dsPIC® DSC Automatic Gain Control (AGC) User's Guide

Table of Contents

NOTES:

Preface

NOTICE TO CUSTOMERS

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

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

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

INTRODUCTION

This 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 (API)"** – This chapter outlines how the API functions provided in the AGC library can be included in your application software via the Application Programming Interface.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	<u>*File>Save*</u>
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options]` *`file`* `[options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

WARRANTY REGISTRATION

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

RECOMMENDED READING

This user's guide describes how to use the AGC library. Other useful documents include:

dsPIC30F Family Reference Manual (DS70046)

Consult 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.

dsPIC30F Family Overview (DS70043)

This document provides an overview of the functionality of the dsPIC® product family. It helps determine how the dsPIC30F 16-bit Digital Signal Controller family fits a specific product application. This document is a supplement to the *dsPIC30F Family Reference Manual.*

dsPIC33F Family Reference Manual Sections

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

dsPIC30F/dsPIC33F 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® ASM30, MPLAB® LINK30 and Utilities User's Guide (DS51317)

This document details Microchip Technology's language tools for dsPIC devices based on GNU technology. The language tools discussed are:

- MPLAB ASM30 Assembler
- MPLAB LINK30 Linker
- MPLAB LIB30 Archiver/Librarian
- Other Utilities

MPLAB C30 C Compiler User's Guide (DS51284)

This document details the use of Microchip's MPLAB C30 C Compiler for dsPIC devices to develop an application. MPLAB C30 is a GNU-based language tool, based on source code from the Free Software Foundation (FSF). For more information about the FSF, see www.fsf.org.

Other GNU language tools available from Microchip are:

- MPLAB ASM30 Assembler
- MPLAB LINK30 Linker
- MPLAB LIB30 Librarian/Archiver

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

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

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

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

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

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

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

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

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB C18 and MPLAB C30 C compilers; MPASM™ and MPLAB ASM30 assemblers; MPLINK™ and MPLAB LINK30 object linkers; and MPLIB™ and MPLAB LIB30 object librarians.
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB ICE 2000, MPLAB ICE 4000, and MPLAB REAL ICE™.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 2.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 and PRO MATE® II device programmers and the PICSTART® Plus and PICkit® 1 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://support.microchip.com.

DOCUMENT REVISION HISTORY

Revision A (August 2009)

This is the initial released revision of this document.

Chapter 1. Introduction

The dsPIC® DSC Automatic Gain Control (AGC) Library provides an algorithm to automatically control the level of a signal. This chapter provides as overview of the library.

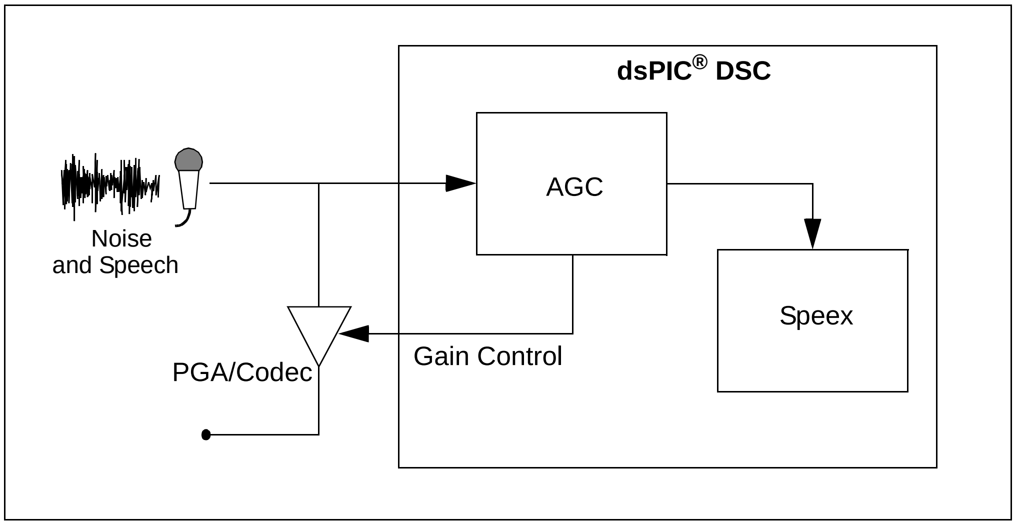
Topics covered include:

- AGC Library Overview
- Features
- Host System Requirements

1.1 AGC LIBRARY OVERVIEW

The AGC library is useful in speech and audio applications where distance between the system user and the microphone varies. The main function of the AGC algorithm is to estimate the short-term peak amplitude of input speech and apply a gain factor so that it is brought up to a desired level that has been 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 the system of which the AGC is a part has been correctly set up for normal use by someone 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 hook is provided for this. The library will also flag amplitude clipping on the input.

1.2 FEATURES

The AGC library features:

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

1.3 HOST SYSTEM REQUIREMENTS

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

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

Chapter 2. Installation

This chapter describes the installation procedure for the AGC library. To use the library, you must install it on your laptop or desktop PC. After installation, the library files can then be included into the target application. Topics covered include:

- Installation Procedure
- AGC Library Files

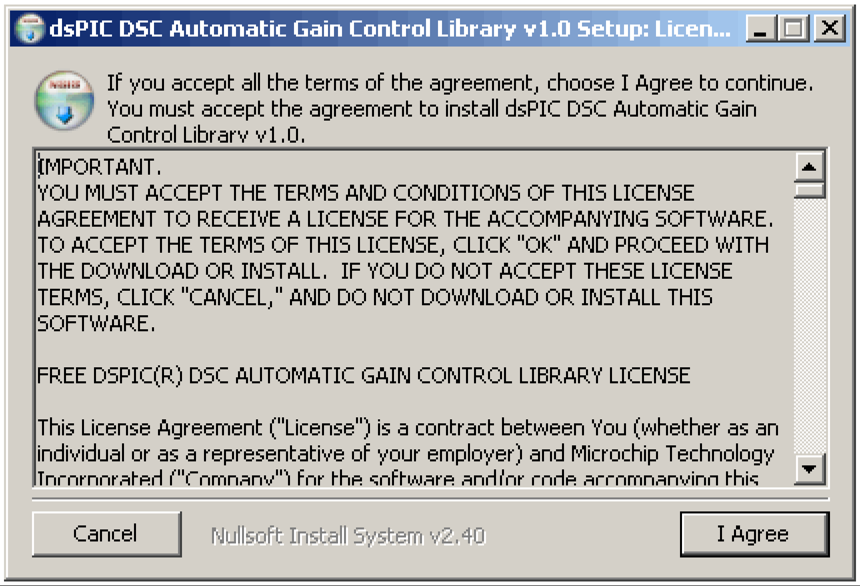
2.1 INSTALLATION PROCEDURE

The AGC library is available on a CD or as an archive file, which can be downloaded from the Microchip web site. In either case, installation of the library requires execution of the `setup.exe` file.

To install the library follow these steps:

1. Run the `setup.exe` file. This will start the installation and a license agreement dialog box will appear.

FIGURE 2-1: LIBRARY LICENSE AGREEMENT


dsPIC DSC Automatic Gain Control Library v1.0 Setup: Licen...
Cancel
Nullsoft Install System v2.40
I Agree

If you accept all the terms of the agreement, choose I Agree to continue. You must accept the agreement to install dsPIC DSC Automatic Gain Control Library v1.0.

IMPORTANT.
YOU MUST ACCEPT THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT TO RECEIVE A LICENSE FOR THE ACCOMPANYING SOFTWARE. TO ACCEPT THE TERMS OF THIS LICENSE, CLICK "OK" AND PROCEED WITH THE DOWNLOAD OR INSTALL. IF YOU DO NOT ACCEPT THESE LICENSE TERMS, CLICK "CANCEL," AND DO NOT DOWNLOAD OR INSTALL THIS SOFTWARE.

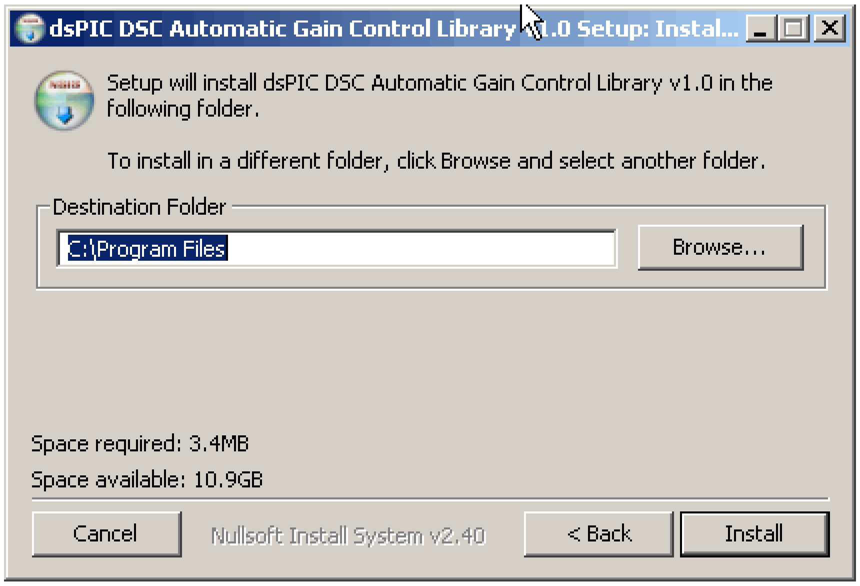
FREE DSPIC(R) DSC AUTOMATIC GAIN CONTROL LIBRARY LICENSE

This License Agreement ("License") is a contract between You (whether as an individual or as a representative of your employer) and Microchip Technology Incorporated ("Company") for the software and/or code accompanying this

2. Review the license agreement and then click **I Agree** to continue the installation.

3. When prompted, specify the destination folder for the library files and folders.

FIGURE 2-2: SPECIFY INSTALLATION FOLDER


dsPIC DSC Automatic Gain Control Library v1.0 Setup: Instal...
Destination Folder
Browse...
Cancel
Nullsoft Install System v2.40
< Back
Install

Setup will install dsPIC DSC Automatic Gain Control Library v1.0 in the following folder.

To install in a different folder, click Browse and select another folder.

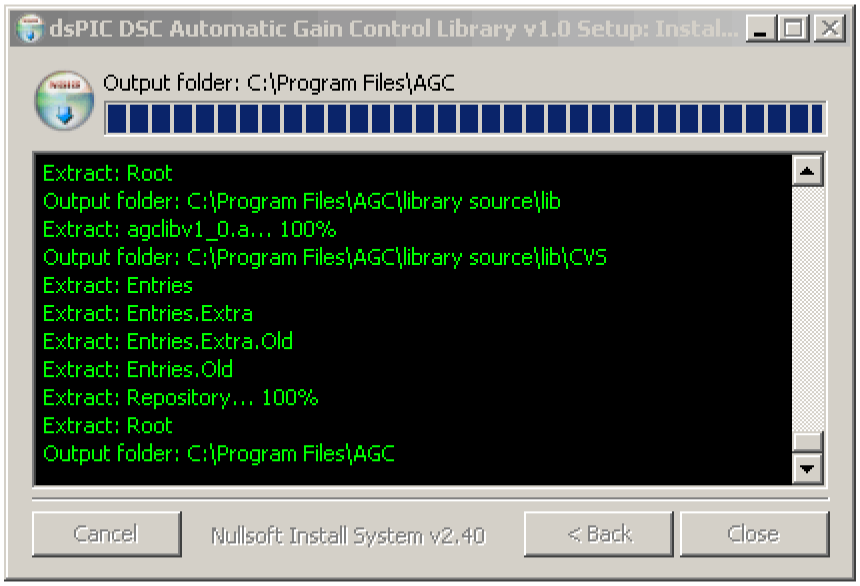
C:\Program Files

Space required: 3.4MB

Space available: 10.9GB

4. The installer will install the library files into the specified folder.

FIGURE 2-3: INSTALLATION IN PROGRESS


dsPIC DSC Automatic Gain Control Library v1.0 Setup: Instal...
Cancel
Nullsoft Install System v2.40
< Back
Close

Output folder: C:\Program Files\AGC

```
Extract: Root
Output folder: C:\Program Files\AGC\library source\lib
Extract: agclibv1_0.a... 100%
Output folder: C:\Program Files\AGC\library source\lib\CVS
Extract: Entries
Extract: Entries.Extra
Extract: Entries.Extra.Old
Extract: Entries.Old
Extract: Repository... 100%
Extract: Root
Output folder: C:\Program Files\AGC
```

5. Click **Close** to close the dialog box.
6. In the next dialog box, click **Yes** to view the help file; otherwise, click **No**. This completes the AGC Library installation.

As part of the installation process, the folder, `AGC`, is created. Refer to **Section 2.2 "AGC Library Files"** for more information on these files.

2.2 AGC LIBRARY FILES

The AGC library installer creates a directory named AGC. This directory contains these six folders.

- demo
- docs
- h
- lib
- wavefiles

2.2.1 demo folder

This folder contains files that are required by the AGC library quick start demonstration, and contains these three subfolders.

- h
- libs
- src

Table 2-1 provides a description of these subfolders.

TABLE 2-1: DEMONSTRATION FILES

File Name	Description
demo\AGC demo.hex	Demonstration hex file.
demo\AGC demo.mcp	Demonstration MPLAB® project file.
demo\AGC demo.mcw	Demonstration MPLAB workspace.
demo\cleanup.bat	Batch file script to clean intermediate build files.
demo\h\dsPICDEM1_1Plus.h	C header file containing dsPICDEM™ 1.1 Plus Development Board routines.
demo\h\lcd.h	C header file defining the interface to the LCD driver.
demo\h\agc_api.h	C header file defining the interface to the AGC library.
demo\h\Si3000Drv.h	C header file defining the interface to the Si3000 Codec driver.
demo\h\libs\agclibv1_0.a	AGC library archive file.
demo\src\dsPICDEM1_1Plus.c	C source file containing the routines for the dsPICDEM 1.1 Plus Development Board.
demo\src\lcd.s	Assembly source code for communicating with the LCD controller.
demo\src\main.c	C source file containing the main speech processing routine.
demo\src\lcd_strings.c	C source file for LCD Display.
demo\src\Si3000Drv.c	C source file containing the implementation of the Si3000 Codec driver.

2.2.2 docs Folder

This folder contains a PDF of the AGC library user's guide. To view this document, Acrobat® Reader® is required. The user's guide can also be downloaded from the Microchip web site (www.microchip.com).

This folder also contains a Windows Help file for the AGC library Application Programming Interface (API) as listed in Table 2-2.

TABLE 2-2: HELP

File Name	Description
`AGC Help.chm`	Windows help file containing a description of the AGC library API.

2.2.3 `h` Folder

This folder contains an include file for the AGC library as listed in Table 2-3.

TABLE 2-3: INCLUDE FILE

File Name	Description
`agc_api.h`	Include file containing the interface to the AGC library. This file must be included in the application to use the AGC library.

2.2.4 `lib` Folder

This folder contains a library archive file for the AGC library as listed in Table 2-4.

TABLE 2-4: LIBRARY FILE

File Name	Description
`agclibv1_0.a`	AGC library archive file. This file must be included in the application in order to use the AGC library.

2.2.5 `wavefiles` Folder

This folder contains two WAVE files, which can be used with the quick start demomstration. The WAVE files can be played back on the PC using a media player with the repeat function ON to make the WAVE file run continuously. A description of these files is provided in Table 2-5.

TABLE 2-5: WAVEFILES

File Name	Description
`microchip_sas_fem_8K.wav`	A WAVE file of a female voice sampled at 8 kHz.
`microchip_sas_male_8K.wav`	A WAVE file of a male voice sampled at 8 kHz.

Chapter 3. Quick Start Demonstration

This chapter describes the AGC library quick start demonstration. Topics covered include:

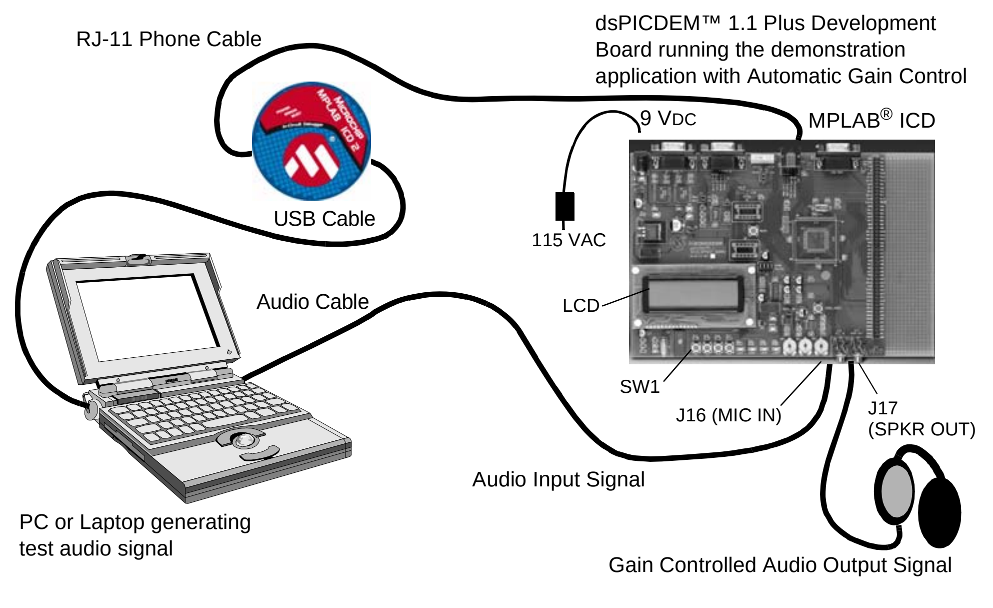
- Demonstration Summary
- Demonstration Setup
- Demonstration Procedure
- Demonstration Code Description

3.1 DEMONSTRATION SUMMARY

A demonstration application program included with the AGC library demonstrates the library's functionality. In the demonstration setup (illustrated in 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.

FIGURE 3-1: SETUP FOR AGC LIBRARY DEMO


RJ-11 Phone Cable
dsPICDEM™ 1.1 Plus Development Board running the demonstration application with Automatic Gain Control
9 VDC
MPLAB® ICD
USB Cable
115 VAC
Audio Cable
LCD
SW1
J16 (MIC IN)
J17 (SPKR OUT)
Audio Input Signal
PC or Laptop generating test audio signal
Gain Controlled Audio Output Signal

You can use the WAVE 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 plays out 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 SW1. LED1 is now switched on and ON is display 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 the Media Player on your PC to allow the WAVE 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 WAVE file. If you want to avoid this, sound editor programs usually provide for continuous looping. A good, freeware sound editor is Audacity, available from http://audacity.sourceforge.net/.

3.2 DEMONSTRATION SETUP

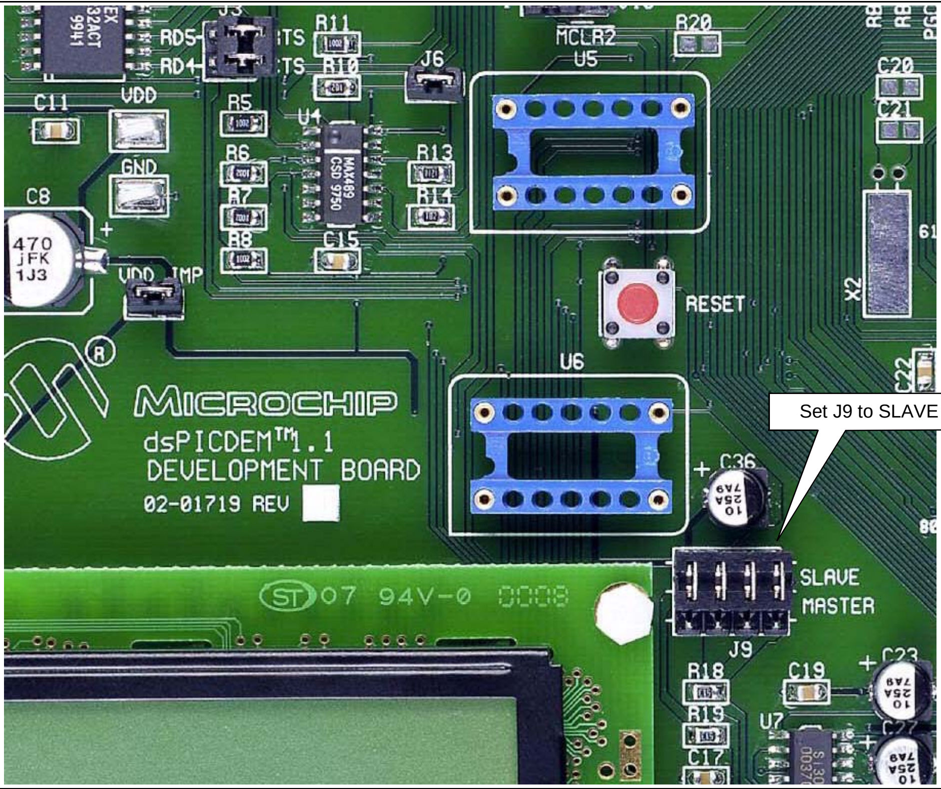
The demonstration application is run on a dsPICDEM™ 1.1 Plus Development Board (not included with the software license). Use the procedures outlined in the following sections to set up the demonstration.

3.2.1 Configure dsPICDEM™ 1.1 Plus Development Board

Before applying power, 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' (J16) jack on the dsPICDEM 1.1 Plus development board.
3. Connect the headset or speaker to the 'SPKR OUT' (J17) jack.
4. Connect the MPLAB ICD 2 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.

FIGURE 3-2: DEVELOPMENT BOARD SETUP

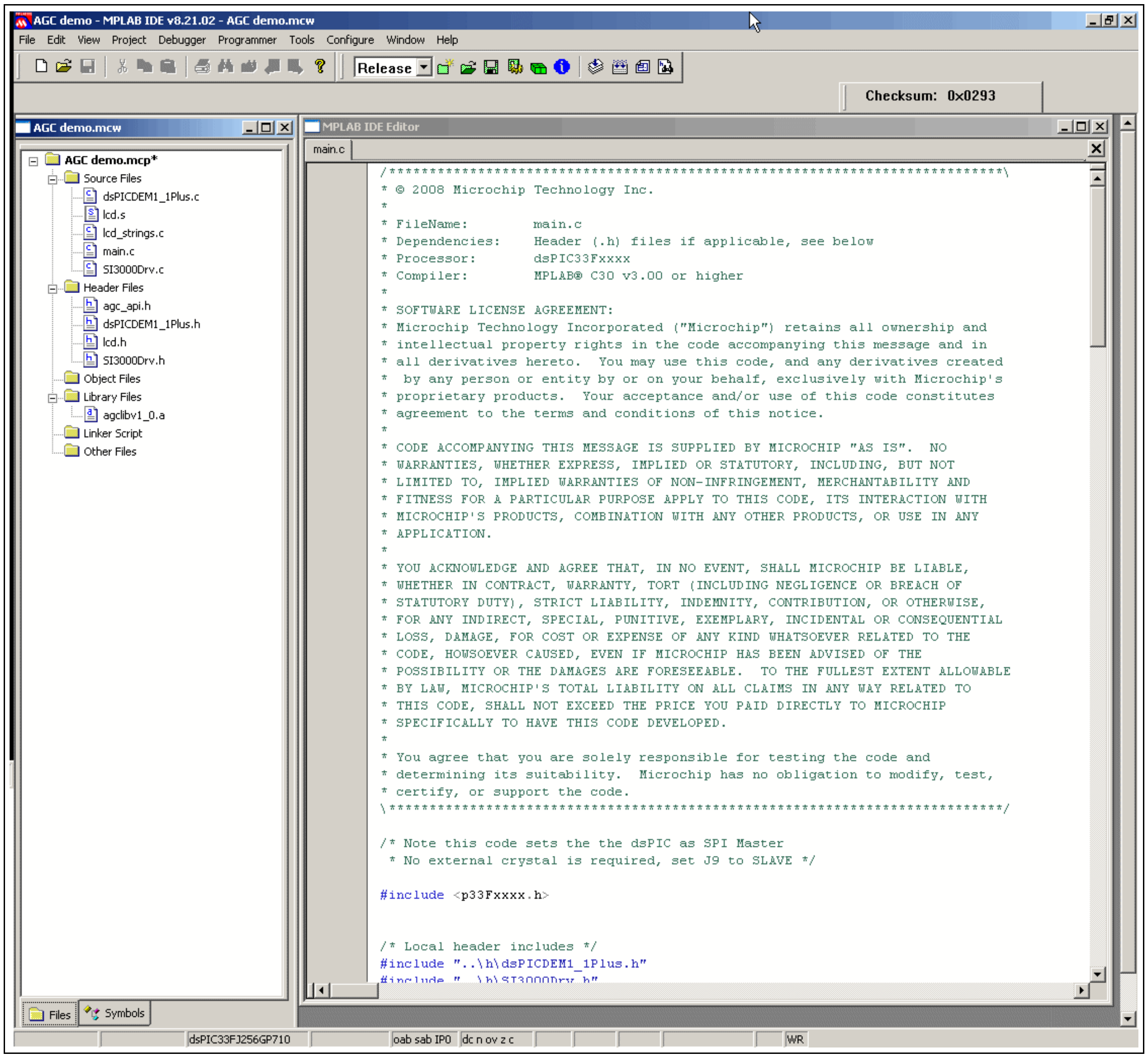


3.2.2 Program the dsPIC® DSC Device

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 AGC `demo.mcp` project located in the `demo` folder (see Figure 3-3). For more information on using MPLAB IDE, refer to the *“MPLAB IDE User’s Guide”* (DS51025).

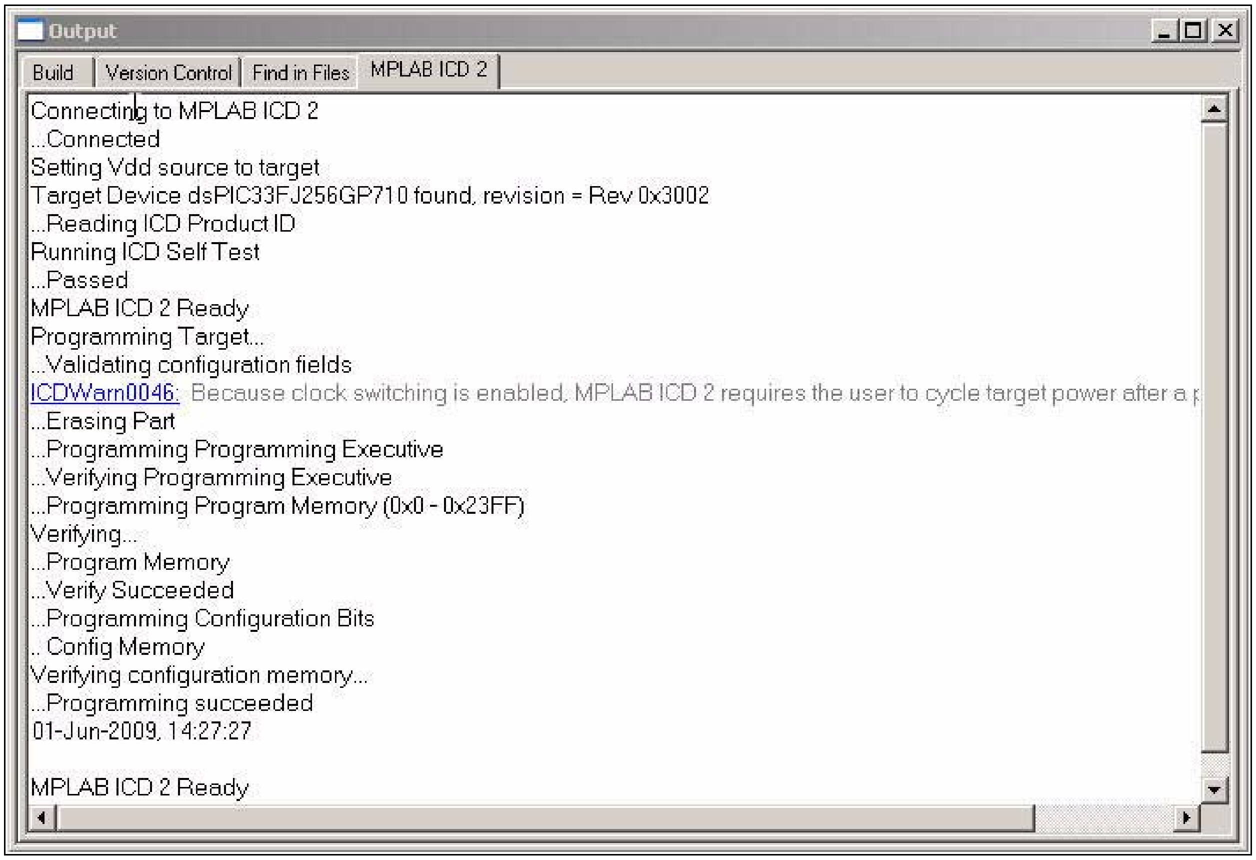
FIGURE 3-3: AGC DEMO PROJECT IN MPLAB IDE



```
/*****************************************************************************\
* © 2008 Microchip Technology Inc.
*
* FileName:        main.c
* Dependencies:    Header (.h) files if applicable, see below
* Processor:       dsPIC33Fxxxx
* Compiler:        MPLAB® C30 v3.00 or higher
*
* SOFTWARE LICENSE AGREEMENT:
* Microchip Technology Incorporated ("Microchip") retains all ownership and
* intellectual property rights in the code accompanying this message and in
* all derivatives hereto.  You may use this code, and any derivatives created
*  by any person or entity by or on your behalf, exclusively with Microchip's
* proprietary products.  Your acceptance and/or use of this code constitutes
* agreement to the terms and conditions of this notice.
*
* CODE ACCOMPANYING THIS MESSAGE IS SUPPLIED BY MICROCHIP "AS IS".  NO
* WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT
* LIMITED TO, IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY AND
* FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS CODE, ITS INTERACTION WITH
* MICROCHIP'S PRODUCTS, COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY
* APPLICATION.
*
* YOU ACKNOWLEDGE AND AGREE THAT, IN NO EVENT, SHALL MICROCHIP BE LIABLE,
* WHETHER IN CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE OR BREACH OF
* STATUTORY DUTY), STRICT LIABILITY, INDEMNITY, CONTRIBUTION, OR OTHERWISE,
* FOR ANY INDIRECT, SPECIAL, PUNITIVE, EXEMPLARY, INCIDENTAL OR CONSEQUENTIAL
* LOSS, DAMAGE, FOR COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE
* CODE, HOWSOEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE
* POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE FULLEST EXTENT ALLOWABLE
* BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO
* THIS CODE, SHALL NOT EXCEED THE PRICE YOU PAID DIRECTLY TO MICROCHIP
* SPECIFICALLY TO HAVE THIS CODE DEVELOPED.
*
* You agree that you are solely responsible for testing the code and
* determining its suitability.  Microchip has no obligation to modify, test,
* certify, or support the code.
\*****************************************************************************/

/* Note this code sets the the dsPIC as SPI Master
 * No external crystal is required, set J9 to SLAVE */

#include <p33Fxxxx.h>


/* Local header includes */
#include "..\h\dsPICDEM1_1Plus.h"
#include "..\h\SI3000Drv.h"
```

2. Import the project hex file (*File>Import>AGC demo.hex*).
3. From the Programmer menu, select **Connect** to link to the MPLAB ICD 2. The Output window shows that the MPLAB ICD 2 is ready.
4. Connect the MPLAB ICD 2 to the dsPICDEM 1.1 Plus Development Board.
5. Program the dsPIC DSC device on the board. From the Programmer menu, select **Program**. The Output window displays the download process and indicates that the programming has succeeded (see Figure 3-4).
6. When the program is loaded, disconnect the MPLAB ICD 2 from the board.

FIGURE 3-4: DOWNLOAD STATUS



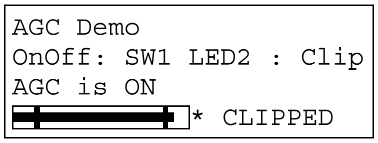
```
Connecting to MPLAB ICD 2
...Connected
Setting Vdd source to target
Target Device dsPIC33FJ256GP710 found, revision = Rev 0x3002
...Reading ICD Product ID
Running ICD Self Test
...Passed
MPLAB ICD 2 Ready
Programming Target...
...Validating configuration fields
ICDWarn0046: Because clock switching is enabled, MPLAB ICD 2 requires the user to cycle target power after a p
...Erasing Part
...Programming Programming Executive
...Verifying Programming Executive
...Programming Program Memory (0x0 - 0x23FF)
Verifying...
...Program Memory
...Verify Succeeded
...Programming Configuration Bits
.. Config Memory
Verifying configuration memory...
...Programming succeeded
01-Jun-2009, 14:27:27

MPLAB ICD 2 Ready
```

3.3 DEMONSTRATION PROCEDURE

With the demonstration application programmed into the device, the demonstration is ready to run. You can use the provided WAVE 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 Si3000 voice band codec and the Data Converter Interface (DCI) module of the dsPIC DSC device. The dsPIC DSC device then plays out 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 via the LEDs 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 3-5.

FIGURE 3-5: DEMONSTRATION RUN-TIME LCD SCREEN



```
AGC Demo
OnOff: SW1 LED2 : Clip
AGC is ON
* CLIPPED
```

The run-time screen displays the following:

1. The name of the demonstration.
2. SW1 is used to switch AGC on and off. LED2 serves as a CLIP indicator.
3. The current state of the algorithm.
4. A VU meter showing the input level. The band 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 level change in the input signal. This can be observed by changing the input level.

The AGC is enabled by pressing 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 level of the input signal at the set level. This can be observed by changing the input level.

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

3.4 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 `AGC_init()` from the AGC library, which initializes the AGC algorithm to its default state.

The main function also calls `SI3000_open()` 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 port bit 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.

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 output 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.

The AGC library in the demonstration is configured to default settings. The target amplitude is 50% of Full scale. The clip threshold is set at 97%. The AGC headroom is set at 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.

NOTES:

Chapter 4. Application Programming Interface (API)

This chapter describes in detail the Application Programming Interface (API) available for the AGC library. Topics covered include:

- 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

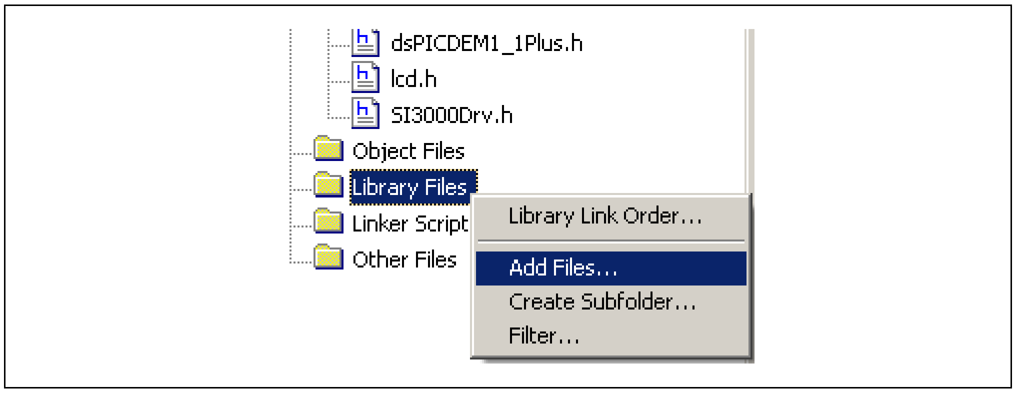
4.1 ADDING THE AGC LIBRARY TO AN APPLICATION

To use the AGC library in an application, the library archive must be added to the application project workspace and the file `agc_api.h` must be included in the application code.

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

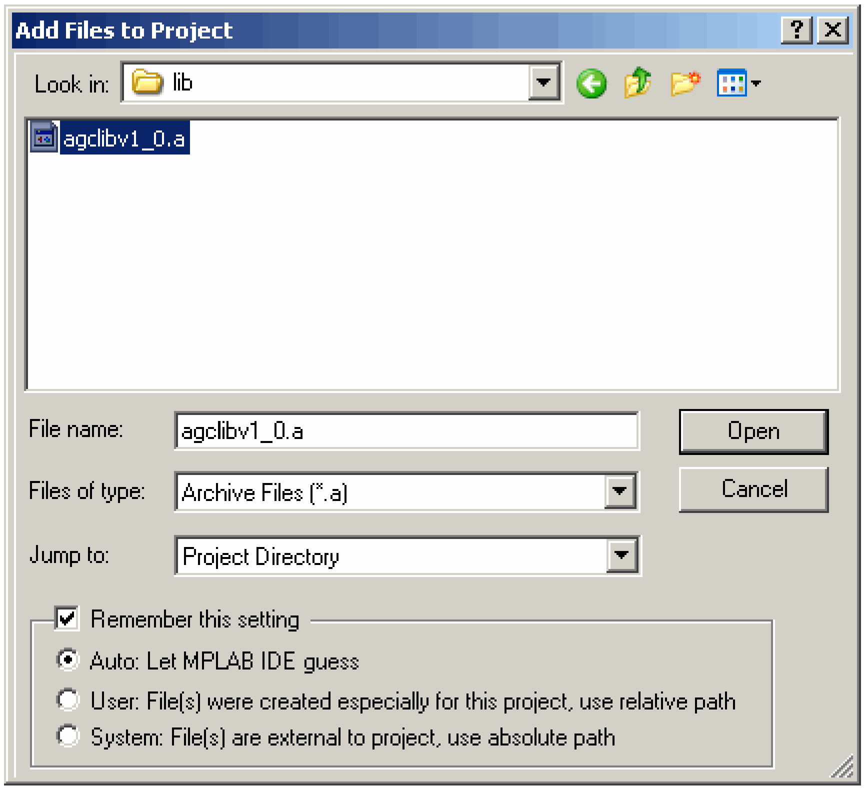
1. In the application MPLAB workspace, right click **Library Files** in the project window and select **Add Files**.

FIGURE 4-1:



2. Browse to the location of the `agclibv1_0.a` file (available in the `lib` folder within the installation directory).
3. Select the file and click **Open**.

FIGURE 4-2:


Add Files to Project

Look in: lib

agclibv1_0.a

File name: agclibv1_0.a

Files of type: Archive Files (*.a)

Jump to: Project Directory

[✓] Remember this setting

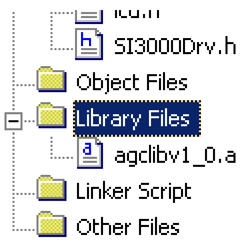
[● Auto: Let MPLAB IDE guess]

User: File(s) were created especially for this project, use relative path

System: File(s) are external to project, use absolute path

4. The library is now added to the application.

FIGURE 4-3:



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

4.2 AGC OPERATION AND PARAMETERS

This section explains the operation of the AGC algorithm. 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 millisecond speech frame. If the frame is classified as being a part of a speech burst, its 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, then the overall amplitude of the signal is decreased at a specified rate, which is called the release rate.

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

4.2.2 AGC Algorithm Parameters

The operation of the AGC algorithm is controlled by changing its parameters. A description of all the parameters is provided here.

- **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 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 a 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 too high a 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 climbs above a critical level. This level is specified by the headroom. The headroom is always more than the Amplitude Target but less than 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 a 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 hooks to the application to control the gain of an external codec or a Programmable Gain Amplifier. This is 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 then 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 `agclibv1_0.a` and `agc_api.h` files in the application. The minimum coding steps required to use the AGC library are listed here:

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`.

   The following coding steps 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 (i.e., 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. To simply usage, the library provides a `Q15()` macro, which converts a floating point number to a Q15 fractional value in hexadecimal format.

4.5 RESOURCE REQUIREMENTS

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

4.5.1 Program Memory Usage

TABLE 4-1:

	Size (bytes)	Section
Code in Program Memory	1140	`.libagc`
Tables in Program Memory	1611	`.const`
Total Program Memory	2751	

4.5.2 Data Memory Usage (8 kHz sampling rate)

TABLE 4-2:

	Size (bytes)	Alignment	Section
`agc_state_mem_x`	292	2	X data memory
`Sin`	160	2	X data memory
Total Data Memory	1284		

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.

4.5.3 Estimated Dynamic Memory Usage

TABLE 4-3:

Section	Size (bytes)
Heap	0
Stack	< 320

4.5.4 Computational Speed

TABLE 4-4:

Function	MIPS	Typical Call Frequency
`AGC_init()`	< 0.5	Once
`AGC_apply()`	0.33 (8 kHz) to 1.98 (48 kHz)	10 ms
All other functions	Minimal	As required

4.5.5 Data Format

`Sin` can be 10-, 12-, or 16-bit linear 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

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

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 (e.g., 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

`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_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

`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

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

`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

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 tips provide information on ways to optimize the AGC algorithm in a given application:

1. The optimal value for the Headroom is 75% of the full scale range. The optimal value for the Amplitude Target 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 too high a value will make the AGC algorithm overly sensitive, giving all syllables a uniformly high stress. Too low a value will make the AGC slow to respond. A value of 0.3 will keep the speech sounding natural.
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 input signals where syllables are exaggerated or where the input signal is more musical in nature.
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 be spiky.
7. For deterministic operation of the AGC algorithm, some guidelines need to be followed:
   a) Amplitude Target < Headroom < 95% of Full Scale.
   b) 0.0 < Attack rate < Recovery Rate.
   c) 0.0 < Leakage Rate < Release rate < Recovery Rate.
   d) Attack rate < Recovery Rate < 0.5.

Index

NOTES:

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://support.microchip.com
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

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

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: 86-10-8528-2100
Fax: 86-10-8528-2104

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

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

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

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

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

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: 886-7-536-4818
Fax: 886-7-536-4803

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

03/26/09