



dsPIC[®] DSC 噪声抑制库 用户指南

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中更安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 (Digital Millennium Copyright Act)》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

提供本文档的中文版本仅为为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、dsPIC、KEELOQ、KEELOQ 徽标、MPLAB、PIC、PICmicro、PICSTART、PIC³² 徽标、rfPIC 和 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

FilterLab、Hampshire、HI-TECH C、Linear Active Thermistor、MXDEV、MXLAB、SEEVAL 和 The Embedded Control Solutions Company 均为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、Application Maestro、chipKIT、chipKIT 徽标、CodeGuard、dsPICDEM、dsPICDEM.net、dsPICworks、dsSPEAK、ECAN、ECONOMONITOR、FanSense、HI-TIDE、In-Circuit Serial Programming、ICSP、Mindi、MiWi、MPASM、MPLAB Certified 徽标、MPLIB、MPLINK、mTouch、Omniscient Code Generation、PICC、PICC-18、PICDEM、PICDEM.net、PICkit、PICtail、REAL ICE、rLAB、Select Mode、Total Endurance、TSHARC、UniWinDriver、WiperLock 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 是 Microchip Technology Inc. 在美国的服务标记。

在此提及的所有其他商标均为各持有公司所有。

© 2011, Microchip Technology Inc. 版权所有。

ISBN: 978-1-61341-763-8

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外，Microchip 在开发系统的设计和和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

目录

第 1 章 简介

1.1 噪声抑制库概述	11
1.2 特性	12
1.3 主机系统要求	12

第 2 章 安装

2.1 安装步骤	13
2.2 噪声抑制库文件	13

第 3 章 快速入门演示

3.1 dsPIC33F 器件系列的快速入门演示	17
3.2 dsPIC33E 器件系列的快速入门演示	22

第 4 章 应用程序编程接口 (API)

4.1 向应用程序添加噪声抑制库	25
4.2 函数库的使用	26
4.3 资源需求	28
4.4 噪声抑制库 API 函数	29
4.5 应用技巧	49

注:

前言

客户须知

所有文档均会过时，本文档也不例外。Microchip 的工具和文档将不断演变以满足客户的需求，因此实际使用中有些对话框和 / 或工具说明可能与本文档所述之内容有所不同。请访问我们的网站 (www.microchip.com) 获取最新文档。

文档均标记有 “DS” 编号。该编号出现在每页底部的页码之前。DS 编号的命名约定为 “DSXXXXXA”，其中 “XXXXX” 为文档编号，“A” 为文档版本。

欲了解开发工具的最新信息，请参考 MPLAB[®] IDE 在线帮助。从 Help（帮助）菜单选择 Topics（主题），打开现有在线帮助文件列表。

简介

本前言包含使用 dsPIC[®] DSC 噪声抑制库前需要了解的一般信息。内容包括：

- [文档编排](#)
- [本指南使用的约定](#)
- [保修登记](#)
- [Microchip 网站](#)
- [开发系统变更通知客户服务](#)
- [客户支持](#)
- [文档版本历史](#)

文档编排

本文档介绍如何使用 dsPIC DSC 噪声抑制库。本手册的内容编排如下：

- **第 1 章 “简介”** —— 介绍 dsPIC DSC 噪声抑制库并简要概述噪声抑制和函数库特性。还概述了对主机 PC 的要求。
- **第 2 章 “安装”** —— 提供在 PC 上安装噪声抑制库演示所需的详细信息。
- **第 3 章 “快速入门演示”** —— 提供实际应用中噪声抑制的动手实践演示。
- **第 4 章 “应用程序编程接口 (API)”** —— 概述如何通过 API 将 dsPIC DSC 噪声抑制库中提供的 API 函数包含在应用程序软件中。

本指南使用的约定

本手册采用以下文档约定：

文档约定

说明	涵义	示例
Arial 字体:		
斜体字	参考书目	<i>MPLAB® IDE User's Guide</i>
	需强调的文字	 仅有的编译器
首字母大写	窗口	Output 窗口
	对话框	Settings 对话框
	菜单选择	选择 Enable Programmer
引用	窗口或对话框中的字段名	“Save project before build”
带右尖括号且有下划线的斜体文字	菜单路径	<i><u>File>Save</u></i>
粗体字	对话框按钮	单击 OK
	选项卡	单击 Power 选项卡
0xn timer	十六进制数，n 是其中一位	0xFFFF, 0x007A
尖括号 <> 括起的文字	键盘上的按键	按 <Enter>, <F1>
Courier New 字体:		
常规 Courier New	源代码示例	#define START
	文件名	autoexec.bat
	文件路径	c:\mcc18\h
	关键字	_asm, _endasm, static
	命令行选项	-Opa+, -Opa-
	二进制位值	0, 1
	常量	0xFF, 'A'
'bnnnn	二进制数，n 是其中一位	'b00100, 'b10
斜体 Courier New	可变参数	<i>file.o</i> , 其中 <i>file</i> 可以是任一有效文件名
方括号 []	可选参数	mcc18 [选项] <i>file</i> [选项]
花括号和竖线: {}	选择互斥参数: “或”选择	errorlevel {0 1}
省略号 ...	代替重复文字	var_name [, var_name...]
	表示由用户提供的代码	void main (void) { ... }

保修登记

请填写随附的保修登记卡（Warranty Registration Card）并尽快寄出。寄出保修登记卡的用户将可收到新产品更新信息。可在 Microchip 网站上获得临时软件版本。

推荐读物

本用户指南介绍了如何使用 dsPIC DSC 噪声抑制库。Microchip 网站 (www.microchip.com) 提供以下 Microchip 文档，建议读者作为补充参考材料。

dsPIC30F 系列参考手册（DS70046E_CN）

有关 dsPIC30F 器件操作的详细信息，请参见此文档。此参考手册介绍了 dsPIC30F 数字信号控制器（Digital Signal Controller, DSC）系列架构和外设模块的工作原理，但并不涉及每款器件的具体信息。具体器件信息，请参见相应器件的数据手册。

dsPIC33F/PIC24H 系列参考手册章节

有关 dsPIC33F/PIC24H 器件操作的详细信息，请参见这些文档。这些参考手册章节介绍了 dsPIC33F/PIC24H DSC 系列架构和外设模块的工作原理，但并不涉及每款器件的具体信息。具体器件信息，请参见相应器件的数据手册。

dsPIC33E/PIC24E 系列参考手册章节

有关 dsPIC33E/PIC24E 器件操作的详细信息，请参见这些文档。这些参考手册章节介绍了 dsPIC33E/PIC24E DSC 系列架构和外设模块的工作原理，但并不涉及每款器件的具体信息。具体器件信息，请参见相应器件的数据手册。

16 位 MCU 和 DSC 程序员参考手册（DS70157E_CN）

此手册是供软件开发人员使用的 16 位 PIC24F/PIC24H MCU 以及 16 位 dsPIC30F/dsPIC33F DSC 系列器件的参考手册。它详细介绍了指令集，并提供了帮助用户进行这些器件系列软件开发的一般信息。

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

PIC24 MCU 和 dsPIC[®] DSC 的 MPLAB 汇编器（以前的 MPLAB ASM30）可从适用于 dsPIC DSC 和 PIC24 MCU 器件系列的符号汇编语言生成可重定位的机器代码。该汇编器是 Windows[®] 控制台应用程序，为开发汇编语言代码提供了平台。该汇编器是自由软件基金会（www.fsf.org）GNU 汇编器的移植版。

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

此文档介绍了优化 C 编译器的特性，包括如何将其与汇编器和链接器配合使用。“MPLAB[®] Assembler, Linker and Utilities for PIC24 MCUs and dsPIC[®] DSCs User's Guide”（DS51317）中详细介绍了汇编器和链接器。

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

有关安装和使用 MPLAB[®] 集成开发环境（Integrated Development Environment, IDE）软件的详细信息，请参见此文档。

MICROCHIP 网站

Microchip 网站 (<http://www.microchip.com>) 为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的互联网浏览器即可访问。网站提供以下信息：

- **产品支持** —— 数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持** —— 常见问题解答 (FAQ)、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务** —— 产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

开发系统变更通知客户服务

Microchip 的客户通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 <http://www.microchip.com>，点击“**变更通知客户**” (Customer Change Notification) 服务并按照注册说明完成注册。

开发系统产品组的分类如下：

- **编译器** —— Microchip C 编译器及其他语言工具的最新信息，包括 MPLAB® C 编译器、MPASM™ 和 MPLAB 16 位汇编器、MPLINK™ 和 MPLAB 16 位目标链接器，以及 MPLIB™ 和 MPLAB 16 位目标库管理器。
- **仿真器** —— Microchip MPLAB REAL ICE™ 在线仿真器的最新信息。
- **在线调试器** —— Microchip 在线调试器 MPLAB ICD 3 和 MPLAB PICKit™ 3 的最新信息。
- **MPLAB® IDE** —— 关于开发系统工具的 Windows® 集成开发环境 Microchip MPLAB IDE 的最新信息，主要针对 MPLAB IDE、MPLAB SIM 模拟器、MPLAB IDE 项目管理器以及一般编辑和调试功能。
- **编程器** —— Microchip 编程器的最新信息，包括 MPLAB PM3 器件编程器以及 PICKit™ 3 开发编程器。

客户支持

Microchip 产品的用户可通过以下渠道获取帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师 (FAE)
- 技术支持

客户应联系其代理商、代表或应用工程师 (FAE) 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 Microchip 网站 <http://support.microchip.com> 获取网上技术支持。

文档版本历史

版本 D（2008 年 8 月）

此版本包含噪声抑制库版本 5.0 的以下更新。该函数库的上一个版本是 4.0。

本文档的名称从《dsPIC30F 噪声抑制库用户指南》重命名为《dsPIC® DSC 噪声抑制库用户指南》。除了版本字母更新外，未对 Microchip 文献编号（DS70133）进行任何更改。

每章均经过大面积修改和重新组织以支持所有 dsPIC DSC 器件。上一版本的文档包含七章，这七章的内容已按下表所列进行合并：

表 1-1： 重大更新

章名	更新说明
第 1 章 “简介”	不再提供可选附件工具包，相关信息已删除。 删除了对五个用户函数的引用。增加了一个新的 API 函数集。请参见第 4.4 节 “噪声抑制库 API 函数”。 更新了主机系统要求一节，以包含 HTML 浏览器要求。
第 2 章 “安装”	更新了安装步骤，以适应安装 CD 文件的变化。请参见第 2.1 节 “安装步骤”。 更新了噪声抑制库文件。inc 文件夹重命名为 h，并增加了 wavefiles 文件夹。请参见第 2.2 节 “噪声抑制库文件”。
第 3 章 “快速入门演示” (以前的第 6 章，噪声抑制演示)	扬声器输出端口号从 J16 更改为 J17。请参见第 3.1 节 “dsPIC33F 器件系列的快速入门演示”。 dsPICDEM™ 1.1 Plus 开发板跳线（J9）设置从 MASTER 更改为 SLAVE。请参见第 3.1.2 节 “演示设置” 和图 3-2。 由于删除了可选声学附件工具包，“演示设置”一节中对 14.7456 MHz 振荡器的引用也已删除。 完全重写了演示步骤，以提供关于操作状态的更多信息。请参见第 3.1.3 节 “演示步骤”。
第 4 章 “应用程序编程接口 (API)”	此章已使用全新的 API 函数集更新。请参见第 4.4 节 “噪声抑制库 API 函数”。 以前的 “噪声抑制算法” 一章中的信息已合并到 “应用程序编程接口 (API)” 一章。请参见第 4.2 节 “函数库的使用”。 以前的 “资源要求” 一章中的信息已合并到 “应用程序编程接口 (API)” 一章。请参见第 4.3 节 “资源需求”。

版本 E（2011 年 6 月）

此版本包含噪声抑制库版本 6.0 的以下更新。该函数库的上一个版本是 5.0。

表 1-2: 更新

章名	更新说明
“前言”	更新了“推荐读物”一节
第 1 章 “简介”	- 更新了第 1 章 “简介” 中的第一段 - 更新了第 1.3 节 “主机系统要求” 中 PC 兼容主机系统要求的最低功能特性列表
第 2 章 “安装”	- 更新了第 2.1 节 “安装步骤” - 删除了第 2.1 节 “安装步骤” 中的图 2-1 至图 2-5 - 删除了第 2.1 节 “安装步骤” 中的注释 - 将第 2.2 节 “噪声抑制库文件” 中标记为 NS v5.0 的目录更新为 NS v6.0 - 更新了第 2.2.1 节 “demo 文件夹” 中的表 2-1 - 更新了第 2.2.4 节 “lib 文件夹” 中的表 2-3 - 更新了第 2.2.4 节 “lib 文件夹”
第 3 章 “快速入门演示”	- 更新了第 3 章 “快速入门演示” 中的第一段 - 更新了第 3.1 节 “dsPIC33F 器件系列的快速入门演示” - 更新了第 3.1.2.2 节 “烧写 dsPIC DSC 器件” 中第 1 步中的项目名称和第 2 步中的文件名 - 删除了第 3.1.2.2 节 “烧写 dsPIC DSC 器件” 中的图 3-3 和图 3-4
第 4 章 “应用程序编程接口 (API)”	- 删除了第 4.1 节 “向应用程序添加噪声抑制库” 中的图 4-1 至图 4-3 - 更新了第 4.1 节 “向应用程序添加噪声抑制库” 中的第 4 步 - 在第 4.2 节 “函数库的使用” 中增加了第 1 步 - 更新了第 4.2 节 “函数库的使用” 中的例 4-1 - 在第 4.2 节 “函数库的使用” 中增加了注释 - 更新了第 4.3 节 “资源需求” 中的表 4-1、表 4-2 和表 4-4

第 1 章 简介

dsPIC DSC 噪声抑制库提供可抑制语音信号中噪声影响的算法。该函数库支持 dsPIC33F 和 dsPIC33E 器件系列。本章提供该函数库的概述，涵盖以下主题：

- [噪声抑制库概述](#)
- [特性](#)
- [主机系统要求](#)

1.1 噪声抑制库概述

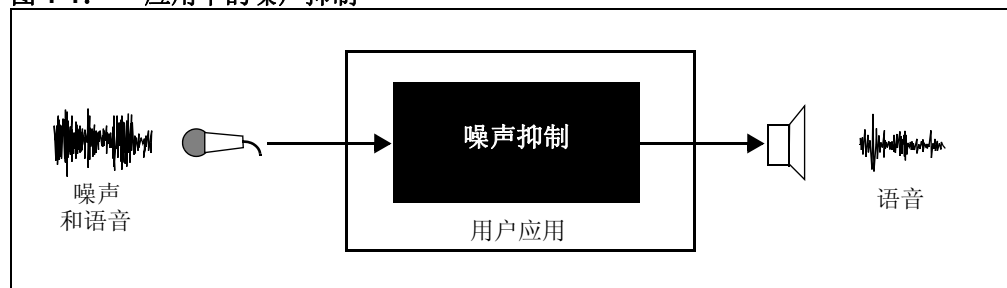
噪声抑制库对基于话筒的应用非常有用，在这类应用中输入的语音（音频）信号容易受到环境噪声的影响。该函数库特别适合于免提移动电话、扬声器电话、对讲机和电话会议系统、耳机以及通常意义上所有需要消除无用噪声的基于话筒的应用。噪声抑制库不需要额外的声学隔离噪声参考。

噪声抑制库几乎完全用汇编语言编写。它经过高度优化，能够充分利用 dsPIC 数字信号控制器（Digital Signal Controller，DSC）器件架构，包括其数字信号处理器（Digital Signal Processor，DSP）指令集和高级寻址模式。所实现的算法考虑到了避免数据溢出的要求。用户应用程序可通过完备的应用程序编程接口（Application Programming Interface，API）轻松调用库函数。

噪声抑制函数可从长度为 10 ms 的以 8 kHz 采样的 16 位语音数据块中去除噪声。此函数主要是一种频域算法，对每个 10 ms 数据块执行快速傅立叶变换（Fast Fourier Transform，FFT），以分析信号的频率分量。之后，使用语音活动检测（Voice Activity Detection，VAD）算法来确定信号段是语音还是噪声。噪声抑制算法会维持一条噪声曲线，并在每次 VAD 检测到噪声段时更新该曲线。然后，根据输入信号频带所含的噪声比例对每个频带进行换算，从而在最后输出的信号中实现显著的噪声抑制。该算法可自适应噪声性质和等级的变化，不需要单独的噪声参考输入。

[图 1-1](#) 显示了使用噪声抑制的用户应用的简化框图。

图 1-1: 应用中的噪声抑制



1.2 特性

dsPIC DSC 噪声抑制库具有以下特性：

- 简单的用户接口，仅一个库文件和一个头文件
- 所有函数都从 C 应用程序调用
- 完全兼容 Microchip C30 编译器、汇编器和链接器
- 高度优化的汇编代码，使用 DSP 指令和高级寻址模式
- 全面的 API，提供对噪声抑制引擎的参数控制
- 降噪级别可控，范围为 0 dB 到 44 dB
- 8 kHz 采样率下的音频带宽为 0 kHz 至 4 kHz
- 函数库还包含：
 - 多个包含不同类型噪声分量的采样 .wav 文件
 - 用户指南
 - 含完整源代码的演示应用程序示例

1.3 主机系统要求

dsPIC DSC 噪声抑制库要求 PC 兼容主机系统具有以下最低功能特性：

- 1 GHz 或更高的处理器
- HTML 浏览器
- 16 MB RAM
- 40 MB 可用硬盘空间
- Microsoft[®] Windows[®] 98、Windows 2000、Windows NT、Windows XP 或 Windows 7

第 2 章 安装

本章介绍 dsPIC DSC 噪声抑制库的安装步骤。要使用该函数库，必须将其安装到笔记本电脑或台式机上。安装后，可将库文件包含到目标应用程序中。本章涵盖的主题包括：

- [安装步骤](#)
- [噪声抑制库文件](#)

2.1 安装步骤

安装函数库：

1. 双击 NS_setup.exe。将显示许可协议屏幕。
2. 查看许可协议，然后单击 **I Agree**（我同意）继续。将显示 Installation Destination（安装目标文件夹）对话框。
3. 指定要安装函数库的位置（即目录），然后单击 **Install**（安装）。
4. 单击 **Close**（关闭）关闭该对话框。这将完成噪声抑制库的安装。

安装过程中将创建文件夹 NS v6.0，其中包含[第 2.2 节“噪声抑制库文件”](#)中介绍的文件。

2.2 噪声抑制库文件

dsPIC DSC 噪声抑制库 CD 将创建名为 NS v6.0 的目录。此目录包含以下文件夹：

- demo
- doc
- h
- lib
- wavefiles

2.2.1 demo 文件夹

此文件夹包含 dsPIC DSC 噪声抑制库快速入门演示所需的文件。此文件夹包含以下子文件夹：

- h
- libs
- src

[表 2-1](#) 列出了这些子文件夹中的文件。

表 2-1: 演示文件

文件名	说明
dsPIC33F NS demo.hex	dsPIC33F 的演示十六进制文件。
dsPIC33E NS demo.hex	dsPIC33E 的演示十六进制文件。
dsPIC33F NS demo.mcp	dsPIC33F 的演示 MPLAB® 项目文件。
dsPIC33E NS demo.mcp	dsPIC33E 的演示 MPLAB 项目文件。
cleanup.bat	用于删除中间编译文件的批处理文件脚本。
h\dsPICDEM1_1Plus.h	dsPICDEM™ 1.1 Plus 开发板程序的 C 头文件。
h\MEB.h	多媒体扩展板 (Multimedia Expansion Board, MEB) 程序的 C 头文件。
h\lcd.h	用于定义 LCD 驱动程序接口的 C 头文件。
h\main.h	main.c 中函数和宏的 C 头文件。
h\ns_api.h	用于定义噪声抑制库接口的 C 头文件。
h\Si3000CodecDrv.h	用于定义 Si3000 编解码器驱动程序接口的 C 头文件。
h\WM8731CodecDrv.h	用于定义 WM8731 编解码器驱动程序接口的 C 头文件。
libs\nslibv6_33F.a	dsPIC33F 的 dsPIC DSC 噪声抑制库归档文件。
libs\nslibv6_33E.a	dsPIC33E 的 dsPIC DSC 噪声抑制库归档文件。
src\dsPICDEM1_1Plus.c	包含 dsPICDEM 1.1 Plus 开发板程序的 C 源文件。
src\MEB.c	包含 MEB 程序的 C 源文件。
src\lcd_strings.c	用于 LCD 显示屏的 C 源文件。
src\main.c	包含主要语音处理程序的 C 源文件。
src\Si3000CodecDrv.c	包含 Si3000 编解码器驱动程序的 C 源文件。
src\WM8731CodecDRV.c	包含 WM8731 编解码器驱动程序的 C 源文件。
src\lcd.s	用于与 LCD 控制器通信的汇编程序。

2.2.2 doc 文件夹

此文件夹包含 dsPIC DSC 噪声抑制库的用户指南。要查看该文档，请双击文件名。还可以从 Microchip 网站 (www.microchip.com) 下载该用户指南。

2.2.3 h 文件夹

此文件夹包含噪声抑制库的包含文件，如表 2-2 所示。

表 2-2: 包含文件

文件名	说明
ns_api.h	包含噪声抑制库接口的包含文件。必须将此文件包含在应用程序中才能使用噪声抑制库。

2.2.4 lib 文件夹

此文件夹包含噪声抑制库的库归档文件，如表 2-3 所列。归档名称以目标器件系列 dsPIC33F 或 dsPIC33E 的名称为后缀。

表 2-3: 库文件

文件名	说明
nslibv6_33F.a	dsPIC33F 的噪声抑制库归档文件。必须将此文件包含在应用程序中才能使用噪声抑制库。
nslibv6_33E.a	dsPIC33E 的噪声抑制库归档文件。必须将此文件包含在应用程序中才能使用噪声抑制库。

2.2.5 wavefiles 文件夹

此文件夹包含可用于评估噪声抑制库在不同噪声环境下的性能的 Wave 文件。Wave 文件可在 PC 上使用媒体播放器进行回放，如开启重复播放功能可连续播放 Wave 文件。表 2-4 中提供了这些文件的说明。

表 2-4: WAVE 文件

文件名	说明
Beverage_plant_babble.wav	含有无法理解或无意义（模糊不清）噪声的语音段。
Critical_white.wav	含有白噪声的语音段。
Friends_70mph.wav	含有类似于在车厢内听到的噪声的语音段。
Late_morning_15mph.wav	含有类似于在车厢内听到的噪声的语音段。
Purple_tie_70mph.wav	含有类似于在车厢内听到的噪声的语音段。
Quench_tone.wav	含有单音噪声的语音段。
Taxi_white.wav	含有白噪声的语音段。
Twilight_babble.wav	含有无法理解或无意义（模糊不清）噪声的语音段。
Unusual_15mph.wav	含有类似于在车厢内听到的噪声的语音段。

注:

第 3 章 快速入门演示

本章介绍 dsPIC33F 和 dsPIC33E 器件系列的 dsPIC DSC 噪声抑制库快速入门演示。

3.1 dsPIC33F 器件系列的快速入门演示

本节涵盖以下主题：

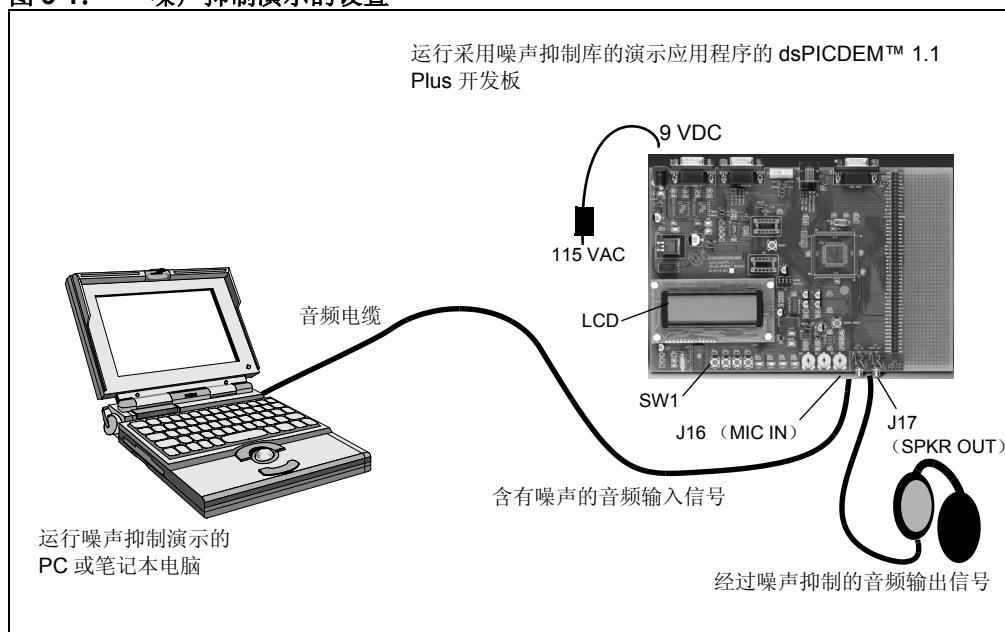
- [演示摘要](#)
- [演示设置](#)
- [演示步骤](#)
- [演示代码说明](#)

3.1.1 演示摘要

dsPIC DSC 噪声抑制库随附的基于 dsPIC33F 的演示应用程序演示了该函数库的功能。在演示设置（见 [图 3-1](#)）中，将 dsPICDEM™ 1.1 Plus 开发板配置为通过话筒输入端口接收含噪声的语音信号、抑制采样信号中的噪声以及通过扬声器输出端口播放已抑制噪声的信号的系统。板上 Si3000 编解码器用作话筒和扬声器接口。

PC 用于通过音频电缆将含噪声的语音信号从 PC 的扬声器输出端口驱动到 dsPICDEM 1.1 Plus 开发板上的 J16（MIC IN）。耳机或扬声器连接到 dsPICDEM 1.1 Plus 开发板上的 J17（SPKR OUT）。

图 3-1： 噪声抑制演示的设置



您可以使用随演示提供的 .wav 文件（位于安装目录的 wavefiles 文件夹中）作为含噪声的语音信号，也可以提供自己的信号。含噪声的输入信号由板上 Si3000 语音频带编解码器和 dsPIC DSC 器件的数据转换器接口（Data Converter Interface, DCI）模块捕捉。然后，dsPIC DSC 器件通过其 DCI 模块和板上 Si3000 编解码器播放经过处理（噪声抑制）的信号。

程序启动时初始化为关闭噪声抑制（通过熄灭 LED1 指示），并将 OFF 写到 LCD 屏幕上。噪声抑制关闭时，在耳机中听到的信号包含明显的噪声。

通过按下开关 SW1 使能噪声抑制。LED1 此时亮起，并且 LCD 上显示 ON。与原始信号相比，在耳机中听到的语音信号变为相对无噪声。使用开关 SW2 和 SW3 控制噪声抑制级别。请注意，提高噪声抑制级别也会增加信号的总衰减。

开启 PC 媒体播放器的重复播放功能以允许 .wav 文件连续播放。观察何时信号有噪声以及何时噪声被抑制。对多个 .wav 文件重复此过程。

注： 某些媒体播放器在每次重复播放 .wav 文件之前会停顿一下。如果想要避免这种情况，可使用声音编辑程序（如 Audacity）实现连续循环播放。Audacity 是免费的跨平台声音编辑器，可从 <http://audacity.sourceforge.net/> 下载。

3.1.2 演示设置

本演示应用程序用于在 dsPICDEM 1.1 Plus 开发板（未随软件许可提供）上运行。请按照以下几节所述的步骤设置演示。

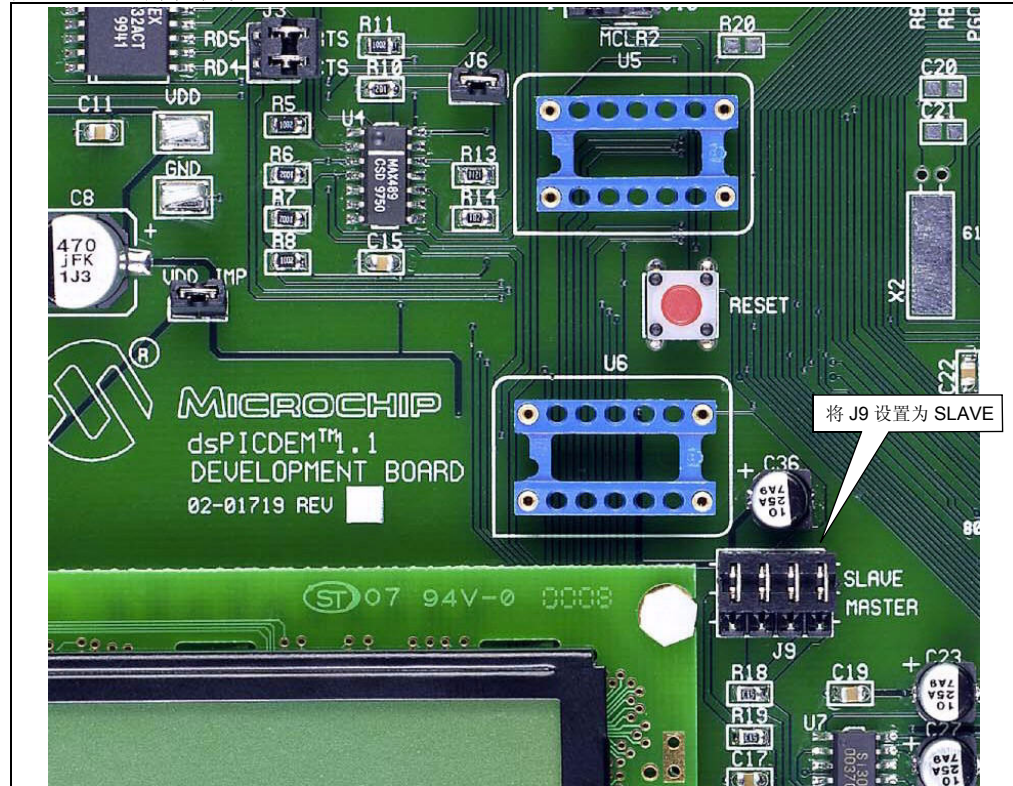
3.1.2.1 配置 dsPICDEM 1.1 PLUS 开发板

在上电之前需要按如下步骤配置开发板：

1. 将跳线 J9（与振荡器插座相邻）设置到 **SLAVE** 位置。此设置允许板上 Si3000 编解码器芯片以串行时钟从模式工作。
2. 在 PC 上的扬声器输出端口与 dsPICDEM 1.1 Plus 开发板上的“Microphone”插孔（J16）之间连接音频电缆。
3. 将耳机或扬声器连接到“SPKR OUT”插孔（J17）。
4. 在 PC（USB 电缆）与 dsPICDEM 1.1 Plus 开发板（RJ-11 电话线）之间连接 MPLAB ICD 3。
5. 连接 9V 电源为 dsPICDEM 1.1 Plus 开发板上电。

注： 可以使用 MPLAB® Real ICE™ 在线仿真器代替 MPLAB ICD 3。

图 3-2: 开发板设置



3.1.2.2 烧写 dsPIC DSC 器件

通过以下过程将噪声抑制演示加载到 dsPICDEM 1.1 Plus 开发板上的 dsPIC DSC 器件中。

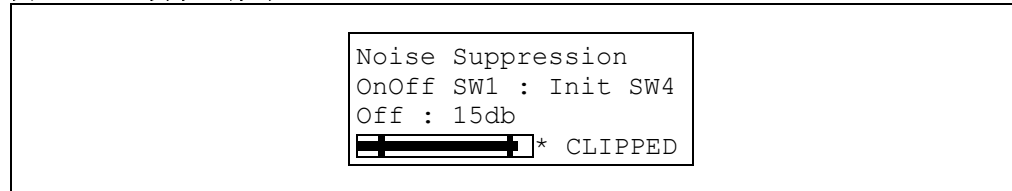
1. 在 PC 上启动 MPLAB IDE 并打开位于 demo 文件夹中的 dsPIC33F NS demo.mcp 项目。
有关使用 MPLAB IDE 的更多信息，请参见 “MPLAB® IDE User's Guide” (DS51025)。
2. 导入项目十六进制文件 (**File > Import > dsPIC33F NS demo.hex**) (文件 > 导入 > dsPIC33F NS demo.hex)。
3. 选择 **Programmer > Connect** (编程器 > 连接) 将 MPLAB ICD 3 链接到 dsPIC DSC 目标器件。Output (输出) 窗口确认 MPLAB ICD 3 已准备就绪。
4. 选择 **Programmer > Program** (编程器 > 编程)。Output 窗口显示下载过程并指示烧写已成功。
5. 程序加载完毕后，断开 MPLAB ICD 3 与开发板的连接 (从 MPLAB ICD 3 连接器中拔出电话线)。

3.1.3 演示步骤

演示应用程序烧写到器件中之后，演示即准备好运行。您可以使用提供的 .wav 文件（位于安装目录的 wavefiles 文件夹中）作为含噪声的语音信号，也可以提供自己的信号。含噪声的输入信号通过板上 Si3000 语音频带编解码器和 dsPIC DSC 器件的 DCI 模块进行采样。然后，dsPIC DSC 器件通过其 DCI 模块和板上 Si3000 编解码器播放经过处理（噪声抑制）的信号。

演示应用程序通过 LED 和 LCD 指示工作状态。当加载应用程序和初始化片上和片外外设时，将显示引导屏幕，然后引导屏幕会自动切换为运行时屏幕，如图 3-3 所示。

图 3-3: 演示运行时 LCD



运行时屏幕显示以下内容：

1. 算法的名称。
2. SW1 用于开启和关闭噪声抑制。SW4 用于重新初始化噪声抑制算法。
3. 算法的当前状态（关闭）和所选的降噪级别。
4. 表示输入级别的音量单位（volume unit, VU）计。频带显示了可接受的输入范围。* 指示噪声抑制已经将此帧解释为语音。当输入信号过大时，将显示 CLIPPED 一词。

程序启动时初始化为关闭噪声抑制（通过熄灭 LED1 指示），并在 LCD 上显示 OFF。噪声抑制关闭时，在耳机中听到的信号包含明显的噪声。

通过按下 SW1 使能噪声抑制。LED1 此时将亮起。LCD 上显示 ON，与原始信号相比，在耳机中听到的语音信号变为相对无噪声。

开启 PC 媒体播放器的重复播放功能以允许 .wav 文件连续播放。观察何时信号有噪声以及何时噪声被抑制。要使用不同类型和级别的噪声进行实验，可播放 wavefiles 文件夹中提供的多个 .wav 文件。

可按下 SW2 以 1 dB 为步长降低噪声级别（最低降至 0 dB）。可按下 SW3 以 1 dB 为步长增加噪声级别（最高增至 44 dB）。按下 SW4 可重新初始化噪声抑制。

3.1.4 演示代码说明

演示代码运行在 **dsPIC33F** 器件上，使用主振荡器作为时钟源，PLL 针对 **40 MIPS** 的工作进行配置。

main.c 文件包含演示应用程序的主函数。该文件在数据存储器中分配 **DCI** 数据缓冲所需的所有变量和数组，还为噪声抑制库函数分配所需的数据存储块。

主函数将调用噪声抑制库中的 **NS_init()** 函数，该函数用于将噪声抑制算法初始化为默认状态。

主函数还调用 **Si3000CodecInit()** 函数来初始化 **DCI** 模块、**Si3000** 编解码器和 **DCI** 中断。**DCI** 模块用作主器件并驱动串行时钟和帧同步线。**Si3000** 编解码器用作从器件。**DCI** 模块设置为多通道帧同步工作模式以及 **16** 位数据字，每帧 **16** 个数据字或时隙。在本演示中仅使用一个发送时隙和一个接收时隙。

随后，使用 **Si3000CodecStart()** 函数初始化 **Si3000** 编解码器。通过将 **dsPIC DSC** 器件的 **RF6** 引脚与 **Si3000** 的复位引脚连接，并将端口位 **RF6** 保持 **100** 个周期的低电平然后驱动为高电平来复位编解码器。编解码器的采样率配置为 **8 kHz**。话筒增益设置为 **10 dB**，接收增益设置为 **0 dB**。两个扬声器均设置为有效，发送增益设置为 **0 dB**。模拟衰减参数设置为 **0 dB**。初始化所有 **Si3000** 控制寄存器之后，引入延时以进行 **Si3000** 校准。最后，允许 **DCI** 中断。

针对整帧数据轮询编解码器驱动程序。当编解码器驱动程序指示整帧数据可用时，编解码器数据缓冲区的内容复制到 **signalIn** 数组中，并以 **signalIn** 为输入数据帧调用噪声抑制库的 **NS_apply** 函数。**signalIn** 数据缓冲区（也是 **NS_apply** 函数执行后的输出）通过耳机的扬声器输出播放。

通过在 **InitSPI** 函数中初始化串行外设接口（**Serial Peripheral Interface, SPI**）模块来实现在 **LCD** 上输出，**LCD** 驱动程序函数和 **LCD** 字符串定义分别位于 **lcd.s** 和 **lcd_strings.c** 文件中。

要在开启和关闭噪声抑制之间切换，请允许 **SW1** 的外部中断。在主循环中，读取 **applyNS** 的值并将其传递到 **NS_apply** 作为使能标志。如果 **applyNS** 为 **0**，则仍然调用噪声抑制库，但不更改输入 / 输出缓冲区。这将使噪声抑制库保持对噪声变化的自适应，即使其并未使能。

3.2 dsPIC33E 器件系列的快速入门演示

本节涵盖以下主题：

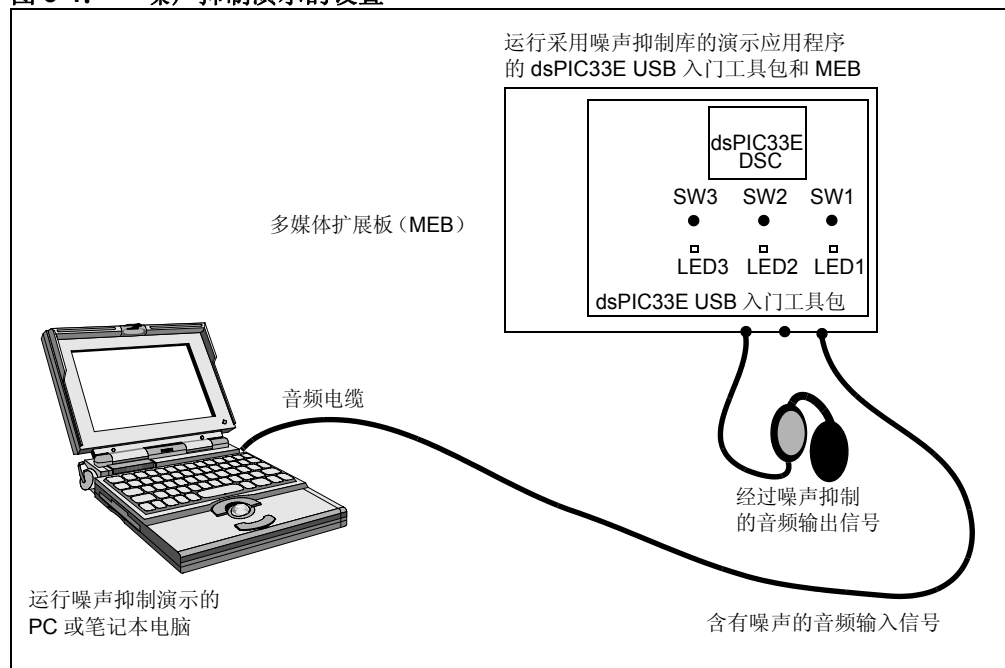
- [演示摘要](#)
- [演示设置](#)
- [演示步骤](#)
- [演示代码说明](#)

3.2.1 演示摘要

dsPIC DSC 噪声抑制库随附了专为 dsPIC33E 系列器件创建的演示应用程序。在演示设置（见[图 3-4](#)）中，将与 dsPIC33E USB 入门工具包配合使用的多媒体扩展板（MEB）配置为通过话筒输入端口接收含噪声的语音信号、抑制采样信号中的噪声以及通过扬声器输出端口输出已抑制噪声的信号的系统。板上 WM8731 编解码器用作话筒和扬声器接口。

PC 用于通过音频电缆将含噪声的语音信号从 PC 的扬声器输出端口驱动到 MEB 的话筒输入。耳机或扬声器连接到 MEB 的扬声器输出。板上 WM8731 编解码器用作话筒和扬声器接口。

图 3-4： 噪声抑制演示的设置



您可以使用随演示提供的 .wav 文件（位于安装目录的 wavefiles 文件夹中）作为含噪声的语音信号，也可以提供自己的信号。含噪声的输入信号由板上 WM8731 语音音频编解码器和 dsPIC DSC 器件的 DCI 模块捕捉。然后，dsPIC DSC 器件通过其 DCI 模块和板上 WM8731 编解码器输出经过处理（噪声抑制）的信号。

程序启动时初始化为关闭噪声抑制（通过熄灭 dsPIC33E USB 入门工具包上的 LED3 指示）。噪声抑制关闭时，在耳机中听到的信号包含明显的噪声。

通过按下 dsPIC33E USB 入门工具包上的开关 SW1 使能噪声抑制。此时 LED3 亮起，与原始信号相比，在耳机中听到的语音信号变为相对无噪声。使用开关 SW2 和 SW3 控制噪声抑制级别。请注意，提高噪声抑制级别也会增加信号的总衰减。

开启 PC 媒体播放器的重复播放功能以允许 .wav 文件连续播放。然后，观察何时信号有噪声以及何时噪声被抑制。对多个 .wav 文件重复此过程。

注： 某些媒体播放器在每次重复播放 .wav 文件之前会停顿一下。如果想要避免这种情况，可使用声音编辑程序（如 Audacity）实现连续循环播放。Audacity 是免费的跨平台声音编辑器，可从 <http://audacity.sourceforge.net/> 下载。

3.2.2 演示设置

本演示应用程序用于在 MEB 和 dsPIC33E USB 入门工具包（未随软件提供）上运行。请按照以下几节所述的步骤设置演示。

3.2.2.1 配置 MEB 和 dsPIC33E USB 入门工具包

在上电之前需要按如下步骤配置多媒体扩展板：

1. 将 dsPIC33E USB 入门工具包插入到 MEB 上的入门工具包连接器中。
2. 在 PC 上的扬声器输出端口与 MEB 上的话筒插孔（J7）之间连接音频电缆。
3. 连接 MEB 的耳机插孔（J8）。
4. 使用随入门工具包提供的 USB A 转 Mini B 电缆将 dsPIC33E USB 入门工具包连接到 PC。

3.2.2.2 烧写 dsPIC DSC 器件

通过以下过程将噪声抑制演示加载到 dsPIC DSC 器件中：

1. 在 PC 上启动 MPLAB IDE 并打开位于 demo 文件夹中的 dsPIC33E NS demo.mcp 项目。
有关使用 MPLAB IDE 的更多信息，请参见 “MPLAB® IDE User's Guide” (DS51025)。
2. 通过选择 **File > Import > dsPIC33E NS demo.hex** 来导入项目十六进制文件。
3. 选择板上的入门工具包作为编程器，然后选择 **Programmer > Connect** 链接到 dsPIC DSC 目标器件。Output 窗口确认目标器件已准备就绪。
4. 选择 **Programmer > Program**。Output 窗口显示下载过程并指示烧写已成功。

注： 烧写后，拔下 USB 电缆并重新连接到入门工具包，以确保可以重新配置 WM8731 音频编解码器。

3.2.3 演示步骤

演示应用程序烧写到器件中之后，演示即准备好运行。您可以使用提供的 .wav 文件（位于安装目录的 wavefiles 文件夹中）作为含噪声的语音信号，也可以提供自己的信号。含噪声的输入信号通过板上 WM8731 语音频带编解码器和 dsPIC DSC 器件的 DCI 模块进行采样。然后，dsPIC DSC 器件通过其 DCI 模块和板上 WM8731 编解码器播放经过处理（噪声抑制）的信号。

程序启动时初始化为关闭噪声抑制（通过熄灭 dsPIC33E USB 入门工具包上的 LED3 指示）。噪声抑制关闭时，在耳机中听到的信号包含明显的噪声。

通过按下 dsPIC33E USB 入门工具包上的开关 SW1 使能噪声抑制。LED3 此时将亮起。与原始信号相比，在耳机中听到的语音信号变为相对无噪声。

开启 PC 媒体播放器的重复播放功能以允许 .wav 文件连续播放。观察何时信号有噪声以及何时噪声被抑制。要使用不同类型和级别的噪声进行实验，可播放 wavefiles 文件夹中提供的多个 .wav 文件。

可按住 SW2 几秒钟以 1 dB 为步长降低噪声级别（最低降至 0 dB）。可按住 SW3 几秒钟以 1 dB 为步长增加噪声级别（最高增至 44 dB）。

可通过按下 MEB 的 LCD 侧上的开关 S1 来重新初始化噪声抑制。

3.2.4 演示代码说明

演示代码运行在 dsPIC33E 器件上，使用主振荡器作为时钟源，PLL 针对 40 MIPS 的工作进行配置。

main.c 文件包含演示应用程序的主函数。该文件 (main.c) 在数据存储器中分配 DCI 数据缓冲所需的所有变量和数组，还为噪声抑制库函数分配所需的数据存储块。

主函数将调用噪声抑制库中的 NS_init() 函数，该函数用于将噪声抑制算法初始化为默认状态。

主函数还调用 WM8731Init() 函数来初始化 DCI 模块、WM8731 编解码器和 DCI 中断。WM8731 编解码器用作主器件并驱动串行时钟和帧同步线。DCI 模块设置为多通道帧同步工作模式以及 16 位数据字，每帧两个数据字或时隙，即，本演示中使用两个发送时隙和两个接收时隙。

随后，使用 WM8731Start() 函数使能 DCI 模块和 I²C 模块。编解码器的采样率配置为 8 kHz。

针对整帧数据轮询编解码器驱动程序。当编解码器驱动程序指示整帧数据可用时，编解码器数据缓冲区的内容复制到 signalIn 数组中，并以 signalIn 为输入数据帧调用噪声抑制库的 NS_apply 函数。signalIn 数据缓冲区（也是 NS_apply 函数执行后的输出）通过耳机的扬声器输出播放。

在主循环中，读取 applyNS 的值并将其传递到 NS_apply 作为使能标志。如果 applyNS 为 0，则仍然调用噪声抑制库，但不更改输入 / 输出缓冲区。这将使噪声抑制库保持对噪声变化的自适应，即使其并未使能。

第 4 章 应用程序编程接口（API）

本章介绍 dsPIC DSC 噪声抑制库中提供的应用程序编程接口（API）。本章涵盖以下主题：

- [向应用程序添加噪声抑制库](#)
- [函数库的使用](#)
- [资源需求](#)
- [噪声抑制库 API 函数](#)
- [应用技巧](#)

4.1 向应用程序添加噪声抑制库

要在应用程序中使用噪声抑制库，必须将库归档添加到应用程序项目工作区，还必须将 `ns_api.h` 头文件包含在应用程序代码中。此文件可从 `h` 文件夹（位于安装目录中）复制到应用程序项目文件夹。

使用以下步骤将该函数库添加到应用程序：

1. 在应用程序 MPLAB 工作区中，右键单击项目窗口中的 **Library Files**（库文件），然后选择 **Add files**（添加文件）。
2. 浏览到 `nslibv6_33F.a` 文件或 `nslibv6_33E.a` 文件的位置，两个文件都位于安装目录的 `libs` 文件夹中。
3. 选择所需文件，然后单击 **Open**（打开）。
4. 函数库即添加到应用程序中。验证函数库归档是否显示在 MPLAB 项目中。

4.2 函数库的使用

噪声抑制库设计为可用于可重入环境。此特性使该算法可处理多个独立音频通道，每个通道均具有自己的设置和参数。

要使用噪声抑制库，至少需要执行以下五个编码步骤：

1. 为输入 / 输出数据数组分配存储器。数组大小应该为 `NS_FRAME`。数组应放入 `X` 存储器并与 4 字节边界对齐。
2. 为噪声抑制算法状态保持器分配存储器：此存储器是 `X` 存储器中与 4 字节地址边界对齐的整型数组。`NS_XSTATE_MEM_SIZE_INT` 宏指定此数组的大小。每个要处理的音频通道都需要有自己的状态保持器。
3. 为噪声抑制算法 `X` 和 `Y` 临时存储器分配存储器：`X` 临时存储器是 `X` 存储器中与 4 字节地址边界对齐的整型数组。`Y` 临时存储器是 `Y` 存储器中与 512 字节地址边界对齐的整型数组。多个音频通道可共享临时存储器。

注： 在某些 dsPIC33E 器件中，`Y` 存储器位于扩展数据空间（Extended Data Space, EDS）中。在这种情况下，`Y` 临时存储器数组必须标记有 `__eds__` 关键字并被分配 `eds` 属性。

4. 初始化每个音频通道的噪声抑制算法状态：使用 `NS_init()` 函数初始化每个音频通道的噪声抑制状态。
5. 对音频帧应用噪声抑制算法：使用 `NS_apply()` 函数对音频帧执行噪声抑制。即使某个帧不需要使用噪声抑制算法进行处理，也仍然应该调用该函数，只是将使能参数设置为 `NS_FALSE`。这样噪声抑制算法会继续自适应音频帧中的噪声。音频帧则不受影响。

例 4-1 提供了展示这五个步骤的代码示例。

例 4-1: 函数库使用示例

```
/* 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 资源需求

在 dsPIC DSC 器件上运行噪声抑制库时，需要以下计算资源。

表 4-1: dsPIC33F 的程序存储器使用情况

类型	大小 (字节)	段
程序存储器中的代码	6675 (dsPIC30F/dsPIC33F) 6414 (dsPIC33E)	.libns
程序存储器中的表	1383 (dsPIC30F/dsPIC33F) 0 (dsPIC33E)	.const
程序存储器合计	8058 (dsPIC30F/dsPIC33F) 6414 (dsPIC33E)	—

表 4-2: 数据存储器使用情况

函数	大小 (字节)	对齐	段
nsStateMemX	768	4	X 数据存储器
scratchMemX	100	4	X 数据存储器
scratchMemY	256	512	Y 数据存储器
signalIn	160	4	X 数据存储器
数据存储器中的表	0 (dsPIC30F/dsPIC33F) 922 (dsPIC33E)	4	X 数据存储器
数据存储器合计	1284 (dsPIC30F/dsPIC33F) 2206 (dsPIC33E)	—	—

表 4-3: 估算的动态存储器使用情况

段	大小 (字节)
堆	0
栈	< 300

表 4-4: 计算速度

函数	MIPS	典型调用频率
NS_init()	< 0.5	一次
NS_apply()	3.6	10 ms
所有其他函数	最小	根据需要

4.3.1 数据格式

噪声抑制库的输入数据可以是 10 位、12 位或 16 位线性脉冲编码调制 (Pulse Code Modulation, PCM) 数据。噪声抑制算法会针对使用的数据格式进行自动调整。

4.4 噪声抑制库 API 函数

本节将列出并介绍 dsPIC DSC 噪声抑制库中提供的 API 函数。下面列出了这些函数以及各自的详细说明。

- `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

说明

初始化噪声抑制算法。

包含文件

ns_api.h

原型

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

参数

ptrStateX	指向本噪声抑制实例的状态存储器的指针
xScratchMem	指向噪声抑制算法在 X RAM 中使用的临时存储区的指针
yScratchMem	指向噪声抑制算法在 Y RAM 中使用的临时存储区的指针

返回值

无。

备注

无。

代码示例

```
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

说明

更改噪声抑制算法使用的 X 存储器临时缓冲区。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

xScratchMem 指向噪声抑制算法使用的新的 X 临时存储器的指针

返回值

无。

备注

重定位 X 临时存储器之后，噪声抑制库不再使用旧的临时存储器。

代码示例

```
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);
噪声抑制算法正在将 nsScratchX 用作临时存储器。
.
.
.
NS_relocateXScratchMem(nsStateMemX, nsScratchX2);
噪声抑制算法正在将 nsScratchX2 用作临时存储器。
```

NS_relocateYScratchMem

说明

更改噪声抑制算法使用的 Y 存储器临时缓冲区。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

yScratchMem 指向噪声抑制算法使用的新的 Y 临时存储器的指针

返回值

无。

备注

重定位 Y 临时存储器之后，噪声抑制库不再使用旧的临时存储器。

代码示例

```
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);
噪声抑制算法正在将 nsScratchY 用作 Y 存储器临时缓冲区。
.
.
.
NS_relocateYScratchMem(nsStateMemX, nsScratchY2);
噪声抑制算法正在将 nsScratchY2 用作临时存储器。
```

NS_apply

说明

对当前数据帧进行降噪。

包含文件

ns_api.h

原型

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

参数

ptrStateX	指向本噪声抑制实例的状态存储器的指针
SignalIn	指向大小为 NS_FRAME 的输入 / 输出缓冲区的指针
Enable	指示此缓冲区是否需要噪声抑制的标志（ NS_TRUE / NS_FALSE ）

返回值

无。

备注

噪声抑制“就地”处理数据，这意味着输出传递回输入缓冲区。将 **Enable** 设置为 **NS_FALSE** 会返回未处理的数据缓冲区，但噪声抑制算法仍会针对数据自适应。

代码示例

```
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

说明

设置降噪级别（以 dB 为单位）。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

leveldB 介于 0 和 44 之间的整型值，表示所需的降噪级别（以 dB 为单位）

返回值

无。

备注

所需的降噪级别用白噪声衡量。其他类型的噪声将得出不同的降噪级别。

代码示例

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

将算法实例 nsStateMemX 所需的降噪级别设置为 18 dB。

NS_getNoiseReduction

说明

返回当前的降噪值（以 dB 为单位）。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

返回值

算法实例 nsStateMemX 当前所需的整型降噪级别（以 dB 为单位）。

备注

无。

代码示例

```
int nrLevel;  
nrLevel = NS_getNoiseReduction(nsStateMemX);  
nrLevel 包含算法实例 nsStateMemX 所需的降噪级别。
```

NS_setVadHangover

说明

设置 VAD 拖尾延迟保护（hangover）值。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

value 从 1 到 100 的整型值，表示所需的 VAD 拖尾延迟保护值

返回值

无。

备注

VAD 拖尾延迟保护用于调整 VAD 期间的迟滞。值越大，VAD 保持有效的时间越长。将 VAD 拖尾延迟保护设置为 1 将禁止迟滞。选择允许范围（1 至 100）以外的值将保留先前的设置。

代码示例

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

将算法实例 nsStateMemX 所需的 VAD 拖尾延迟保护设置为 10 帧（表示 100 ms 的数据）。

NS_getVadHangover

说明

返回当前的 VAD 拖尾延迟保护值。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

返回值

VAD 拖尾延迟保护值。

备注

无。

代码示例

```
int vadHangover;  
vadHangover = NS_getVadHangover(nsStateMemX);  
vadHangover 包含为算法实例 nsStateMemX 设置的 VAD 拖尾延迟保护值。
```

NS_setNoiseFactor

说明

设置噪声系数。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

value 介于 1 和 15 之间的整型值，表示所需的噪声系数

返回值

无。

备注

设置 VAD 使用的噪声系数以从噪声中确定语音。值越高，标记为噪声的数据帧就越多。合法值在 1 至 15 的范围内（包括 1 和 15）。

代码示例

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

将算法实例 nsStateMemX 的噪声系数设置为 3。

NS_getNoiseFactor

说明

返回当前的噪声系数。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

返回值

噪声系数。

备注

无。

代码示例

```
int noiseFactor;  
noiseFactor = NS_getNoiseFactor(nsStateMemX);  
noiseFactor 包含为算法实例 nsStateMemX 设置的噪声系数值。
```

NS_setNoiseEstimationFactor

说明

设置噪声估算系数。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

level NS_SLOW、NS_NORMAL 或 NS_FAST

返回值

无。

备注

此系数用于确定噪声抑制算法在检测到噪声帧后更新其噪声估算的速度。

代码示例

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

将算法实例 nsStateMemX 的噪声系数设置为正常速度。

NS_getNoiseEstimationFactor

说明

返回当前的噪声估算系数。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

返回值

噪声估算系数。

备注

无。

代码示例

```
int noiseEstFactor;  
noiseEstFactor = NS_getNoiseEstimationFactor(nsStateMemX);  
noiseEstFactor包含为算法实例nsStateMemX设置的噪声估算系数值，为NS_SLOW、  
NS_NORMAL 或 NS_FAST。
```

NS_setNoiseFloor

说明

设置基底噪声，低于基底噪声时噪声抑制算法不会更新。

包含文件

ns_api.h

原型

void NS_setNoiseFloor(int* ptrStateX, long floor);

参数

- ptrStateX 指向本噪声抑制实例的状态存储器的指针
- floor 表示背景噪声级别的长整型值，低于该值时噪声抑制算法不会更新

返回值

无。

备注

系统噪声表示由应用硬件产生的噪声。不包括输入信号中包含的噪声。例如，如果硬件产生的噪声通过实验确定为 -60 dBm0 或 -66 dBov，则按公式 4-1 中所示计算十六进制形式的基底噪声值。

公式 4-1: 基底噪声计算示例

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

注: 粗体数字表示需要确定的值。

在本例中，将基底噪声设置为 **0x15123**（该值提供给函数值），使噪声抑制算法在噪声能量低于 -60 dBm0 时停止自适应。要禁用此功能，则将值设置为零。

代码示例

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

将算法实例 nsStateMemX 的基底噪声设置为 **0x1D1917**。

NS_getNoiseFloor

说明

返回当前的基底噪声。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

返回值

无。

备注

返回当前的基底噪声值。

代码示例

```
long noiseFloor;  
noiseFloor = NS_getNoiseFloor(nsStateMemX);  
noiseFloor 包含为算法实例 nsStateMemX 设置的基底噪声值。
```

NS_getIsSpeech

说明

返回一个标志指示噪声抑制算法认为当前帧是语音还是噪声。

包含文件

ns_api.h

原型

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

参数

ptrStateX 指向本噪声抑制实例的状态存储器的指针

返回值

整型标志 NS_TRUE 或 NS_FALSE。

备注

这是输入信号中是否存在语音的指示。此标志受 VAD 拖尾延迟保护、基底噪声和功率因数影响。

代码示例

```
int speech;  
speech = NS_getIsSpeech(nsStateMemX);  
speech 是表示噪声抑制算法是否认为当前帧包含语音的标志。
```

NS_estimateSNR

说明

估算信噪比（Signal-to-Noise Ratio， SNR）。

包含文件

ns_api.h

原型

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

参数

preNS 指向应用噪声抑制算法之前的数据缓冲区的指针

postNS 指向应用噪声抑制算法之后的数据缓冲区的指针

返回值

SNR 的估算值。

备注

这是当前数据帧 SNR 的粗略估算值。

代码示例

```
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 包含 SNR 的估算值。

NS_TRUE

说明

用于向噪声抑制算法指示真（true）。

值

1

NS_FALSE

说明

用于向噪声抑制算法指示假（false）。

值

0

NS_SLOW

说明

与 NS_setNoiseEstimationFactor 和 NS_getNoiseEstimationFactor 配合使用，用于调整对噪声的自适应速度。

值

100

NS_NORMAL

说明

与 NS_setNoiseEstimationFactor 和 NS_getNoiseEstimationFactor 配合使用，用于调整对噪声的自适应速度。

值

200

NS_FAST

说明

与 NS_setNoiseEstimationFactor 和 NS_getNoiseEstimationFactor 配合使用，用于调整对噪声的自适应速度。

值

300

NS_FRAME

说明

处理的输入缓冲区的大小。

值

80

NS_XSTATE_MEM_SIZE_INT

说明

X 状态存储器所需的存储器单元的大小（整型）。

值

383

NS_XSCRATCH_MEM_SIZE_INT

说明

X 临时存储器所需的存储器单元的大小（整型）。

值

50

NS_YSCRATCH_MEM_SIZE_INT

说明

Y 临时存储器所需的存储器单元的大小（整型）。

值

128

NS_NRLEVELDEFAULT

说明

默认降噪值（以 dB 为单位）。

值

15

NS_VADHANGOVERDEFAULT

说明

默认 VAD 拖尾延迟保护值。

值

6

NS_NOISEFLOORDEFAULT

说明

默认基底噪声值。

值

0x1D1917

NS_POWERFACTORDEFAULT

说明

噪声功率因数的默认值。

值

2

4.5 应用技巧

尽管噪声抑制算法会尽可能抑制信号中的噪声，但还可以通过适当选择参数来优化其性能。一般来说，我们鼓励使用此函数库进行实验。

下面是几个有关影响算法性能的技巧：

1. 通常认为用于测试音频系统的最佳输入信号级别在 **-10 dBm0** 和 **-30 dBm0** 之间。如果数字输入语音级别的峰值达到满量程的四分之三，则说明充分利用了可用精度；高于此值的语音级别会产生限幅的风险。
2. 选择最适合应用的降噪级别范围。非常高的噪声抑制级别不仅会抑制噪声，而且会在一定程度上降低语音级别。
3. **VAD** 拖尾延迟保护可用于调整噪声抑制算法的自适应速度。较大的 **VAD** 拖尾延迟保护值会增加算法对噪声级别变化的自适应时间。较小的 **VAD** 拖尾延迟保护值会使噪声抑制算法将更多的输入帧当作噪声帧，因而自适应速度更快。
4. 如果噪声级别高于语音级别，则可能需要增大噪声系数值。
5. 噪声估算系数用于确定噪声抑制算法在检测到噪声帧后更新噪声估算的速度。如果已知噪声环境本身比较稳定，可考虑将此参数设置为 **NS_SLOW**。

无论语音信号中包含的噪声如何，应用硬件都会像所有电路一样产生少量噪声，这种噪声在断开话筒时也仍然存在。在这种情况下，噪声抑制算法应冻结自适应，以便在重新连接话筒后像平常一样继续工作。但是，如果发现噪声抑制有偏差，则说明默认基底噪声过低。要解决此问题，请使用 **NS_setNoiseFloor** 函数增加基底噪声（可能需要一些测试）。

注:

索引

A

API.....	5, 25
API 函数	29
NS_apply.....	33
NS_estimateSNR.....	40, 45
NS_FALSE.....	46
NS_FAST.....	47
NS_FRAME.....	47
NS_getIsSpeech.....	44
NS_getNoiseEstimationFactor	41
NS_getNoiseFactor.....	39
NS_getNoiseFloor.....	43
NS_getNoiseReduction	35
NS_getVadHangover.....	37
NS_init.....	30
NS_NOISEFLOORDEFAULT	48
NS_NORMAL	46
NS_NRLEVELDEFAULT.....	48
NS_POWERFACTORDEFAULT.....	48
NS_relocateXScratchMem.....	31
NS_relocateYScratchMem.....	32
NS_setNoiseEstimationFactor	40
NS_setNoiseFactor.....	38
NS_setNoiseFloor.....	42
NS_setNoiseReduction	34
NS_setVadHangover.....	36
NS_SLOW.....	46
NS_TRUE.....	46
NS_VADHANGOVERDEFAULT	48
NS_XSCRATCH_MEM_SIZE_INT.....	47
NS_XSTATE_MEM_SIZE_INT.....	47
NS_YSCRATCH_MEM_SIZE_INT.....	48

安装

安装函数库.....	13
------------	----

B

保修登记.....	7
-----------	---

H

互联网地址.....	8
------------	---

K

客户通知服务.....	8
客户支持.....	8
库文件	
demo 文件夹.....	13
doc 文件夹.....	14
h 文件夹	14
wavefiles 文件夹.....	15
快速傅立叶变换 (FFT)	11

M

Microchip 互联网网站	8
MPLAB IDE 用户指南.....	7

S

算法概述	11
------------	----

W

WWW 地址	8
文档	
编排	5
约定	6

Y

演示	
板配置	18, 23
器件烧写	19, 23
设置	17, 22
运行演示	20, 24
语音活动检测 (VAD)	11

Z

主机系统要求	12
资源要求	
程序存储器使用情况	28
估算的动态存储器使用情况.....	28
计算速度	28
数据存储器使用情况	28
数据格式	28

全球销售及服务网点

美洲

公司总部 **Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277

技术支持:

<http://www.microchip.com/support>

网址: www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 1-408-961-6444
Fax: 1-408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario, Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 **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

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 杭州
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

中国 - 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 - 高雄
Tel: 886-7-536-4818
Fax: 886-7-330-9305

台湾地区 - 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

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

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

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

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

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

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

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

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

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

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

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

欧洲

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

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

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

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

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

荷兰 Netherlands - Druenen
Tel: 31-416-690399
Fax: 31-416-690340

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

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