC.

Return Value

A 16-bit positive integer representing the parameter as a fraction.

Remarks

None.

Code Example

```
int recoValue;  
  
recoValue = AGC_getRecovery(ptrStateX);
```

AGC_TRUE

Description

Used to indicate true to the AGC algorithm.

Value

1

AGC_FALSE

Description:

Used to indicate false to the AGC algorithm.

Value:

0

AGC_SAMPLE_RATE

Description:

Defines the sample rate at which the AGC operates. This value can be changed to suit the application requirements.

Value:

8000 (default)

AGC_PROC_FRAME

Description:

Length of input/output audio buffer (shared with other audio functions).

Value:

80 at 8 kHz sampling rate; in general, equivalent to 10 ms.

AGC_AMP_HIST

Description

Number frames look-back for the AGC algorithm. Since this determines memory size it may not be changed at run-time.

Value

128

AGC_XSTATE_MEM_SIZE_INT

Description

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

Value

(2*AGC_AMP_HIST + 42)

AGC_CLIP_THRES_DEFAULT

Description

Threshold to detect input amplitude clipping (16-bit integer interpreted as Q15 fractional).

Value

Q15(0.97f)

AGC_PEAK_DETECT_DEFAULT

Description

Amplitude ratio used to detect possible speech (16-bit integer interpreted as Q15 fractional).

Value

Q15(0.3f)

AGC_MAX_GAIN_DEFAULT

Description

Maximum permitted amplification in dB applied by the AGC. It is a positive integer.

Value

45

AGC_MIN_GAIN_DEFAULT

Description

Minimum permitted amplification in dB applied by the AGC. It must not be negative.

Value

0

AGC_INIT_GAIN_DEFAULT

Description

Initial value of gain applied by AGC in dB: if it is known roughly how much gain is needed, this will speed the initial convergence of the AGC.

Value

0

AGC_AMPLITUDE_TARGET_DEFAULT

Description

Target amplitude for the AGC, as a fraction of full scale (16-bit integer interpreted as Q15 fractional). This is understood to be an average of speech peak levels – actual amplitudes may be much higher.

Value

Q15(0.50f)

AGC_HEADROOM_DEFAULT

Description

Amplitude threshold to prevent clipping, as a fraction of full scale (16-bit integer interpreted as Q15 fractional). Note that it must leave some room for overshoot.

Value

Q15(0.75f)

AGC_ATTACK_DEFAULT

Description

Smoothing ratio when an increase in gain is demanded (16-bit integer interpreted as Q15 fractional).

Value

Q15(0.02f)

AGC_RELEASE_DEFAULT

Description

Smoothing ratio when a decrease in gain is demanded (16-bit integer interpreted as Q15 fractional).

Value

Q15(0.001f)

AGC_LEAKAGE_FACTOR_DEFAULT

Description

Smoothing ratio to return gain to default level when speech is not detected (16-bit integer interpreted as Q15 fractional). Set to zero to disable the leakage feature.

Value

Q15(0.0001f)

AGC_RECOVERY_DEFAULT

Description

Smoothing ratio to respond to sudden increase in input speech level (16-bit integer interpreted as Q15 fractional).

Value

Q15(0.1f)

4.7 APPLICATION TIPS

The default values of the AGC parameters configure the AGC library to operate in a typical voice application. The following application tips provide information on ways to optimize the AGC algorithm for a given application:

1. The optimal headroom value is 75% of the full scale range. The optimal amplitude target value is 65% of the full scale range. Setting up the parameters in this way ensures good use of the available precision. The difference between the amplitude target and the headroom accommodates for overshoot.
2. Setting the peak detect parameter to a higher value will make the AGC algorithm overly sensitive, giving all syllables a uniformly high stress. A value that is too low causes the AGC to be slow to respond. A value of 0.3 allows for natural sounding speech.
3. The AGC algorithm is designed for speech signals and uses the statistical nature of speech signals as a basis for its operation. The algorithm will not process the input signals where syllables are exaggerated, or where the input signal contains musical sounds.
4. In typical applications, the minimum gain is set to 0 dB (or unity gain). It can be set to a non-zero positive value to provide a default gain to signal.
5. The clip detection function should be used to check whether the input signal is too large. This feature can prove useful as a protection mechanism for subsequent signal processing stages or for protecting output devices.
6. The amplitude target could be set to a value between 25% to 85% of full scale. A low value should be used if the input is expected to contain momentary increases in loudness.
7. For deterministic operation of the AGC algorithm, some guidelines need to be followed:
 - Amplitude Target < Headroom < 95% of Full Scale
 - 0.0 < Attack rate < Recovery Rate
 - 0.0 < Leakage Rate < Release rate < Recovery Rate
 - Attack rate < Recovery Rate < 0.5

NOTES:



dsPIC[®] DSC AUTOMATIC GAIN CONTROL (AGC) LIBRARY USER'S GUIDE

Index

A	
Adding the AGC Library to an Application.....	25
AGC Algorithm Parameters.....	26
Amplitude Target	26
Attack Rate	27
Headroom	27
Input Clip Threshold.....	26
Leakage Rate	27
Maximum Gain.....	26
Minimum Gain.....	26
Peak Detect Ratio.....	26
Recovery Rate.....	27
Release Rate	27
AGC Library	
Application Example	11
Features.....	12
Overview	11
Requirements	12
AGC Library Demonstration Setup.....	18, 19, 22, 23
AGC Library Files	
demo Folder.....	14
docs Folder	15
AGC Operation	26
API Functions	
AGC_AMP_HIST	55
AGC_AMPLITUDE_TARGET_DEFAULT.....	57
AGC_apply	32
AGC_ATTACK_DEFAULT.....	58
AGC_CLIP_THRES_DEFAULT.....	56
AGC_detectInputClip	33
AGC_FALSE.....	55
AGC_getAmplitudeTarget.....	44
AGC_getAttack	48
AGC_getDesiredGain	34
AGC_getHeadroom	46
AGC_getInputClipThreshold	36
AGC_getLeakageFactor	52
AGC_getMaxGain.....	40
AGC_getMinGain.....	42
AGC_getPeakDetect	38
AGC_getRecovery	54
AGC_getRelease	50
AGC_HEADROOM_DEFAULT.....	57
AGC_init	31
AGC_INIT_GAIN_DEFAULT	57
AGC_LEAKAGE_FACTOR_DEFAULT	58
AGC_MAX_GAIN_DEFAULT	56
AGC_MIN_GAIN_DEFAULT	57
AGC_PEAK_DETECT_DEFAULT.....	56
AGC_PROC_FRAME	55
AGC_RECOVERY_DEFAULT.....	58
AGC_RELEASE_DEFAULT	58
AGC_SAMPLE_RATE	55
AGC_setAmplitudeTarget	43
AGC_setAttack	47
AGC_setHeadroom.....	45
AGC_setInputClipThreshold	35
AGC_setLeakageFactor	51
AGC_setMaxGain	39
AGC_setMinGain	41
AGC_setPeakDetect.....	37
AGC_setRecovery	53
AGC_setRelease	49
AGC_TRUE	55
AGC_XSTATE_MEM_SIZE_INT	56
Application Tips	59
C	
Configuration	
dsPICDEM™ 1.1 Plus Development Board.....	19
MEB and dsPIC33E USB Starter Kit.....	23
Customer Notification Service.....	8
Customer Support	8
D	
Demonstration	
Board Configuration	19, 23
Code	21, 24
Procedure	20, 24
Development Board Setup	19
Documentation	
Conventions	6
Layout	5
E	
External Gain Control.....	27
M	
MPLAB IDE User's Guide	7
P	
Programming the dsPIC [®] DSC Device	20, 23
R	
Reading, Recommended	7
Resource Requirements	29
Computational Speed	29
Data Format	29
Data Memory Usage	29
Estimated Dynamic Memory Usage.....	29
Program Memory Usage.....	29

U

Using the AGC Library	27
Allocating Memory for the AGC State Memory .	27
Applying AGC to the Audio Channel.....	27
Initializing the AGC State Memory	27
Selecting the Sampling Rate.....	27
Setting the Amplitude Target	27

W

Warranty Registration.....	7
----------------------------	---

NOTES:



MICROCHIP

Worldwide Sales and Service

AMERICAS

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

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

EUROPE

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

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

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

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

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

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

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

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

05/02/11