dsPIC® DSC Noise Suppression Library User's Guide

ISBN: 978-1-61341-246-6

dsPIC® DSC NOISE SUPPRESSION LIBRARY 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 online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This preface contains general information that will be useful to know before using the dsPIC® DSC Noise Suppression Library. Items discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Warranty Registration
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This user's guide describes how to use the dsPIC DSC Noise Suppression Library. This user's guide is composed of the following chapters:

- **Chapter 1. "Introduction"** – This chapter introduces the dsPIC DSC Noise Suppression 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 Noise Suppression Library demonstration on a PC.
- **Chapter 3. "Quick Start Demonstration"** – This chapter provides a hands-on demonstration of noise suppression in a working application.
- **Chapter 4. "Application Programming Interface (API)"** – This chapter outlines how the API functions provided in the dsPIC DSC Noise Suppression Library can be included in your application software through the API.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	<u>*File>Save*</u>
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
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 (in source code)	`0xFF, 'A'`
`Italic Courier New`	A variable argument	*`file.o`*, where `file` can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options] file [options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`

	Represents code supplied by user	`void main (void)`
		`{ ...`
		`}`

WARRANTY REGISTRATION

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

RECOMMENDED READING

This user's guide describes how to use the dsPIC DSC Noise Suppression Library. The following Microchip documents are available from the Microchip web site (www.microchip.com), and are recommended as supplemental reference resources.

dsPIC30F Family Reference Manual (DS70046)

Refer to this document for detailed information on dsPIC30F device operation. This reference manual explains the operation of the dsPIC30F Digital Signal Controller (DSC) family architecture and peripheral modules but does not cover the specifics of each device. For more information, refer to the specific device data sheet.

dsPIC33F/PIC24H Family Reference Manual

Refer to this document for detailed information on dsPIC33F/PIC24H device operation. These reference manual sections explain the operation of the dsPIC33F/PIC24H DSC family architecture and peripheral modules, but do not cover the specifics of each device. For more information, refer to the specific device data sheet.

dsPIC33E/PIC24E Family Reference Manual

Refer to this documents for detailed information on dsPIC33E/PIC24E device operation. These reference manual sections explain the operation of the dsPIC33E/PIC24E DSC family architecture and peripheral modules, but do not cover the specifics of each device. For more information, refer to the specific device data sheet.

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

This manual is a software developer's reference for the 16-bit PIC24F and PIC24H MCU and 16-bit dsPIC30F and dsPIC33F DSC families of devices. It describes the instruction set in detail and also provides general information to assist in developing software for these device families.

MPLAB® Assembler, Linker and Utilities for PIC24 MCUs and dsPIC® DSCs User's Guide (DS51317)

MPLAB Assembler for PIC24 MCUs and dsPIC® DSCs (formerly MPLAB ASM30) produces relocatable machine code from symbolic assembly language for the dsPIC DSC and PIC24 MCU device families. The assembler is a Windows console application that provides a platform for developing assembly language code. The assembler is a port of the GNU assembler from the Free Software Foundation (www.fsf.org).

MPLAB® C Compiler for PIC24 MCUs and dsPIC® DSCs User's Guide (DS51284)

This document describes the features of the optimizing C compiler, including how it works with the assembler and linker. The assembler and linker are discussed in detail, in the *"MPLAB® Assembler, Linker and Utilities for PIC24 MCUs and dsPIC® DSCs User's Guide"* (DS51317).

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 via our web site at: http://www.microchip.com. This web site makes files and information easily available to customers. Accessible by most Internet browsers, 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 listings
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listings of seminars and events; and 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 the Microchip MPLAB REAL ICE™ In-Circuit Emulator.
- **In-Circuit Debuggers** – The latest information on the Microchip In-Circuit Debuggers, MPLAB ICD 3 and MPLAB PICkit™ 3.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 device programmer and the PICkit™ 3 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

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

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

DOCUMENT REVISION HISTORY

Revision D (August 2008)

This revision includes the following updates for version 5.0 of the Noise Suppression Library. The previous version of the library was 4.0.

This document has been renamed from "*dsPIC30F Noise Suppression Library User's Guide"* to its new name of "*dsPIC® DSC Noise Suppression Library User's Guide"*. There has been no change to the Microchip literature number (DS70133) other than a revision letter update.

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

TABLE 1-1: MAJOR UPDATES

Chapter Name	Update Description
Chapter 1. "Introduction"	The optional accessory kit is no longer available and the related information has been removed. The reference to the five user functions has been removed. A new set of API functions has been added. See **4.4 "Noise Suppression Library API Functions"**. The host system requirements section was updated to include the HTML browser requirement.
Chapter 2. "Installation"	The installation procedure was updated to accommodate the installation CD file changes. See **2.1 "Installation Procedure"**. The Noise Suppression Library files have been updated. The `inc` folder was renamed to `h`, and the `wavefiles` folder was added. See **2.2 "Noise Suppression Library Files"**.
Chapter 3. "Quick Start Demonstration" (formerly Chapter 6. Noise Suppression Demo)	The Speaker Out port number has been changed in from J16 to J17. See **3.1 "Quick Start Demonstration for dsPIC33F Device Family"**. The dsPICDEM™ 1.1 Plus Development Board jumper (J9) setting has been changed from MASTER to SLAVE. See **3.1.2 "Demonstration Setup"** and Figure 3-2. Due to the removal of the optional Acoustic Accessory Kit, the reference to the 14.7456 MHz oscillator in the Demonstration Setup section has been removed. The demonstration procedure has been completely rewritten to provide more information on the state of operation. See **3.1.3 "Demonstration Procedure"**.
Chapter 4. "Application Programming Interface (API)"	This chapter has been updated with a completely new set of API functions. See **4.4 "Noise Suppression Library API Functions"**. The information in the formerly known Noise Suppression Algorithm chapter was consolidated and relocated to the Application Programming Interface (API) chapter. See **4.2 "Library Usage"**. The information in the formerly known Resource Requirements chapter was consolidated and relocated to the Application Programming Interface (API) chapter. See **4.3 "Resource Requirements"**.

Revision E (June 2011)

This revision includes the following updates for version 6.0 of the Noise Suppression Library. The previous version of the library was 5.0.

TABLE 1-2: UPDATES

Chapter Name	Update Description
"Preface"	Updated the **"Recommended Reading"** section
Chapter 1. "Introduction"	• Updated the first paragraph in **Chapter 1. "Introduction"** • Updated the list of minimum characteristics that a PC-compatible host system requires, in **1.3 "Host System Requirements"**
Chapter 2. "Installation"	• Updated **2.1 "Installation Procedure"** • Removed **FIGURE 2-1** through **FIGURE 2-5** in **2.1 "Installation Procedure"** • Removed the Note in **2.1 "Installation Procedure"** • Updated the directory labeled `NS v5.0` to `NS v6.0` in **2.2 "Noise Suppression Library Files"** • Updated Table 2-1 in **2.2.1 "demo Folder"** • Updated Table 2-3 in **2.2.4 "lib Folder"** • Updated **2.2.4 "lib Folder"**
Chapter 3. "Quick Start Demonstration"	• Updated the first paragraph in **Chapter 3. "Quick Start Demonstration"** • Updated **3.1 "Quick Start Demonstration for dsPIC33F Device Family"** • Updated the project name in step 1 and file name in step 2, in **3.1.2.2 "Programming the dsPIC DSC Device"** • Removed **FIGURE 3-3** and **FIGURE 3-4** in **3.1.2.2 "Programming the dsPIC DSC Device"**
Chapter 4. "Application Programming Interface (API)"	• Removed **FIGURE 4-1** through **FIGURE 4-3** in **4.1 "Adding the Noise Suppression Library to an Application"** • Updated step 4 in **4.1 "Adding the Noise Suppression Library to an Application"** • Added Step 1 in **4.2 "Library Usage"** • Updated Example 4-1 in **4.2 "Library Usage"** • Added a note in **4.2 "Library Usage"** • Updated Table 4-1, Table 4-2 and Table 4-4, in **4.3 "Resource Requirements"**

Chapter 1. Introduction

The dsPIC DSC Noise Suppression Library provides an algorithm to suppress the effect of noise in a speech signal. The library supports the dsPIC33F and dsPIC33E device families. This chapter provides an overview of the library, and covers these topics:

- Noise Suppression Library Overview
- Features
- Host System Requirements

1.1 NOISE SUPPRESSION LIBRARY OVERVIEW

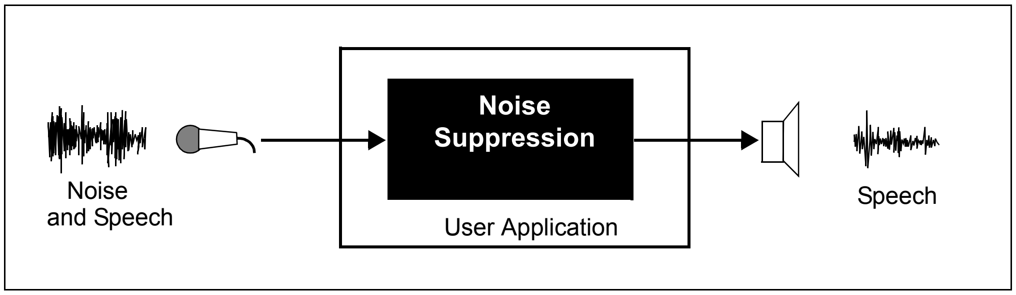
The Noise Suppression Library is useful for microphone-based applications in which the incoming speech (audio) signal is susceptible to ambient noise. This library is especially suitable for systems such as hands-free cell phones, speaker phones, intercom and teleconferencing systems, headsets, and in general, any microphone-based application that needs to eliminate unwanted noise. The Noise Suppression Library does not need an additional acoustically isolated noise reference.

The Noise Suppression Library is written almost entirely in Assembly language. It is highly optimized to make extensive use of the dsPIC Digital Signal Control (DSC) device architecture including its Digital Signal Processor (DSP) instruction set and advanced addressing modes. The algorithm has been implemented keeping in mind the need to avoid data overflow. The library functions can be easily called by the user application through a well-documented Application Programmer's Interface (API).

The Noise Suppression function removes noise from a 10 ms block of 16-bit speech data sampled at 8 kHz. This function is primarily a Frequency Domain algorithm in which a Fast Fourier Transform (FFT) is performed on each 10 ms block of data to analyze the frequency components of the signal. Thereafter, a Voice Activity Detection (VAD) algorithm is used to determine if the signal segment is speech or noise. The Noise Suppression algorithm maintains a profile of the noise and updates it every time a noise segment is detected by the VAD. Every frequency band of the input signal is then scaled according to the proportion of noise contained in that frequency band, thereby causing a significant degree of noise suppression in the resultant signal. The algorithm adapts to changes in the nature and level of noise and does not require a separate noise reference input.

Figure 1-1 illustrates a simplified block diagram of a user application using noise suppression.

FIGURE 1-1: NOISE SUPPRESSION IN AN APPLICATION



1.2 FEATURES

The dsPIC DSC Noise Suppression Library provides the following features:

- Simple user interface with only one library file and one header file
- All functions are called from a C application program
- Full compliance with the Microchip C30 Compiler, Assembler and Linker
- Highly optimized assembly code that uses DSP instructions and advanced addressing modes
- Comprehensive API that provides parametric control of the Noise Suppression engine
- Noise reduction level can be controlled from 0 dB to 44 dB
- Audio bandwidth of 0 kHz to 4 kHz at an 8 kHz sampling rate
- Library also includes:
  - Several sample `.wav` files that incorporate different types of noise components
  - User's Guide
  - Sample demonstration application with complete source code

1.3 HOST SYSTEM REQUIREMENTS

The dsPIC DSC Noise Suppression 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, or Windows 2007

Chapter 2. Installation

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

- Installation Procedure
- Noise Suppression Library Files

2.1 INSTALLATION PROCEDURE

To install the library:

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

The installation process creates the folder, `NS v6.0`, which contains the files described in **2.2 "Noise Suppression Library Files"**.

2.2 NOISE SUPPRESSION LIBRARY FILES

The dsPIC DSC Noise Suppression Library CD creates a directory titled `NS v6.0`. This directory contains these folders:

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

2.2.1 `demo` Folder

This folder contains files that are required by the dsPIC DSC Noise Suppression Library Quick Start Demonstration. This folder contains these sub-folders:

- `h`
- `libs`
- `src`

Table 2-1 describes the files in these sub-folders.

TABLE 2-1: DEMONSTRATION FILES

File Name	Description
`dsPIC33F NS demo.hex`	Demo hex file for dsPIC33F.
`dsPIC33E NS demo.hex`	Demo hex file for dsPIC33E.
`dsPIC33F NS demo.mcp`	Demo MPLAB Project File for dsPIC33F.
`dsPIC33E NS demo.mcp`	Demo MPLAB Project File for dsPIC33E.
`cleanup.bat`	A batch file script to delete the intermediate build files.
`h\dsPICDEM1_1Plus.h`	C header file for the dsPICDEM™ 1.1 Plus development board routines.
`h\MEB.h`	C header file for the Multimedia Expansion Board (MEB) routines.
`h\lcd.h`	C header file defining interface to the LCD driver.
`h\main.h`	C header file for function and macros in `main.c`.
`h\ns_api.h`	C header file defining the interface to the Noise Suppression Library.
`h\Si3000CodecDrv.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\nslibv6_33F.a`	The dsPIC DSC Noise Suppression Library archive file for dsPIC33F
`libs\nslibv6_33E.a`	The dsPIC DSC Noise Suppression Library archive file for dsPIC33E.
`src\dsPICDEM1_1Plus.c`	C source file containing routines for the dsPICDEM 1.1 Plus development board.
`src\MEB.c`	C source file containing routines for MEB.
`src\lcd_strings.c`	C source file for LCD display.
`src\main.c`	C source file containing the main speech processing routine.
`src\Si3000CodecDrv.c`	C source file containing the driver routines for Si3000 Codec.
`src\WM8731CodecDRV.c`	C source file containing the driver routines for WM8731 Codec.
`src\lcd.s`	Assembly routines for communicating with the LCD controller.

2.2.2 `doc` Folder

This folder contains the user's guide for the dsPIC DSC Noise Suppression Library. To view this document, double-click the file name. The user's guide can also be downloaded from the Microchip web site (www.microchip.com).

2.2.3 `h` Folder

This folder contains an include file for the Noise Suppression Library as shown in Table 2-2.

TABLE 2-2: INCLUDE FILE

File Name	Description
`ns_api.h`	Include file that contains the interface to the Noise Suppression Library. This file must be included in the application to use the Noise Suppression Library.

2.2.4 `lib` Folder

This folder contains the library archive files for the Noise Suppression Library as listed in Table 2-3. The archive names are suffixed with the names of the target device families, dsPIC33F or dsPIC33E.

TABLE 2-3: LIBRARY FILES

File Name	Description
`nslibv6_33F.a`	This is the Noise Suppression Library archive file for dsPIC33F. This file must be included in the application to use the Noise Suppression Library.
`nslibv6_33E.a`	This is the Noise Suppression Library archive file for dsPIC33E. This file must be included in the application to use the Noise Suppression Library.

2.2.5 `wavefiles` Folder

This folder contains the Wave files that can be used to evaluate the performance of the Noise Suppression Library when exposed to different noise environment. 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. The descriptions of these files are provided in Table 2-4.

TABLE 2-4: WAVE FILES

File Name	Description
`Beverage_plant_babble.wav`	Speech segment with unintelligible or meaningless (babble) noise.
`Critical_white.wav`	Speech segment with white noise.
`Friends_70mph.wav`	Speech segment with noise similar to that heard in a car cabin.
`Late_morning_15mph.wav`	Speech segment with noise similar to that heard in a car cabin.
`Purple_tie_70mph.wav`	Speech segment with noise similar to that heard in a car cabin.
`Quench_tone.wav`	Speech segment with single tone noise.
`Taxi_white.wav`	Speech segment with white noise.
`Twilight_babble.wav`	Speech segment with unintelligible or meaningless (babble) noise.
`Unusual_15mph.wav`	Speech segment with noise similar to that heard in a car cabin.

NOTES:

Chapter 3. Quick Start Demonstration

This chapter describes the dsPIC DSC Noise Suppression Library quick start demonstration for the dsPIC33F and dsPIC33E device families.

3.1 QUICK START DEMONSTRATION FOR dsPIC33F DEVICE FAMILY

The following topics are covered in this section:

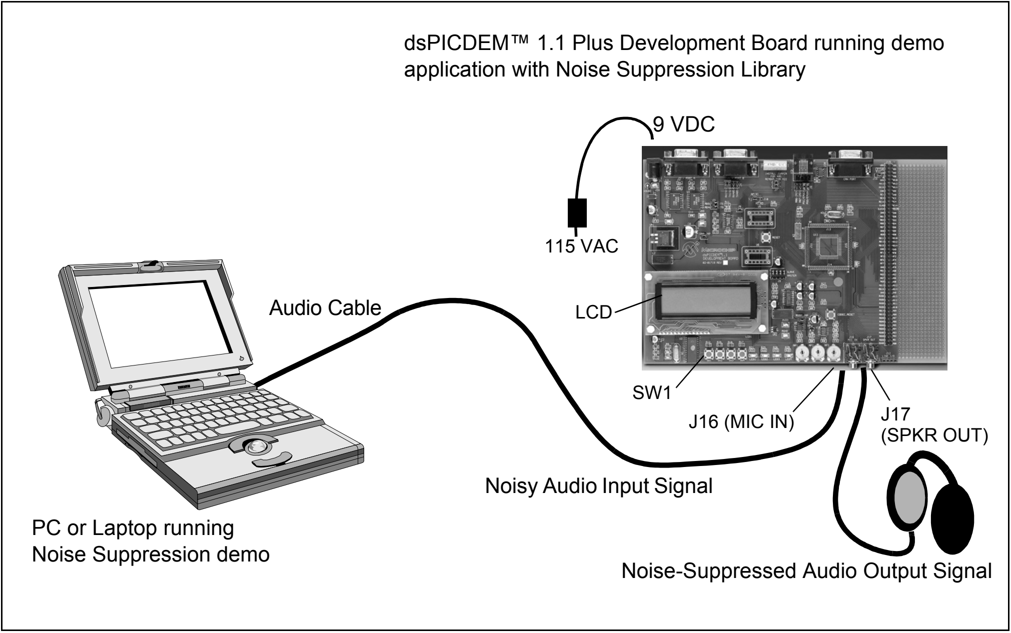
- Demonstration Summary
- Demonstration Setup
- Demonstration Procedure
- Demonstration Code Description

3.1.1 Demonstration Summary

A dsPIC33F-based demonstration application program included with the dsPIC DSC Noise Suppression Library demonstrates the functionality of the library. In the demonstration setup (see Figure 3-1), a dsPICDEM™ 1.1 Plus development board is configured as a system that receives a noise-corrupted speech signal through its microphone input port, suppresses the noise in the sampled signal, and plays out the noise-suppressed 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 noise-corrupted speech 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 J17 (SPKR OUT) on the dsPICDEM 1.1 Plus development board.

FIGURE 3-1: SETUP FOR NOISE SUPPRESSION DEMO


dsPICDEM™ 1.1 Plus Development Board running demo application with Noise Suppression Library
9 VDC
115 VAC
Audio Cable
LCD
SW1
J16 (MIC IN)
J17 (SPKR OUT)
Noisy Audio Input Signal
PC or Laptop running Noise Suppression demo
Noise-Suppressed Audio Output Signal

You can use the `.wav` files provided with the demo (in the `wavefiles` folder of the installation directory) as noise-corrupted speech signals, or you can provide your own signals. The noise-corrupted input signal is captured by 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 (noise-suppressed) signal through the device's DCI module and the on-board Si3000 codec.

When started, the program initializes with noise suppression turned OFF, indicated by LED1 turned OFF and OFF being written to the LCD screen. With noise suppression OFF, the signal heard in the headset contains noticeable noise.

The noise suppression is enabled by pressing the switch, SW1. LED1 is now turned ON and ON is written to the LCD. The speech signal heard on the headset becomes relatively noise-free compared to the original signal. Use switches, SW2 and SW3, to control the noise suppression level. Note that increasing the noise suppression level also increases the overall attenuation of the signal.

Turn on the Repeat function in your PC's media player, to allow the `.wav` file to run continuously. Observe when the signal is noisy and when it is noise suppressed. Repeat the process with several `.wav` files.

Note: Some media players insert a break before each repeat of the `.wav` file. If you want 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 intended to run on a dsPICDEM 1.1 Plus development board (not included with the software license).

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

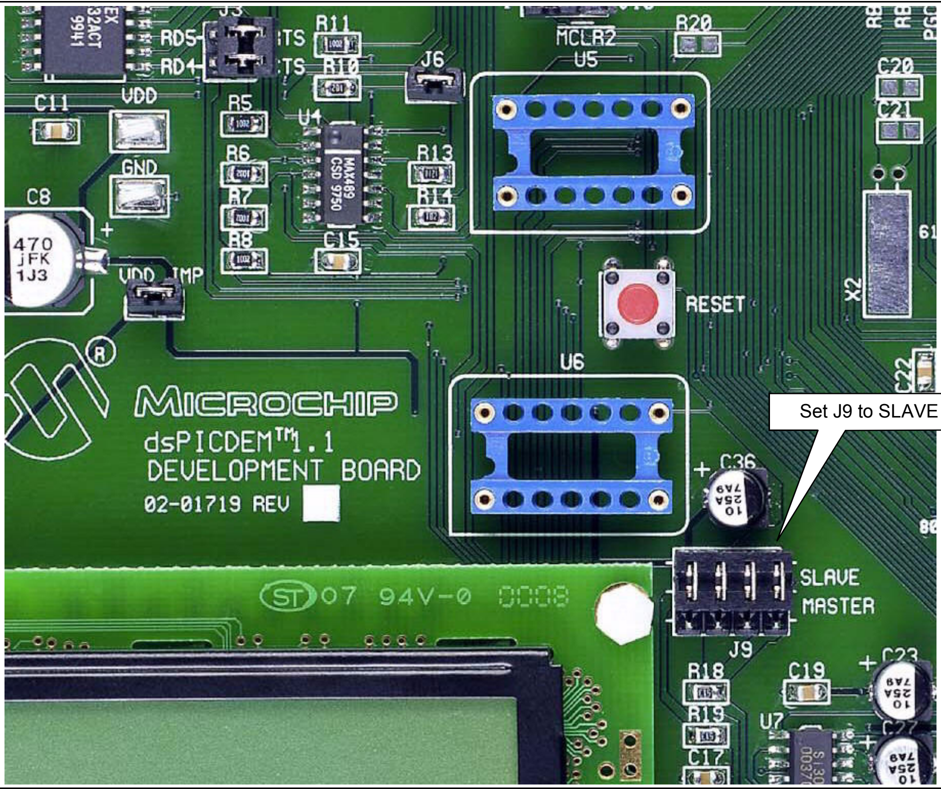
3.1.2.1 CONFIGURE dsPICDEM 1.1 PLUS DEVELOPMENT BOARD

Before applying the power, you need to configure the board:

1. Set the 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 'Microphone' 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™ In-Circuit Emulator can be used instead of MPLAB ICD 3.

FIGURE 3-2: DEVELOPMENT BOARD SETUP



3.1.2.2 PROGRAMMING THE dsPIC DSC DEVICE

Use the following process to load the noise suppression demonstration into the dsPIC DSC device on the dsPICDEM 1.1 Plus development board.

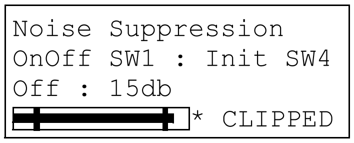
1. On your PC, launch MPLAB IDE and open the `dsPIC33F NS demo.mcp` project that is located in the `demo` folder.
   For more information on using MPLAB IDE, refer to the *"MPLAB® IDE User's Guide"* (DS51025).
2. Import the project hex file (*File > Import > `dsPIC33F NS demo.hex`*).
3. Select *Programmer > Connect* to link the MPLAB ICD 3 to the dsPIC DSC target device. The Output window confirms that the MPLAB ICD 3 is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming has succeeded.
5. When the program is loaded, disconnect the MPLAB ICD 3 from the board (remove the phone cable from the MPLAB ICD 3 connector).

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, which are located in the `wavefiles` folder of the installation directory as noise-corrupted speech signals, or you can provide your own signals. The noise-corrupted 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 plays the processed (noise-suppressed) signal through the DCI module of the device 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 illustrated in Figure 3-3).

FIGURE 3-3: DEMONSTRATION RUN-TIME LCD



```
Noise Suppression
OnOff SW1 : Init SW4
Off : 15db
           * CLIPPED
```

The run-time screen displays the following:

1. The name of the algorithm.
2. SW1 is used to turn the noise suppression ON and OFF. SW4 is used to reinitialize the Noise Suppression algorithm.
3. The current state of the algorithm (OFF) and the selected noise reduction level.
4. A volume unit (VU) meter showing the input level. The bands show an acceptable input range. The * indicates that noise suppression has interpreted the frame as speech. The word CLIPPED is displayed when the input signal is too large.

When started, the program initializes with noise suppression turned OFF, indicated by LED1 being turned OFF and OFF displayed on the LCD. With noise suppression OFF, the signal heard in the headset contains noticeable noise.

The noise suppression is enabled by pressing SW1. LED1 is now turned ON. ON is displayed on the LCD and the speech signal heard on the headset becomes relatively noise-free compared to the original signal.

Turn ON the Repeat function in your PC's media player, to allow the `.wav` file to run continuously. Observe when the signal is noisy and when it is noise suppressed. To experiment with different types and levels of noise, play several of the `.wav` files provided in the `wavefiles` folder.

The amount of noise reduction can be reduced in 1 dB steps (down to 0 dB) by pressing SW2. The amount of noise reduction can be increased in 1 dB steps (up to 44 dB) by pressing SW3. The noise suppression can be reinitialized by pressing SW4.

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 Noise Suppression Library functions.

The main function calls the `NS_init()` function from the Noise Suppression Library, which initializes the Noise Suppression algorithm to its default state.

The main function also calls the `Si3000CodecInit()` 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.

Subsequently, the `Si3000CodecStart()` function is used to initialize 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 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 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 `signalIn` array and the `NS_apply` function from the Noise Suppression Library is called with `signalIn` as the input data frame. The `signalIn` data buffer, which is also the output of the `NS_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 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.

To toggle the noise suppression ON and OFF, external interrupts for SW1 are enabled. In the main loop, the value of `applyNS` is read and passed to `NS_apply` as the enable flag. If `applyNS` is ‘`0`’, the Noise Suppression Library is still called, but the input/output buffer is not changed. This enables the Noise Suppression Library to maintain adaptation to changes in noise, even though it is not enabled.

3.2 QUICK START DEMONSTRATION FOR dsPIC33E DEVICE FAMILY

The following topics are covered in this section:

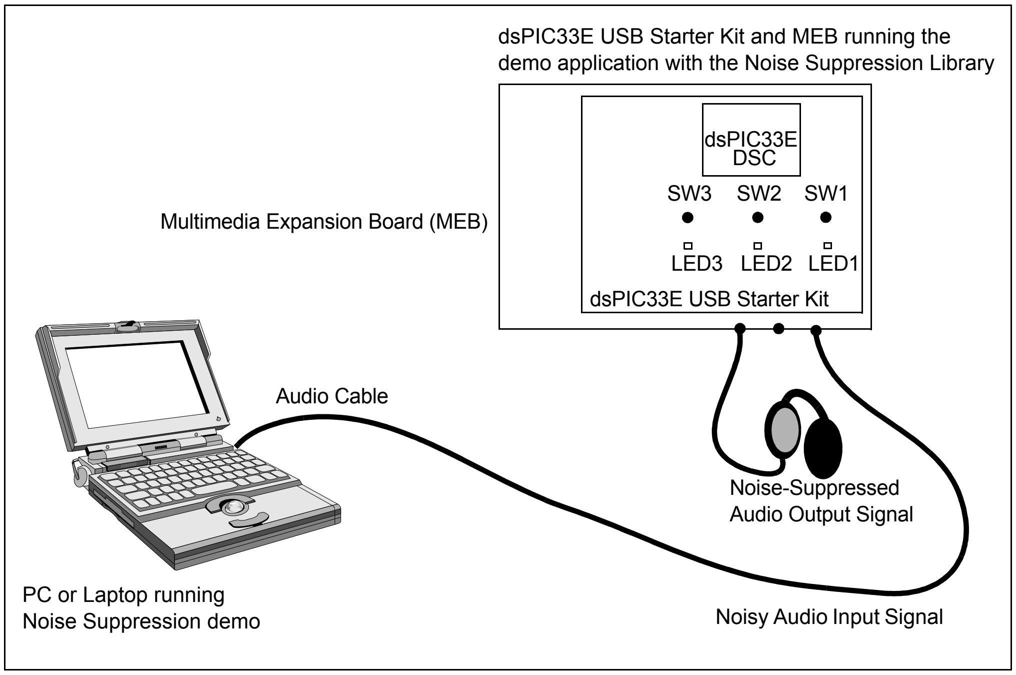
- Demonstration Summary
- Demonstration Setup
- Demonstration Procedure
- Demonstration Code Description

3.2.1 Demonstration Summary

A demonstration application program created specifically for the dsPIC33E family of devices is included with the dsPIC DSC Noise Suppression Library. In the demonstration setup (see Figure 3-4), a Multimedia Expansion Board (MEB) in conjunction with a dsPIC33E USB Starter Kit is configured as a system that receives a noise-corrupted speech signal through its microphone input port, suppresses the noise in the sampled signal, and outputs the noise-suppressed 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 noise-corrupted speech 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 speaker output of the MEB.

FIGURE 3-4: SETUP FOR NOISE SUPPRESSION DEMO


dsPIC33E USB Starter Kit and MEB running the demo application with the Noise Suppression Library
dsPIC33E DSC
SW3
SW2
SW1
LED3
LED2
LED1
dsPIC33E USB Starter Kit
Multimedia Expansion Board (MEB)
Audio Cable
Noise-Suppressed Audio Output Signal
PC or Laptop running Noise Suppression demo
Noisy Audio Input Signal

You can use the `.wav` files provided with the demo (in the `wavefiles` folder of the installation directory) as noise-corrupted speech signals, or you can provide your own signals. The noise-corrupted input signal is captured by the on-board WM8731 voice audio codec and the DCI module of the dsPIC DSC device. The dsPIC DSC device then outputs the processed (noise-suppressed) signal through the device's DCI module and the on-board WM8731 codec.

When started, the program initializes with noise suppression turned OFF, indicated by LED3 being turned OFF on the dsPIC33E USB Starter Kit. With noise suppression OFF, the signal heard in the headset contains noticeable noise.

The noise suppression is enabled by pressing the switch SW1on the dsPIC33E USB Starter Kit. LED3 is now turned ON and the speech signal heard on the headset becomes relatively noise-free compared to the original signal. Use switches SW2 and SW3 to control the noise suppression level. Note that increasing the noise suppression level also increases the overall attenuation of the signal.

Turn ON the Repeat function in your PC's media player to allow the `.wav` file to run continuously. Then, observe when the signal is noisy and when it is noise suppressed. Repeat the process with several `.wav` files.

Note: Some media players insert a break before each repeat of the `.wav` file. If you want 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 intended to run on an MEB and dsPIC33E USB Starter Kit (not included with the software). Use the procedures outlined in the following sections to set up the demonstration.

3.2.2.1 CONFIGURE MEB AND dsPIC33E USB STARTER KIT

Before applying power, you need to configure the board:

1. Insert a dsPIC33E USB Starter Kit into the starter kit connector on 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 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 this process to load the noise suppression demo into the dsPIC DSC device:

1. On your PC, launch MPLAB IDE and open the `dsPIC33E NS demo.mcp` project located in the `demo` folder.
   For more information on using MPLAB IDE, refer to *"MPLAB® IDE User's Guide"* (DS51025).
2. Import the project hex file by selecting *File > Import > `dsPIC33E NS demo.hex`*.
3. Select Starter Kit on Board as the Programmer, and then select *Programmer > Connect* to link to the dsPIC DSC target device. The Output window confirms that the target device is ready.
4. Select *Programmer > Program*. The Output window displays the download process and indicates that the programming 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 demo 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 as noise-corrupted speech signals, or you can provide your own signals. The noise-corrupted input signal is sampled through the on-board WM8731 voice band codec and the DCI module of the dsPIC DSC device. The dsPIC DSC device then plays out the processed (noise-suppressed) signal through the device's DCI module and the on-board WM8731 codec.

When started, the program initializes with noise suppression turned OFF, indicated by LED3 turned OFF on the dsPIC33E USB Starter Kit. With noise suppression OFF, the signal heard in the headset contains noticeable noise.

The noise suppression is enabled by pressing switch SW1 on the dsPIC33E USB Starter Kit. LED3 is now turned ON. The speech signal heard on the headset becomes relatively noise-free compared to the original signal.

Turn on the Repeat function in your PC's media player to allow the `.wav` file to run continuously. Observe when the signal is noisy and when it is noise suppressed. To experiment with different types and levels of noise, play several of the `.wav` files provided in the `wavefiles` folder.

The amount of noise reduction can be reduced in 1 dB steps (down to 0 dB) by pressing SW2 for a few seconds. The amount of noise reduction can be increased in 1 dB steps (up to 44 dB) by pressing SW3 for a few seconds.

The noise suppression can be reinitialized by pressing switch S1 on the LCD side of the MEB.

3.2.4 Demonstration Code Description

The demonstration code runs on a dsPIC33E device, using the Primary Oscillator as the clock source with the PLL configured for 40 MIPS operation.

The file, `main.c`, contains the main function for the demonstration application. This file (`main.c`) 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 Noise Suppression Library functions.

The main function calls the `NS_init()` function from the Noise Suppression Library, which initializes the Noise Suppression 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, that is, two transmit slots and two receive slots are used in this demonstration.

Subsequently, the `WM8731Start ()` function is used to enable the DCI module and I^{2}C module. The codec is configured for a sample rate of 8 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 `signalIn` array and the `NS_apply` function from the Noise Suppression Library is called with `signalIn` as the input data frame. The `signalIn` data buffer, which is also the output of the `NS_apply` function after it has been executed, is played out on the speaker output of the headset.

In the main loop, the value of `applyNS` is read and passed to `NS_apply` as the enable flag. If `applyNS` is '`0`', the Noise Suppression Library is still called, but the input/output buffer is not changed. This enables the Noise Suppression Library to maintain adaptation to changes in noise, even though it is not enabled.

Chapter 4. Application Programming Interface (API)

This chapter describes the Application Programming Interface (API) available in the dsPIC DSC Noise Suppression Library. The following topics are covered in this chapter:

4.1 ADDING THE NOISE SUPPRESSION LIBRARY TO AN APPLICATION

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

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

1. In the application MPLAB workspace, right-click **Library Files** in the Project Window and select **Add files**.
2. Browse to the location of either the `nslibv6_33F.a` file or the `nslibv6_33E.a` file, both of which are available in the `libs` folder in the installation directory.
3. Select the desired file, and then click **Open**.
4. The library is added to the application. Verify that the library archive is shown in the MPLAB project.

4.2 LIBRARY USAGE

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

There are at least five coding steps that need to be performed to enable use of the Noise Suppression Library:

1. Allocate the memory for the input/output data array. The array size should be `NS_FRAME`. The array should be placed in X memory and should be aligned to a 4-byte boundary.
2. Allocate the memory for the Noise Suppression algorithm state holder: This memory is an integer array in X memory aligned at an address boundary of 4 bytes. The `NS_XSTATE_MEM_SIZE_INT` macro specifies the size of this array. Every audio channel to be processed requires its own state holder.
3. Allocate the memory for the Noise Suppression algorithm X and Y scratch memories: The X scratch is an integer array in X memory aligned at an address boundary of 4 bytes. This Y scratch memory is an integer array in Y memory aligned at an address boundary of 512 bytes. Multiple audio channels can share the scratch memories.

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

4. Initialize the Noise Suppression algorithm state for each audio channel: Use the `NS_init()` function for initializing the noise suppression state for each audio channel.
5. Apply the Noise Suppression algorithm to an audio frame: Use the `NS_apply()` function to perform noise suppression on an audio frame. If a frame is not required to be processed by the Noise Suppression algorithm, the function should still be called with the enable parameter set to `NS_FALSE`. This allows the Noise Suppression algorithm to continue adapting to the noise in the audio frame. The audio frame stays unaffected.

Example 4-1 provides a code example showing these five steps.

EXAMPLE 4-1: LIBRARY USAGE EXAMPLE

```
/* Data array */
int input1[NS_FRAME]_XBSS(4);                              /* Step 1 */
int input2[NS_FRAME]_XBSS(4);                              /* Step 1 */

/* Channel 1 memory structures */
int nsStateMemX1 [NS_XSTATE_MEM_SIZE_INT]   _XBSS(4); /* Step 2 */

/* Channel 2 memory structures */
int nsStateMemX2 [NS_XSTATE_MEM_SIZE_INT]   _XBSS(4); /* Step 2 */

/* Each instance can share the same X and Y scratch memory. */

int nsScratchX [NS_XSCRATCH_MEM_SIZE_INT]_XBSS(4);    /* Step 3 */
int nsScratchY [NS_YSCRATCH_MEM_SIZE_INT]_YBSS(512);  /* Step 3 */
.
.
.
void main()
{
   NS_init(nsStateMemX1, nsScratchX, nsScratchY);     /* Step 4 */
   NS_init(nsStateMemX2, nsScratchX, nsScratchY);     /* Step 4 */
   .
   .
   .
   while(1)
   {
      /* input1 and input2 are two different audio streams */

      NS_apply(nsStateMemX1, input1, NS_TRUE);        /* Step 5 */
      NS_apply(nsStateMemX2, input2, NS_TRUE);        /* Step 5 */
   }
}
```

4.3 RESOURCE REQUIREMENTS

The Noise Suppression Library requires the following computational resources while running on the dsPIC DSC device.

TABLE 4-1: PROGRAM MEMORY USAGE FOR dsPIC33F

Type	Size (bytes)	Section
Code in Program Memory	6675 (dsPIC30F/dsPIC33F) 6414 (dsPIC33E)	`.libns`
Tables in Program Memory	1383 (dsPIC30F/dsPIC33F) 0 (dsPIC33E)	`.const`
Total Program Memory	8058 (dsPIC30F/dsPIC33F) 6414 (dsPIC33E)	—

TABLE 4-2: DATA MEMORY USAGE

Function	Size (bytes)	Alignment	Section
`nsStateMemX`	768	4	X data memory
`scratchMemX`	100	4	X data memory
`scratchMemY`	256	512	Y data memory
`signalIn`	160	4	X data memory
Tables in Data Memory	0 (dsPIC30F/dsPIC33F) 922 (dsPIC33E)	4	X data memory
Total Data Memory	1284 (dsPIC30F/dsPIC33F) 2206 (dsPIC33E)	—	—

TABLE 4-3: ESTIMATED DYNAMIC USAGE

Section	Size (bytes)
Heap	0
Stack	< 300

TABLE 4-4: COMPUTATIONAL SPEED

Function	MIPS	Typical Call Frequency
`NS_init()`	< 0.5	Once
`NS_apply()`	3.6	10 ms
All other functions	Minimal	As required

4.3.1 Data Format

The input data to the Noise Suppression Library can be 10-bit, 12-bit or 16-bit linear Pulse Code Modulation (PCM) data. The Noise Suppression algorithm automatically adjusts for the data format used.

4.4 NOISE SUPPRESSION LIBRARY API FUNCTIONS

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

- `NS_init`
- `NS_relocateXScratchMem`
- `NS_relocateYScratchMem`
- `NS_apply`
- `NS_setNoiseReduction`
- `NS_getNoiseReduction`
- `NS_setVadHangover`
- `NS_getVadHangover`
- `NS_setNoiseFactor`
- `NS_getNoiseFactor`
- `NS_setNoiseEstimationFactor`
- `NS_getNoiseEstimationFactor`
- `NS_setNoiseFloor`
- `NS_getNoiseFloor`
- `NS_getIsSpeech`
- `NS_estimateSNR`
- `NS_TRUE`
- `NS_FALSE`
- `NS_SLOW`
- `NS_NORMAL`
- `NS_FAST`
- `NS_FRAME`
- `NS_XSTATE_MEM_SIZE_INT`
- `NS_XSCRATCH_MEM_SIZE_INT`
- `NS_YSCRATCH_MEM_SIZE_INT`
- `NS_NRLEVELDEFAULT`
- `NS_VADHANGOVERDEFAULT`
- `NS_NOISEFLOORDEFAULT`
- `NS_POWERFACTORDEFAULT`

NS_init

Description

Initializes the Noise Suppression algorithm.

Include

`ns_api.h`

Prototype

```
void NS_init(int* ptrStateX, int* xScratchMem, int*
yScratchMem);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`xScratchMem`	a pointer to an area of scratch memory in X RAM used by the Noise Suppression algorithm
`yScratchMem`	a pointer to an area of scratch memory in Y RAM used by the Noise Suppression algorithm

Return Value

None.

Remarks

None.

Code Example

```
   int nsStateMemX   [NS_XSTATE_MEM_SIZE_INT]_XBSS(4);
   int nsScratchX    [NS_XSCRATCH_MEM_SIZE_INT]_XBSS(4);
   int nsScratchY    [NS_YSCRATCH_MEM_SIZE_INT]_YBSS(512);
   .
   .
   .
   NS_init(nsStateMemX, nsScratchX, nsScratchY);
```

NS_relocateXScratchMem

Description

Changes the X memory scratch buffer used by the Noise Suppression algorithm.

Include

`ns_api.h`

Prototype

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

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`xScratchMem`	a pointer to the new Y scratch memory to be used by the Noise Suppression algorithm

Return Value

None.

Remarks

After relocating the X scratch memory, the older scratch memory is not used by the Noise Suppression Library.

Code Example

```
int nsStateMemX   [NS_XSTATE_MEM_SIZE_INT]   _XBSS(4);
int nsScratchX    [NS_XSCRATCH_MEM_SIZE_INT] _XBSS(4);
int nsScratchX2   [NS_XSCRATCH_MEM_SIZE_INT] _XBSS(4);
int nsScratchY    [NS_YSCRATCH_MEM_SIZE_INT] _YBSS(512);
.
.
.
NS_init(nsStateMemX, nsScratchX, nsScratchY);
```

Noise Suppression algorithm is using `nsScratchX` for its scratch memory.

```
.
.
.
NS_relocateXScratchMem(nsStateMemX, nsScratchX2);
```

Noise Suppression algorithm is now using `nsScratchX2` for its scratch memory.

NS_relocateYScratchMem

Description

Changes the Y memory scratch buffer used by the Noise Suppression algorithm.

Include

`ns_api.h`

Prototype

```
void NS_relocateYScratchMem(int* ptrStateX, int* yScratchMem);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`yScratchMem`	a pointer to the new Y scratch memory to be used by the Noise Suppression algorithm

Return Value

None.

Remarks

After relocating the Y scratch memory, the older scratch memory is not used by the Noise Suppression Library.

Code Example

```
int nsStateMemX   [NS_XSTATE_MEM_SIZE_INT]   _XBSS(4);
int nsScratchX    [NS_XSCRATCH_MEM_SIZE_INT] _XBSS(4);
int nsScratchY    [NS_YSCRATCH_MEM_SIZE_INT] _YBSS(512);
int nsScratchY2   [NS_YSCRATCH_MEM_SIZE_INT] _YBSS(512);
.
.
.
NS_init(nsStateMemX, nsScratchX, nsScratchY);
```

Noise Suppression algorithm is using `nsScratchY` for its Y memory scratch buffer.

```
.
.
.
NS_relocateYScratchMem(nsStateMemX, nsScratchY2);
```

Noise Suppression algorithm is now using `nsScratchY2` for its scratch memory.

NS_apply

Description

Applies noise reduction to the current frame of data.

Include

`ns_api.h`

Prototype

```
void NS_apply(int* ptrStateX, int* signalIn, int enable);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`SignalIn`	a pointer to the input/output buffer of size NS_FRAME
`Enable`	a flag to indicate if Noise Suppression is required for this buffer (`NS_TRUE/NS_FALSE`)

Return Value

None.

Remarks

The Noise Suppression processes the data “in-place”, meaning that the output is passed back in the input buffer. Setting Enable to `NS_FALSE` returns an unprocessed buffer of data, but the Noise Suppression algorithm still adapts on the data.

Code Example

```
int nsStateMemX   [NS_XSTATE_MEM_SIZE_INT]   _XBSS(4);
int nsScratchY    [NS_YSCRATCH_MEM_SIZE_INT] _YBSS(512);
int nsScratchX    [NS_XSCRATCH_MEM_SIZE_INT] _XBSS(4);
.
.
.
NS_init(nsStateMemX, nsScratchX, nsScratchY);
.
.
.
NS_apply(nsStateMemX, input, NS_TRUE);
```

NS_setNoiseReduction

Description

Sets the noise reduction level (in dB).

Include

`ns_api.h`

Prototype

```
void NS_setNoiseReduction(int* ptrStateX, int leveldB);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`leveldB`	an integer value between 0 and 44 representing the desired noise reduction level in dB

Return Value

None.

Remarks

The desired noise reduction level is measured using white noise. Other types of noise result in different amounts of noise reduction.

Code Example

```
   NS_setNoiseReduction(nsStateMemX, 18);
```

Sets the desired noise reduction level to 18 dB for the instance of the algorithm `nsStateMemX`.

NS_getNoiseReduction

Description

Returns the current noise reduction value (in dB).

Include

`ns_api.h`

Prototype

```
int NS_getNoiseReduction(int* ptrStateX);
```

Arguments

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

Return Value

The current desired noise reduction level in dB as an integer for the instance of the algorithm `nsStateMemX`.

Remarks

None.

Code Example

```
    int nrLevel;
    nrLevel = NS_getNoiseReduction(nsStateMemX);
```

`nrLevel` contains the desired noise reduction level for the instance of the algorithm `nsStateMemX`.

NS_setVadHangover

Description

Sets the VAD hangover value.

Include

`ns_api.h`

Prototype

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

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`value`	an integer value from 1 to 100 representing the desired VAD hangover

Return Value

None.

Remarks

The VAD hangover is used to adjust the hysteresis around the VAD. A larger value keeps the VAD active longer. Setting VAD hangover to '`1`' disables the hysteresis. Selecting a value outside the permitted range (1 to 100) results in the previous setting being retained.

Code Example

```
   NS_setVadHangover(nsStateMemX, 10);
```

Sets the desired VAD hangover to 10 frames (representing 100 ms of data) for the instance of the algorithm `nsStateMemX`.

NS_getVadHangover

Description

Returns the current VAD hangover.

Include

`ns_api.h`

Prototype

```
int NS_getVadHangover(int* ptrStateX);
```

Arguments

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

Return Value

VAD hangover value.

Remarks

None.

Code Example

```
    int vadHangover;
    vadHangover = NS_getVadHangover(nsStateMemX);
```

`vadHangover` contains the VAD hangover value set for the instance of the algorithm `nsStateMemX`.

NS_setNoiseFactor

Description

Sets the noise factor.

Include

`ns_api.h`

Prototype

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

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`value`	an integer value between 1 and 15 representing the desired noise factor

Return Value

None.

Remarks

Sets the noise factor used by the VAD to determine speech from noise. A higher value will flag more frames of data as noise. Legal values are in the range 1 to 15 inclusive.

Code Example

```
   NS_setNoiseFactor(nsStateMemX, 3);
```

Sets the noise factor to 3 for the instance of the algorithm `nsStateMemX`.

NS_getNoiseFactor

Description

Returns the current noise factor.

Include

```
ns_api.h
```

Prototype

```
int NS_getNoiseFactor(int* ptrStateX);
```

Arguments

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

Return Value

Noise factor.

Remarks

None.

Code Example

```
    int noiseFactor;
    noiseFactor = NS_getNoiseFactor(nsStateMemX);
```

`noiseFactor` contains the noise factor value set for the instance of the algorithm `nsStateMemX`.

`NS_setNoiseEstimationFactor`

Description

Sets the noise estimation factor.

Include

`ns_api.h`

Prototype

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

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`level`	either `NS_SLOW`, `NS_NORMAL` or `NS_FAST`

Return Value

None.

Remarks

This factor determines how quickly the Noise Suppression algorithm updates its noise estimate after detecting a noise frame.

Code Example

```
   NS_setNoiseEstimationFactor(nsStateMemX, NS_NORMAL);
```

Sets the noise factor to normal speed for the instance of the algorithm `nsStateMemX`.

NS_getNoiseEstimationFactor

Description

Returns the current noise estimation factor.

Include

`ns_api.h`

Prototype

```
int NS_getNoiseEstimationFactor(int* ptrStateX);
```

Arguments

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

Return Value

Noise estimation factor.

Remarks

None.

Code Example

```
    int noiseEstFactor;
    noiseEstFactor = NS_getNoiseEstimationFactor(nsStateMemX);
```

`noiseEstFactor` contains the noise estimation factor value set for the instance of the algorithm `nsStateMemX`, either `NS_SLOW`, `NS_NORMAL`, or `NS_FAST`.

NS_setNoiseFloor

Description

Sets the noise floor, below which the Noise Suppression algorithm will not update.

Include

```
ns_api.h
```

Prototype

```
void NS_setNoiseFloor(int* ptrStateX, long floor);
```

Arguments

`ptrStateX`	a pointer to the state memory for this instance of Noise Suppression
`floor`	a long integer value between representing the background noise level, below which the Noise Suppression algorithm will not update

Return Value

None.

Remarks

System noise represents the noise contributed by the application hardware. It does not include the noise contained in the input signal. For example, if the noise generated by hardware is experimentally determined to be -60 dBm0 or -66 dBov, the representative hexadecimal value for the noise floor is computed as shown in Equation 4-1.

EQUATION 4-1: NOISE FLOOR CALCULATION EXAMPLE

$$10^{\frac{\mathbf{-66}}{10}} \times 80 \times (2^{32} - 1) \approx 86307 = 0x15123$$

Note: The number in bold type represents the value under consideration.

In this case, setting the noise floor to 0x15123, which is supplied to the function, causes the Noise Suppression algorithm to stop adapting when the noise energy is below -60 dBm0. To disable this feature, set the value to zero.

Code Example

```
   NS_setNoiseFloor(nsStateMemX, 0x1D1917);
```

Sets the noise floor to 0x1D1917 for the instance of the algorithm `nsStateMemX`.

NS_getNoiseFloor

Description

Returns the current noise floor.

Include

`ns_api.h`

Prototype

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

Arguments

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

Return Value

None.

Remarks

Returns the current noise floor value.

Code Example

```
   long noiseFloor;
   noiseFloor = NS_getNoiseFloor(nsStateMemX);
```

`noiseFloor` contains the noise floor value set for the instance of the algorithm `nsStateMemX`.

NS_getIsSpeech

Description

Returns a flag indicating if the Noise Suppression algorithm thinks the current frame is speech or noise.

Include

`ns_api.h`

Prototype

```
int NS_getIsSpeech(int* ptrStateX);
```

Arguments

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

Return Value

An integer flag of either `NS_TRUE` or `NS_FALSE`.

Remarks

This is an indication of the presence of speech in the input signal. This flag is effected by the VAD hangover, noise floor and power factor.

Code Example

```
int speech;
speech = NS_getIsSpeech(nsStateMemX);
```

`speech` is a flag representing whether the Noise Suppression algorithm thinks the current frame contains speech.

NS_estimateSNR

Description

Estimates the Signal-to-Noise Ratio (SNR).

Include

`ns_api.h`

Prototype

```
int NS_estimateSNR(int* preNS, int* postNS);
```

Arguments

`preNS`	a pointer to the buffer of data before the Noise Suppression algorithm has been applied
`postNS`	a pointer to the buffer of data after the Noise Suppression algorithm has been applied

Return Value

An estimate of the SNR.

Remarks

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

Code Example

```
int snr;
for (i = 0; i < NS_FRAME; i++)
{
   signalBefore[i] = signalIn[i];
}
NS_apply(nsStateMemX, nsScratchY, signalIn, NS_TRUE);
snr = NS_estimateSNR(signalBefore, SignalIn);
```

`snr` contains the estimate of the SNR.

`NS_TRUE`

Description

Used to indicate true to the Noise Suppression algorithm.

Value

1

`NS_FALSE`

Description

Used to indicate false to the Noise Suppression algorithm.

Value

0

`NS_SLOW`

Description

Used in conjunction with `NS_setNoiseEstimationFactor` and `NS_getNoiseEstimationFactor`, to adjust the speed of adaptation to noise.

Value

100

`NS_NORMAL`

Description

Used in conjunction with `NS_setNoiseEstimationFactor` and `NS_getNoiseEstimationFactor`, to adjust the speed of adaptation to noise.

Value

200

NS_FAST

Description

Used in conjunction with `NS_setNoiseEstimationFactor` and `NS_getNoiseEstimationFactor`, to adjust the speed of adaptation to noise.

Value

300

NS_FRAME

Description

The size of the input buffer processed.

Value

80

NS_XSTATE_MEM_SIZE_INT

Description

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

Value

383

NS_XSCRATCH_MEM_SIZE_INT

Description

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

Value

50

`NS_YSCRATCH_MEM_SIZE_INT`

Description

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

Value

128

`NS_NRLEVELDEFAULT`

Description

Value in dB for the default noise reduction.

Value

15

`NS_VADHANGOVERDEFAULT`

Description

Value of the default VAD Hangover.

Value

6

`NS_NOISEFLOORDEFAULT`

Description

Value of the default noise floor.

Value

0x1D1917

`NS_POWERFACTORDEFAULT`

Description

Default value for the noise power factor.

Value

2

4.5 APPLICATION TIPS

Although the Noise Suppression algorithm makes a best effort to suppress the noise level in a signal, its performance can be optimized by proper selection of parameters. In general, experimentation with this library is encouraged.

A few tips for affecting the performance of the algorithm are as follows:

1. The optimum input signal levels for testing audio systems are generally considered to be between -10 dBm0 and -30 dBm0. If digital input speech levels have peaks that are up to three-fourths of full range, then good use is being made of the available precision; levels higher than this carry a risk of amplitude clipping.
2. Choose the noise reduction level range that best fits the application. A very high noise suppression level not only suppresses the noise, but also reduces the speech level to some extent.
3. The VAD Hangover can be used to adjust the rate at which the Noise Suppression algorithm adapts. Having a larger VAD Hangover increases the time the algorithm takes to adapt to changes in the noise level. A shorter VAD Hangover will allow the Noise Suppression algorithm to treat more input frames as noise frames, and therefore, adapt faster.
4. If the level of the noise compared to speech is high, you may want to increase the value of the Noise Factor.
5. The Noise Estimation Factor determines how quickly the Noise Suppression algorithm's Noise Estimate is updated after a noise frame has been detected. You may consider setting this parameter to `NS_SLOW` when the noise environment is known to be more static in nature.

Independent of the noise contained in the speech signal, the application hardware, like all electrical circuits, generates a small amount of noise which is still there when the microphone is disconnected. The Noise Suppression algorithm should freeze adaptation in this case, so that when the microphone is reconnected it will carry on as if nothing has happened. However, if it is found that the noise suppression has diverged, then the default noise floor is too low. To counter this, use the `NS_setNoiseFloor` function to increase the noise floor (some trial and error may be needed).

NOTES:

Index

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