PAC193X
Microsoft® Windows® 10驱动程序
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最安全的产品之一。
- 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。

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生产方面的质量体系也已通过了 ISO 9001:2000认证。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

ISBN：978-1-5224-2653-0

目录

前言

客户通知

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“DSXXXXXXXXA_CN”，其中 “XXXXXXXX”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

本章包含使用PAC193X Microsoft® Windows® 10驱动程序前需要了解的一般信息。本章讨论的内容包括：

- 文档编排
- 本指南中使用的约定
- 推荐读物
- Microchip网站
- 客户支持
- 文档版本历史

文档编排

本文档介绍用户为访问器件报告的电量测量结果而需要了解的PAC193X Microsoft Windows 10驱动程序的技术内容。手册编排如下：

- **第1章“产品概述”** ——有关PAC193X Microsoft Windows 10驱动程序的重要信息。
- **第2章“驱动程序安装”**——包含有关安装和启动PAC193X Microsoft Windows 10驱动程序的说明。
- **第3章“器件实例化和初始化”**——有关PAC193X Microsoft Windows 10驱动程序初始配置的重要信息。
- **第4章“PAC193X器件接口”**——有关PAC193X Microsoft Windows 10驱动程序接口的信息。
- **第5章“PAC193X驱动程序特性”**——有关PAC193X Microsoft Windows 10驱动程序特性的信息。
- **附录A“驱动程序版本历史”** ——包含驱动程序故障修复和更改。
- **附录B“PAC193X器件控制接口”**——包含EMI和控制接口数据结构的C定义。

本指南中使用的约定

本手册使用以下文档约定：

文档约定

说明	表示	示例
Arial字体：		
斜体字	参考书目	*MPLAB® IDE User's Guide*
	需强调的文字	……为*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	*File>Save*
粗体字	对话框按钮	单击**OK**
	选项卡	单击**Power**选项卡
N‘Rnnnn	verilog格式的数字，其中N为总位数，R为基数，n为其中一位。	4‘b0010，2‘hF1
尖括号< >括起的文字	键盘上的按键	按下<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’`
斜体Courier New	可变参数	*file*.o，其中*file*可以是任一有效文件名
方括号 [ ]	可选参数	`mcc18 [选项] file [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) { ... }`

推荐读物

本用户指南介绍如何使用PAC193X Microsoft Windows 10驱动程序。下面列出其他有用文档。以下Microchip文档作为补充参考资源提供和推荐：

- **PAC193X应用笔记——*“PAC193X Integration Notes for Microsoft® Windows® 10 Driver Support”*（DS00002534A）**
- **PAC1934数据手册——*“PAC1934 Power/Energy Monitor with Accumulator Data Sheet”*（DS20005850A）**
- **PAC1934用户指南——《PAC1934评估板用户指南》（DS50002673A_CN）**

MICROCHIP网站

Microchip网站（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存档软件
- **一般技术支持**——常见问题（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

客户支持

Microchip产品的用户可通过以下渠道获得帮助：

- 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。

可通过http://www.microchip.com/support获得网上技术支持。

文档版本历史

版本A（2017年9月）

- 本文档的初始版本。
- 本文档中的信息适用于以下PAC193X驱动程序版本：1.0、1.1和1.2。

注：

第1章 产品概述

1.1 简介

本文档介绍用户为访问PAC193X器件报告的电量测量结果而需要了解的PAC193X Microsoft® Windows® 10驱动程序的技术内容。

从Microsoft Windows 10操作系统开始，可以监测笔记本电脑或其他便携式设备的各个组件的能耗，以确定能够最大限度延长电池寿命的最佳配置选项。Microsoft Windows 10中的电量估算引擎（Energy Estimation Engine，E3）服务采用估算算法或通过连接到系统各电源轨的计量IC所测得的实时电量监测能耗。

计量IC器件驱动程序必须通过标准软件接口即电量计量接口（Energy Metering Interface，EMI）报告电量测量结果。E3服务或用户应用程序可以通过调用一组标准EMI IOCTL服务访问电量数据。

表1-1： 术语表与首字母缩略词

术语	说明
ACPI	高级配置与电源接口
ASL	ACPI源语言
BIOS	基本输入/输出系统
CPU	中央处理单元
E3	电量估算引擎
EC	嵌入式控制器
EMI	电量计量接口
强行关断	通过改写ACPI电源按钮将系统G0/S0临时转换到G2/S5
GUID	全局唯一标识符
I^2C	内部集成电路
IC	集成电路
IOCTL	I/O控制（I/O control，IOCTL）*DeviceIoControl*系统调用是一个可以将控制代码发送到器件的通用接口。各控制代码分别代表一个器件驱动程序要执行的操作
OEM	原始设备制造商
OS	操作系统
SMBUS	系统管理总线
SPB	简单外设总线
SoC	片上系统
SpbCx	SPB框架扩展
UUID	通用唯一标识符
WDF	Windows驱动程序基础

注：

第2章 驱动程序安装

2.1 先决条件

在将驱动程序安装到目标系统上之前，必须先解决以下驱动程序依赖项：

- 在主机系统中集成PAC193X器件
- I²C总线控制器的设备驱动程序符合SPB框架扩展（SpbCx）标准

2.1.1 在主机系统中集成PAC193X

2.1.1.1 ACPI ASL

请参见PAC193X应用笔记中列出的说明，并对ACPI ASL进行必要的更改，确保Microsoft® Windows®设备管理器枚举PAC193X器件。

2.1.1.2 SLOW引脚

PAC器件的VDD引脚应连接到一个不会因系统待机或掉电序列而关闭的系统电源轨。SLOW引脚应连接到一个在系统电源关闭后驱动为高电平的引脚。

注意

系统正常工作期间，请勿使用 SLOW 引脚更改器件采样率。系统在 S0 状态下运行时，Windows 驱动程序不监测 SLOW 引脚转换。

PAC193X Microsoft Windows 10驱动程序支持以下SLOW引脚用例：

- 确保PAC器件在系统崩溃后被操作员或其他故障安全机制强制关闭时切换到慢速采样模式的故障安全机制
- 重新打开系统电源后，由高到低的缓慢转换信息保存到系统关闭时累加数据的器件可读寄存器中

2.1.2 符合SpbCx标准的I²C总线控制器的设备驱动程序

为访问I²C总线并与PAC器件进行通信，PAC193X Microsoft Windows 10驱动程序使用由Windows SpbCx框架实现的标准接口与协议。这意味着I²C总线控制器的Microsoft Windows设备驱动程序也必须符合SpbCx框架标准。通常，该驱动程序由I²C控制器制造商创建，并通过Microsoft Windows安装或Microsoft Windows更新提供。

2.2 驱动程序安装

PAC193X Microsoft Windows 10驱动程序有多种安装选项：

- 通过Microsoft Windows更新进行安装。
- 使用从Microchip网站下载的设备管理器和驱动程序包进行安装。
- 运行驱动程序的安装应用程序：PAC193xDriverInstaller_x86或PAC193xDriverInstaller_x64，包含在从Microchip网站下载的驱动程序包中。
- 使用DevCon实用程序和从Microchip网站下载的驱动程序包。

2.2.1 使用Microsoft Windows更新或设备管理器进行安装

有关通过Microsoft Windows更新或设备管理器安装驱动程序的详细信息，请参见以下Microsoft文章：《如何在Microsoft Windows 10中安装和更新驱动程序》。

2.2.2 驱动程序的安装应用程序

打开PAC193xDriverInstaller_x86和PAC193xDriverInstaller_x64驱动程序的安装应用程序，连续单击确定直到显示“Microchip最终用户许可协议”（End User License Agreement，EULA）对话框。接受条款和条件后，驱动程序将自动安装完成。

> **注：** 根据目标系统上安装的操作系统，选择32位（PAC193xDriverInstaller_x86）或64位（PAC193xDriverInstaller_x64）安装应用程序。

2.2.3 使用DevCon实用程序进行安装

DevCon实用程序随Windows驱动程序工具包（Windows Driver Kit，WDK）一同提供。

1. 根据目标系统的操作系统（32位或64位），从驱动程序包中提取下列文件：
   - PAC193x.sys
   - PAC193x.inf
   - PAC193x.cat
2. 将驱动程序文件和DevCon实用程序复制到目标系统上。
3. 用提升命令提示符打开，将目录更改为驱动程序文件所复制到的文件夹，然后执行命令：

```
devcon.exe update pac193x.inf ACPI\MCHP1930
```

2.2.3.1 验证安装

要验证PAC193X Microsoft Windows 10驱动程序安装是否成功完成，请打开设备管理器，然后检查“Microchip Energy Metering Devices”（Microchip电量计量器件）类别下面是否列出PAC193X器件以及其是否报告为正常工作。请参见图2-1。

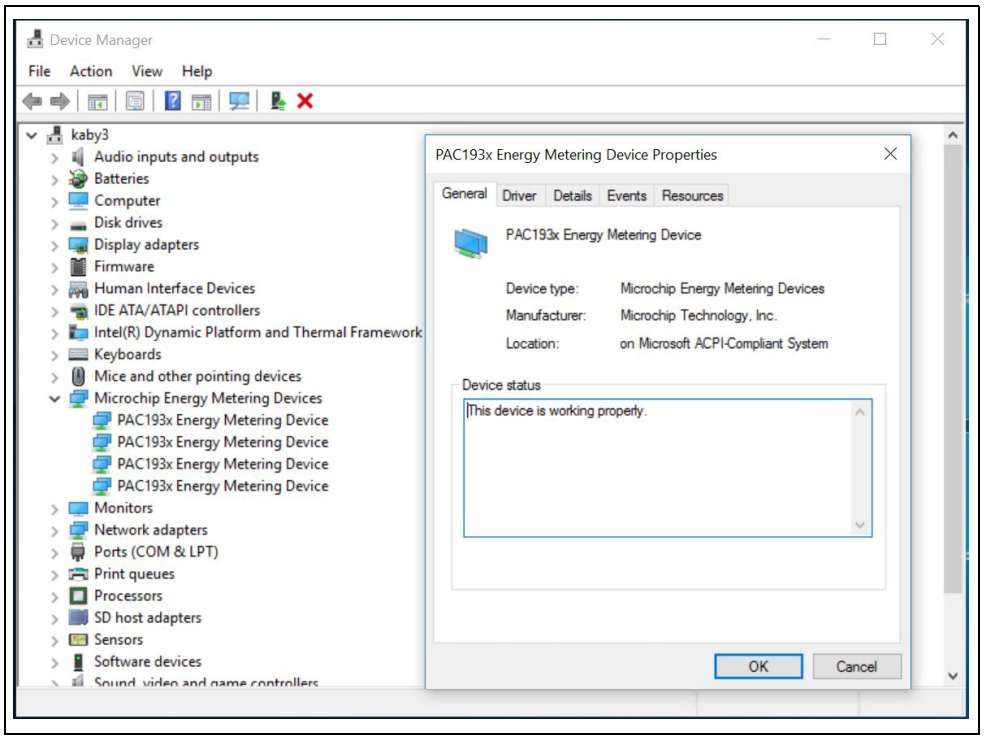

图2-1： *设备管理器显示PAC193X器件。*

要打开设备管理器：

1. 单击桌面左下方的**开始**按钮，在搜索框中输入设备管理器，然后单击菜单上的**设备管理器**。
2. 按<Windows>+<X>按键打开“快速访问菜单”，然后选择**设备管理器**。

注：

第3章 器件实例化和初始化

3.1 入门

连接到系统I²C总线的PAC193X器件由高级配置和电源接口（ACPI）按照其功能枚举为总线驱动程序，如ACPI\MCHP1930\<_UID>，其中_UID是ACPI ASL代码中为各器件定义的值。有关特定于PAC193X的ACPI ASL代码详细信息，请参见PAC193x应用笔记。

PAC193X Microsoft® Windows® 10驱动程序是一个标准内核模式驱动程序框架（Kernel-Mode Driver Framework，KMDF）功能驱动程序，根据其INF，每当总线驱动程序枚举硬件ID为*ACPI\MCHP1930*的器件时，PnP管理器便对此驱动程序进行实例化。

器件初始化期间，PAC193X Microsoft Windows 10 驱动程序评估设备的ACPI BIOS_DSM方法，其唯一标识ID（UUID）是*033771E0-1705-47B4-9535-D1BBE14D9A09*。_DSM方法返回的信息包括：

- 各通道的分流电阻（R_{SENSE}）值，用毫欧表示。
  - 电阻值用于电量计算，且必须为整数。当前驱动程序版本不支持分数形式的毫欧。
  - R_{SENSE} = 0表示未连接到电源轨的通道。驱动程序报告无此通道数据。
- Windows指定由通道监测的电源轨名称
  - 驱动程序创建用于所有指定通道的EMI器件接口。有关详细信息，请参见以下Microsoft文章：《电量计量接口》。如果Windows E3服务名称与Windows电源轨分类法兼容，则此服务可以使用由EMI接口输出的数据。
  - 如果通道名称为空字符串，则驱动程序不创建EMI。器件中为该通道累加的数据仍然通过PAC193X控制接口收集、处理和报告。各PAC器件仅有一个接口，因此用户可以看到哪个驱动程序创建了哪个器件接口。

3.2 器件初始配置

PAC193X Microsoft Windows 10 驱动程序使用与默认器件POR配置相同的配置：

- I²C/SMBus接口配置为I²C。Microsoft Windows目前不支持SMBus。驱动程序不允许更改任何通信设置。
- 采样率（SRN）：1024 Hz。只有所有器件通道为非EMI通道，驱动程序才会为用户提供更改选项。
- SLOW/ALERT引脚配置为SLOW。驱动程序不允许更改任何配置。

- SLOW信号转换触发受限REFRESH。更新所有结果寄存器且复位累加器与累加器计数后，定义由SLOW引脚触发的受限REFRESH，但不允许对器件配置进行任何更改。
- 使能所有器件通道。除非这些通道有开放式EMI接口，否则用户可以更改状态。
- 默认情况下，所有通道都是单向的。用户可以更改V_{BUS}、V_{SENSE}或两者的极性配置。相应调整驱动程序算法。更改有符号的测量值也意味着这些测量值失去1位分辨率。

此外，驱动程序还会执行必要的配置操作，以便为影响芯片初始版本的勘误表实现相应解决方法（版本ID = 0 x 01）。有关版本ID的详细信息，请参阅PAC193X数据手册。

第4章 PAC193X器件接口

4.1 简介

器件接口包含用户应用程序用于指引输入/输出（I/O）请求进入器件的符号链路名称。驱动程序创建接口时，还会关联接口类GUID（接口类型标识符）。

用户应用程序可以使用“*SetupDiEnumDeviceInterfaces*”函数调用发现属于某一GUID类的接口，然后使用“*SetupDiGetDeviceInterfaceDetail*”函数调用获取器件的符号链路名称。（本段所述的两个“SetupDi...”函数由`setupapi.lib`实现，且适用于Microsoft® Windows® 2000及以后版本）。

接下来，用户应用程序可以将符号链路名称传送到“*CreateFile*”内核函数，以打开I/O设备并接收分配的文件句柄。用户应用程序可以使用“*DeviceIoControl*”内核函数和文件句柄与器件进行通信，指定标识器件请求操作的IOCTL代码（输入/输出控制代码）。器件驱动程序将请求转换为器件特定的操作，并将结果返回给请求程序。

本章介绍由PAC193X Microsoft Windows 10驱动程序为其创建的各类型接口所实现的GUID类和IOCTL代码：

- 电量计量接口
- 器件控制接口

4.2 电量计量接口

驱动程序为各个器件通道创建一个EMI接口，并在ACPI ASL中分配一个非空Windows轨名称。有关EMI接口的详细信息，请参见以下Microsoft文章：《电量计量接口》。有关定义详细信息，请参见Microsoft Windows驱动程序工具包附带的文件`emi.h`。

EMI器件接口的GUID为：

45BD8344-7ED6-49cf-A440-C276C933B053

表4-1： EMI IOCTL

IOCTL名称	说明
`IOCTL_EMI_GET_MEASUREMENT`	检索当前电量测量和测量所用的时间。 • 输入：无 • 输出：`EMI_MEASUREMENT_DATA`
`IOCTL_EMI_GET_METADATA`	检索器件的EMI元数据。 • 输入：无 • 输出：`EMI_METADATA`
`IOCTL_EMI_GET_METADATA_SIZE`	通过发出`IOCTL_EMI_GET_METADATA`请求，检索可从器件获取的EMI元数据对象的大小。 • 输入：无 • 输出：`EMI_METADATA_SIZE`

表4-1： EMI IOCTL（续）

IOCTL名称	说明
IOCTL_EMI_GET_VERSION	检索器件支持的当前版本EMI接口。 • 输入：无 • 输出：EMI_VERSION

表4-2： EMI枚举和结构

主题	类型	说明
EMI_MEASUREMENT_DATA	输出结构	电量测量和测量所用时间，与IOCTL_EMI_GET_MEASUREMENT的报告相同。 • 电量单位：1 pWh • 电量值和时间戳值：64位无符号。 • 时间戳单位：100 ns。
EMI_MEASUREMENT_UNIT	枚举	显示可由 IOCTL_EMI_GET_MEASUREMENT报告的电量测量结果的可用单位。 • EMI V1仅实现1 pWh单位。
EMI_METADATA	输出结构	提供以下信息： • 计量器件：硬件型号和版本 • 计量电源轨：轨名称 与IOCTL_EMI_GET_METADATA的报告相同
EMI_METADATA_SIZE	输出结构	EMI_METADATA结构大小，与IOCTL_EMI_GET_METADATA_SIZE的报告相同
EMI_VERSION	输出结构	此器件支持的EMI版本，与IOCTL_EMI_GET_VERSION的报告相同 • 目前仅定义和支持EMI V1规范

附录B“PAC193X器件控制接口”包含EMI接口数据结构的C定义。

4.3 器件控制接口

驱动程序为各枚举的PAC193X器件创建一个控制接口。与EMI接口相比，该接口接受IOCTL_PAC193x_*请求，使用户可以访问更多数据和器件配置选项：

- 所有测量值（除累加电量外，还有如V_{BUS}和V_{SENSE}）
- 所有通道（包括非EMI的通道）的数据
- 各器件和通道的配置选项（采样率、通道极性和通道开/关）

注意

某些器件或通道配置选项被驱动程序暂时或永久阻止：

- **用户无法访问SMB/I^2C通信选项或SLOW/ALERT引脚配置选项**
- **只有为通道定义的任何EMI接口未受到配置更改的影响，才会允许访问器件采样率和通道开/关控制选项**

控制接口的GUID为：

4166FE9F-A865-4314-8942-7C12ABA290F6

表4-3： 器件控制IOCTL

IOCTL名称	说明
IOCTL_PAC193X_GET_DEVICE_INFO_SIZE	针对必须保留器件信息数据接口的缓冲区返回相关大小分配要求。 • 输入缓冲区：无 • 输出缓冲区：PAC193X_GET_DEVICE_INFO_SIZE
IOCTL_PAC193X_GET_DEVICE_INFO	返回有关器件版本和通道连接的信息（电源轨名称、检测电阻和连接状态）。 • 输入缓冲区：无 • 输出缓冲区：PAC193X_GET_DEVICE_INFO
IOCTL_PAC193X_GET_CTRL	返回器件寄存器CTRL、CTRL_ACT和CTRL_LAT的内容 • 输入缓冲区：无 • 输出缓冲区：PAC193X_GET_CTRL
IOCTL_PAC193X_SET_CTRL	通过CTRL寄存器：SRN、SLEEP和SING提供用户对某些器件配置位的有限控制：如果器件具有开放的EMI通道，则驱动程序拒绝配置请求。 • 输入缓冲区：PAC193X_SET_CTRL • 输出缓冲区：无（返回零字节）
IOCTL_PAC193X_GET_MEASUREMENTS	为所有通道返回采样时的数据测量结果、时间戳和器件配置。 • 输入缓冲区：无 • 输出缓冲区：PAC193X_MEASUREMENTS

表4-3： 器件控制IOCTL（续）

IOCTL名称	说明
IOCTL_PAC193X_GET_CHANNEL_CFG	返回I²C/SMB和通道开/关配置寄存器的内容：CHANNEL_DIS、CHANNEL_DIS_ACT和CHANNEL_DIS_LAT • 输入缓冲区：无 • 输出缓冲区：PAC193X_GET_CHANNEL_CFG
IOCTL_PAC193X_SET_CHANNEL_CFG	通过CHANNEL_DIS寄存器：CH1、CH2、CH3和CH4提供用户对某些配置位的有限控制。 驱动程序忽略更改I²C/SMB配置位的请求：TIMEOUT、BYTECOUNT和NOSKIP。驱动程序忽略更改指针跳过配置位的请求：NOSKIP。 驱动程序忽略具有开放EMI通道的那些通道的更改值。 • 输入缓冲区：PAC193X_SET_CHANNEL_CFG • 输出缓冲区：无（返回零字节）
IOCTL_PAC193X_GET_NEG_POWER	返回通道极性配置寄存器NEG_PWR、NEG_PWR_ACT和NEG_PWR_LAT的内容 • 输入缓冲区：无 • 输出缓冲区：PAC193X_GET_NEG_POWER
IOCTL_PAC193X_SET_NEG_POWER	提供用户对通道极性配置寄存器：NEG_PWR的控制。通道极性在驱动器初始化时设置为单极性，但无论EMI状态如何，用户可随时更改。 • 输入缓冲区：PAC193X_SET_NEG_POWER • 输出缓冲区：无（返回零字节）
IOCTL_PAC193X_GET_OVERFLOW	返回CTRL_ACT和CTRL_LAT中的OVF位的状态。 根据输入标志的值，该请求也可以清除CTRL寄存器中的OVF标志（复位为0）。 • 输入缓冲区：PAC193X_GET_OVERFLOW • 输出缓冲区：PAC193X_GET_OVERFLOW
IOCTL_PAC193X_DATA_REFRESH	允许器件报告测量值，无需复位累加器。使用REFRESH_V器件命令更新软件累加器 • 输入缓冲区：无 • 输出缓冲区：无（返回零字节）
IOCTL_PAC193X_DATA_REFRESH_RESET	使用REFRESH器件命令更新软件累加器。根据输入标志值，该请求也可以清除没有开放EMI接口的通道软件的累加器（复位为零）。 • 输入缓冲区：PAC193X_REFRESH_FLAGS • 输出缓冲区：无（返回零字节）

表4-3： 器件控制IOCTL（续）

IOCTL名称	说明
IOCTL_PAC193X_DATA_REFRESH_RESET_GLOBAL	对系统中所有PAC193X器件的软件累加器执行全局更新。根据输入标志的值，该请求也可以清除没有开放EMI接口的通道的软件累加器（复位为0）。 由于Microsoft Windows SpbCx框架不支持I^2C广播寻址（0 x 00），驱动程序无法使用REFRESH_G器件命令实现此请求。然而，驱动程序却可以实现此请求向系统中所有器件发送连续REFRESH命令。 • 输入缓冲区：PAC193X_REFRESH_FLAGS • 输出缓冲区：无（返回零字节）
IOCTL_PAC193X_GET_LAST_SLOW_DATA	在因系统状态从其他非供电状态转换到S0导致最后一个SLOW信号HL转换（SLOW触发的有限更新）的情况下，为所有通道返回由最后一个转换采样的所有数据测量。 • 输入缓冲区：无 • 输出缓冲区：PAC193X_LAST_SLOW_DATA
IOCTL_PAC193X_GET_SLOW	返回SLOW寄存器的值。 SLOW寄存器配置由驱动程序专门控制。 • 输入缓冲区：无 • 输出缓冲区：PAC193X_GET_SLOW

表4-4： 器件控制结构

结构名称	类型	说明
PAC193X_GET_DEVICE_INFO_SIZE	输出	PAC193X_GET_DEVICE_INFO结构的大小，如IOCTL_PAC193X_GET_DEVICE_INFO中所报告的
PAC193X_GET_DEVICE_INFO	输出	IOCTL_PAC193X_GET_DEVICE_INFO返回的器件和通道信息： • 制造商ID、产品ID和芯片版本 • 连接的通道数及其开/关状态 • 各通道轨名称、检测电阻值和EMI接口状态（是否创建） *注意：*该结构大小可变，具体取决于器件所拥有的通道数及通道名称大小。
PAC193X_GET_CTRL	输出	由下列函数返回的控制寄存器内容 IOCTL_PAC193X_GET_CTRL • CTRL • CTRL_ACT • CTRL_LAT
PAC193X_SET_CTRL	输入	用户必须向IOCTL_PAC193X_SET_CTRL提供的CTRL寄存器值

表4-4： 器件控制结构（续）

结构名称	类型	说明
PAC193X_MEASUREMENTS	输出	由IOCTL_PAC193X_GET_MEASUREMENTS返回的数据结构，包含器件寄存器值： • ACC_COUNT • VPOWERn_ACC • VBUSn和VBUSn_AVG • VSENSEn和VSENSEn_AVG • VPOWERn • NEG_PWR_LAT • CTRL_LAT • CHANNEL_DIS_LAT以及由器件驱动程序实现的软件寄存器： • 最新采样报告时间戳 • 软件累加器计数 • 无符号软件累加器 • 有符号软件累加器 *注意：* 如果为某些通道使能双向电压和/或电流，则报告值的二进制格式为2进制补码形式（MSB是符号位）。将报告值转换为后续处理所需的数据类型。 驱动程序已经实现影响版本ID = 0 x 01器件的VPOWER_ACC和VPOWERN勘误表解决方法
PAC193X_GET_CHANNEL_CFG	输出	由IOCTL_PAC193X_GET_CHANNEL_CFG返回的通道控制寄存器值： • CHANNEL_DIS， • CHANNEL_DIS_ACT， • CHANNEL_DIS_LAT
PAC193X_SET_CHANNEL_CFG	输入	用户必须向IOCTL_PAC193X_SET_CHANNEL_CFG提供的CHANNEL_DIS寄存器值
PAC193X_GET_NEG_POWER	输出	由IOCTL_PAC193X_GET_NEG_POWER返回的通道极性配置寄存器： • NEG_PWR • NEG_PWR_ACT • NEG_PWR_LAT
PAC193X_SET_NEG_POWER	输出	用户必须向IOCTL_PAC193X_SET_NEG_POWER提供的NEG_PWR寄存器值
PAC193X_GET_OVERFLOW	输入/输出	IOCTL_PAC193X_GET_OVERFLOW输入/输出结构，包含一个输入标志和两个输出标志： • 输入标志（若非零）请求复位CTRL中的OVF位。 • 输出标志返回CTRL_ACT和CTRL_LAT中的OVF位

表4-4： 器件控制结构（续）

结构名称	类型	说明
PAC193X_ REFRESH_FLAGS	输入	IOCTL_PAC193X_DATA_REFRESH_RESET和IOCTL_PAC193X_DATA_REFRESH_RESET_GLOBAL的输入结构要求包含一个输入标志。设置为PAC193X_REFRESH_FLAG_RESET_ACCUMULATORS（定义为1）的标志值也指示清除软件累加器的请求。
PAC193X_LAST_SLOW _DATA	输出	由IOCTL_PAC193X_GET_LAST_SLOW_DATA返回的数据结构，其包含在因系统状态从其他非供电状态转换到S0导致SLOW信号HL转换（SLOW触发的有限更新）的情况下，由SLOW信号HL转换而采样的器件寄存器值。 • ACC_COUNT • VPOWERn_ACC • VBUSn和VBUSn_AVG • VSENSEn和VSENSEn_AVG • VPOWERn • NEG_PWR_LAT • CTRL_LAT • CHANNEL_DIS_LAT
PAC193X_GET_SLOW	输出	由IOCTL_PAC193X_GET_SLOW返回的SLOW寄存器值

附录B“PAC193X器件控制接口”包含Control接口数据结构的C定义。

注：

第5章 PAC193X驱动程序特性

5.1 驱动程序特性

本节针对处理驱动程序报告数据的用户应用程序介绍可能影响其设计决策的驱动程序功能。

- 多个I/O请求
- 电量计量接口（EMI）与器件控制接口的比较
- 电量软件累加器
- 低功耗模式和电源状态

5.2 多个I/O请求

Windows® E3模块和多个其他用户应用程序（或同一应用程序的多个实例）可能会同时发出数据报告请求或器件配置命令。驱动程序将所有请求排成队列，以运行到完成的方式按顺序每次逐个执行请求，确保I^2C事务连续。

5.3 电量计量接口（EMI）与器件控制接口的比较

对于各PAC器件，PAC193X Microsoft® Windows 10驱动程序创建一个控制接口及零个或多个EMI接口，如果各连接通道（R_{SENSE} > 0）的非空EMI电源轨名称是用ACPI ASL代码分配指定的，则为每个连接通道创建一个EMI接口。

EMI接口仅报告上次测量的正向电量累加和时间戳。

控制接口报告由PAC193X器件执行的所有测量结果，包括由EMI接口报告的电量和时间戳数值，即使未定义任何EMI接口。在这种情况下，所有EMI轨名称为空字符串（“”）。

Windows电量估算引擎（E3）仅收集由EMI接口报告的电量数值（EMI电源轨名称必须符合Microsoft命名约定，以确保E3正确使用数据）。

用户应用程序可以从EMI接口、控制接口或两种类型接口收集数据：

- EMI接口具有Microsoft定义的控制代码和数据接口，使应用程序能够收集任何符合EMI标准的计量器件供应商的数据
- 控制接口使用Microchip定义的控制代码和数据结构提供更多信息

控制接口还提供对器件配置寄存器的访问，但有一些限制：

- 某些器件配置选项仅由驱动程序控制：I^2C接口参数、指针跳过、ALERT和SLOW引脚配置。

- 只有为通道开放的EMI接口未受到配置更改的影响，才会接受采样率和通道使能/禁止状态更改。

5.4 电量软件累加器

驱动程序为各PAC器件实现一组电量软件累加器，以便：

- 测量驱动器根据器件电源产品累加器值、R_{SENSE}值和采样率计算的实际电量值。这是通过EMI和Control接口报告的电量值，以皮瓦小时（pWh）为单位：
- 将电量累加时限延长到器件能力范围以外。软件累加器的64位分辨率可将到寄存器饱和时的17分钟最小累加时间延长至到软件累加器溢出时的大约2年。

器件采样计数器寄存器带有一个由驱动程序实现的软件采样计数器对。

每当驱动程序接收电量报告请求时，均会读取器件累加器寄存器、更新软件累计器并清除器件累加器。

例如，Windows E3模块通常每几分钟收集一次EMI电量。但即使没有EMI接口和用户应用程序请求电量报告，驱动程序也具有一个内部看门狗定时器，且仍可定期更新软件累加器并清除器件累加器。在当前实现中，看门狗时间设置为4分钟。

5.5 低功耗模式和电源状态

为PAC193X Microsoft Windows 10驱动程序描述的用例假定无论主机操作系统的电源状态如何，只要电源没有从系统中物理移除，PAC器件都接收电源。有关更多详细信息，请参见**第2章“驱动程序安装”**和PAC193X应用笔记。由于只要电源未移除，PAC器件就可以累积电量，所以有几个电源模式问题必须由驱动程序解决：

- 确定PAC器件是否已完成上电复位（POR），因此器件未初始化，在这种情况下，器件寄存器内容不可信且不要将数据添加到软件累加器；
- 收集由SLOW引脚从高到低转换保存的数据
  - 系统关闭（S5）、休眠（S4）或进入混合睡眠模式（无需移除电源）
- 当系统进入待机状态（S1至S3）或连接电源的待机状态（CS）时，将PAC器件配置为低功耗模式，以降低自身功耗。

5.5.1 低功耗模式

操作系统将PAC器件切换到非D0电源状态后，驱动程序将采样率重新配置为最低频率（8 Hz），以减少PAC器件自身的功耗。驱动程序不会将器件置于睡眠模式。因此，即使器件电源状态为D3（对于大多数外设器件而言，这表示关闭），PAC器件也会继续累加数据。

要确定在CS系统上何时进入低功耗模式，驱动程序还会注册以下电源设置回调：

- GUID_CONSOLE_DISPLAY_STATE
- GUID_GLOBAL_USER_PRESENCE

驱动程序退出低功耗模式后，如果自上次POR以来尚未设置采样率，则器件的采样频率从IOCTL_PAC193X_SET_CTRL返回到最后设置的采样率或最高采样率（1024个采样/秒）。

下列情况为真时退出低功耗模式：

- 器件处于D0状态
- 控制台显示状态为**PowerMonitorOn**或用户在线状态为**PowerUserActive**

通常，这意味着进入D0不会引起驱动程序立即从低功耗模式转出。

表5-1说明驱动程序应何时处于低功耗模式。

表5-1： 低功耗模式真值表

D0状态（工作中）	PowerMonitorOff	PowerUserInactive	低功耗模式
—	X	X	X
—	—	X	X
—	X	—	X
—	—	—	X
X	X	X	X
X	—	X	—
X	X	—	—
X	—	—	—

注：

附录A 驱动程序版本历史

A.1 故障修复和更改

表A-1： 驱动程序版本历史

版本	日期	说明
首次公开发布	2017年9月7日	• 为LAST_SLOW_DATA添加了三个锁存寄存器 • SET_NEG_POWER IOCTL允许设置寄存器，无论器件上是否存在公共通道 • SET_CHANNEL_CFG IOCTL允许使能或禁止个人专用通道，而器件上则无需公共通道 • 用户无法再通过SET_CHANNEL_CFG IOCTL设置ByteCount和Timeout字段； • TotalAccumulationCount从上一周期开始缓存，以正确支持REFRESH_V； • 将清除选项指定给REFRESH_RESET IOCTL时清除SoftwareAccumulatorCount； • 支持单次模式； • 轮询定时器重置不是在收到IOCTL时，而是仅限在清除硬件累加器时； • 通过资源重整修复了一个未解决的问题（HLK期间存在的问题）； • 添加了对第二个软件版本的支持。 • PACTest调整并显示IOCTL更改。

注：

附录B PAC193X器件控制接口

B.1 简介

PAC193X Microsoft® Windows® 10驱动程序创建的PAC193X器件控制接口由本附录中所述的C头文件示例定义：

- PAC193x_HW.h：包含PAC193X器件特定的定义
- PAC193x_INTF.h：包含接口和IOCTL定义

B.2 PAC193X_HW.H头文件

```
//////////////////////////////////////////////////////////////////////////////// //
//      (C) Copyright 2016-2017 OSR Open Systems Resources, Inc.
//      (C) 2017 Microchip Technology Inc. and its subsidiaries. You may use
//      this software and any derivatives exclusively with Microchip products.
//
//      THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
//      EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
//      IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR
//      A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
//      COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
//
//      IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL,
//      PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF
//      ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF
//      MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE
//      FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S
//      TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL
//      NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO
//      MICROCHIP FOR THIS SOFTWARE.
//
//      MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
//      THESE TERMS.
//
//    MODULE:
//       PAC193x_HW.h
//
//    ABSTRACT:
//       Device-Specific Hardware Interface for the Microchip PAC 193x Driver
//
//    AUTHOR(S):
//       OSR Open Systems Resources, Inc.
//       Microchip Technology, Inc.
////////////////////////////////////////////////////////////////////////////////
```

```
#pragma once
//////////////////////////
// DEVICE CONSTANTS   //
//////////////////////////
#define PAC_MAXIMUM_CHANNELS 4
////////////////////////////////////
// DEVICE REGISTER VALUE CONSTANTS //
////////////////////////////////////
#define PAC_PRODUCT_ID_1934         0x5B
#define PAC_PRODUCT_ID_1933         0x5A
#define PAC_PRODUCT_ID_1932         0x59

#define PAC_MANUFACTURER_ID         0x5D
#define PAC_REVISION                0x01
////////////////////////////////
// DEVICE REGISTER ADDRESSES //
////////////////////////////////
//
// Main control registers
//
#define PAC_REGISTER_REFRESH    0x00
#define PAC_REGISTER_CONTROL    0x01
//
// Measurement registers
//
#define PAC_REGISTER_ACC_COUNT          0x02
#define PAC_REGISTER_CH1_ACC_COUNT      0x03
#define PAC_REGISTER_CH2_ACC_COUNT      0x04
#define PAC_REGISTER_CH3_ACC_COUNT      0x05
#define PAC_REGISTER_CH4_ACC_COUNT      0x06

#define PAC_REGISTER_VBUS1              0x07
#define PAC_REGISTER_VBUS2              0x08
#define PAC_REGISTER_VBUS3              0x09
#define PAC_REGISTER_VBUS4              0x0A

#define PAC_REGISTER_VSENSE1            0x0B
#define PAC_REGISTER_VSENSE2            0x0C
#define PAC_REGISTER_VSENSE3            0x0D
#define PAC_REGISTER_VSENSE4            0x0E

#define PAC_REGISTER_VBUS1_AVG          0x0F
#define PAC_REGISTER_VBUS2_AVG          0x10
#define PAC_REGISTER_VBUS3_AVG          0x11
#define PAC_REGISTER_VBUS4_AVG          0x12

#define PAC_REGISTER_VSENSE1_AVG        0x13
#define PAC_REGISTER_VSENSE2_AVG        0x14
```

```
#define PAC_REGISTER_VSENSE3_AVG        0x15
#define PAC_REGISTER_VSENSE4_AVG        0x16

#define PAC_REGISTER_VPOWER1            0x17
#define PAC_REGISTER_VPOWER2            0x18
#define PAC_REGISTER_VPOWER3            0x19
#define PAC_REGISTER_VPOWER4            0x1A
//
// Various other control registers
//
#define PAC_REGISTER_CHANNEL_DIS        0x1C
#define PAC_REGISTER_NEG_PWR            0x1D
#define PAC_REGISTER_REFRESH_G          0x1E
#define PAC_REGISTER_REFRESH_V          0x1F
#define PAC_REGISTER_SLOW               0x20
#define PAC_REGISTER_CTRL_ACT           0x21
#define PAC_REGISTER_CHANNEL_DIS_ACT    0x22
#define PAC_REGISTER_NEG_PWR_ACT        0x23
#define PAC_REGISTER_CTRL_LAT           0x24
#define PAC_REGISTER_CHANNEL_DIS_LAT    0x25
#define PAC_REGISTER_NEG_PWR_LAT        0x26
//
// Chip IDs
//
#define PAC_REGISTER_PRODUCT_ID         0xFD
#define PAC_REGISTER_MANUFACTURER_ID    0xFE
#define PAC_REGISTER_CHIP_REVISION      0xFF
////////////////////////////////
// DEVICE REGISTER DEFINITIONS //
////////////////////////////////
#pragma push(pack:1)
typedef struct _PAC193X_CTRL_REGISTER {
    UCHAR Overflow             : 1;
    UCHAR OverflowAlert        : 1;
    UCHAR AlertConversion      : 1;
    UCHAR AlertPin             : 1;
    UCHAR SingleShotMode       : 1;
    UCHAR Sleep                : 1;
    UCHAR SampleRateNormalMode : 2;
} PAC193X_CTRL_REGISTER, *PPAC193X_CTRL_REGISTER;
typedef struct _PAC193X_CHANNEL_DIS_ACT_REGISTER {
    UCHAR             : 1;
    UCHAR NoSkip      : 1;
    UCHAR ByteCount   : 1;
    UCHAR Timeout     : 1;
    UCHAR Channel4Off : 1;
    UCHAR Channel3Off : 1;
```

```
    UCHAR Channel2Off : 1;
    UCHAR Channel1Off : 1;
} PAC193X_CHANNEL_DIS_REGISTER, *PPAC193X_CHANNEL_DIS_REGISTER;
typedef struct _PAC193X_NEG_PWR_REGISTER {
    UCHAR Channel4BIDV : 1;
    UCHAR Channel3BIDV : 1;
    UCHAR Channel2BIDV : 1;
    UCHAR Channel1BIDV : 1;
    UCHAR Channel4BIDI : 1;
    UCHAR Channel3BIDI : 1;
    UCHAR Channel2BIDI : 1;
    UCHAR Channel1BIDI : 1;
} PAC193X_NEG_PWR_REGISTER, *PPAC193X_NEG_PWR_REGISTER;
typedef struct _PAC193X_SLOW_REGISTER {
    UCHAR PowerOnReset : 1;
    UCHAR RefreshVFall : 1;
    UCHAR RefreshFall  : 1;
    UCHAR RefreshVRise : 1;
    UCHAR RefreshRise  : 1;
    UCHAR SlowHighLow  : 1;
    UCHAR SlowLowHigh  : 1;
    UCHAR Slow         : 1;
} PAC193X_SLOW_REGISTER, *PPAC193X_SLOW_REGISTER;
#pragma pop(pack)
```

B.3 PAC193X_INTF.H头文件

```
//////////////////////////////////////////////////////////////////////////////
//      (C) Copyright 2016-2017 OSR Open Systems Resources, Inc.
//      (C) 2017 Microchip Technology Inc. and its subsidiaries. You may use
//      this software and any derivatives exclusively with Microchip products.
//
//      THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
//      EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
//      IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR
//      A PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS,
//      COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
//
//      IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL,
//      PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF
//      ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF
//      MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE
//      FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S
//      TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL
//      NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO
//      MICROCHIP FOR THIS SOFTWARE.
//
//      MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF
//      THESE TERMS.
//
//    MODULE:
//       PAC193x_INTF.h
//
//    ABSTRACT:
//       Device-Specific Interface and IOCTL definitions for
//       the Microchip PAC 193x EMI Driuver
//
//    AUTHOR(S):
//        OSR Open Systems Resources, Inc.
//        Microchip Technology, Inc.
//////////////////////////////////////////////////////////////////////////////
#pragma once
#include <initguid.h>
#include <emi.h>
// {4166FE9F-A865-4314-8942-7C12ABA290F6}
// Identifies the control (non-EMI) portion of the interface
//
#define GUID_PAC193X_CONTROL_INTERFACE_STR L"{4166FE9F-A865-4314-8942-7C12ABA290F6}"
DEFINE_GUID(GUID_PAC193X_CONTROL_INTERFACE,
    0x4166fe9f, 0xa865, 0x4314, 0x89, 0x42, 0x7c, 0x12, 0xab, 0xa2, 0x90, 0xf6);
//
// Register definitions
//
```

```
#include "PAC193x_HW.h"
#define FILE_DEVICE_PAC193X   0x913E

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_DEVICE_INFO_SIZE
//
// Returns the size allocation requirements for the buffer that must hold
// the device info data structure.
// - input buffer: none
// - output buffer: PAC193X_GET_DEVICE_INFO_SIZE
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_GET_DEVICE_INFO_SIZE {
    ULONG BufferSize;
}PAC193X_GET_DEVICE_INFO_SIZE, *PPAC193X_GET_DEVICE_INFO_SIZE;
#define IOCTL_PAC193X_GET_DEVICE_INFO_SIZE CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1929,\
                                                    METHOD_BUFFERED,\
                                                    FILE_READ_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_DEVICE_INFO
//
// Returns information about the device version and about the channel
// connections (rail names, sense resistors, connection status).
// - input buffer: none
// - output buffer: PAC193X_GET_DEVICE_INFO
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_DEVICE_CHANNEL_INFO {
    ULONG NextChannelInfoOffset;

    BOOLEAN ChannelInUse;               // TRUE if EMI is created
                                        // (the channel is named)

    ULONG RsenseMOhm;                   // Rsense in milliohms
                                        // 0 indicates non-connected channel

    WCHAR ChannelName[ANYSIZE_ARRAY];   // EMI channel name.
                                        // If the string is empty (""),
                                        // EMI is not created for this channel
                                        // reported also as
                                        // EMI_METADATA.MeteredHardwareName
} PAC193X_DEVICE_CHANNEL_INFO, *PPAC193X_DEVICE_CHANNEL_INFO;

typedef struct _PAC193X_GET_DEVICE_INFO {
    UCHAR ProductId;           // PRODUCT ID register value
    UCHAR ManufacturerId;      // MANUFACTURER ID register value
    UCHAR ProductRevision;     // REVISION ID register value
                               // reported also as EMI_METADATA.HardwareRevision
```

```
    UCHAR ChannelDescRegister; // CHANNEL_DIS register value
    ULONG ChannelCount;        // the number of "connected"
                               // channels (Rsense > 0)

    PAC193X_DEVICE_CHANNEL_INFO ChannelInfo;  // there are 2,3 or 4
                                              // "CHANNEL_INFO" structures
                                              // depending on device type:
                                              // PAC1932, PAC1933 or PAC1934
} PAC193X_GET_DEVICE_INFO, *PPAC193X_GET_DEVICE_INFO;

#define IOCTL_PAC193X_GET_DEVICE_INFO      CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1930,\
                                                    METHOD_BUFFERED,\
                                                    FILE_READ_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_CTRL
//
// Returns the content of the device registers CTRL, CTRL_ACT, CTRL_LAT
// - input buffer: none
// - output buffer: PAC193X_GET_CTRL
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_GET_CTRL {
    PAC193X_CTRL_REGISTER ControlRegister;      // CTRL register value
    PAC193X_CTRL_REGISTER ControlAct;           // CTRL_ACT register value
    PAC193X_CTRL_REGISTER ControlLat;           // CTRL_ACT register value
} PAC193X_GET_CTRL, *PPAC193X_GET_CTRL;

#define IOCTL_PAC193X_GET_CTRL             CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1931,\
                                                    METHOD_BUFFERED,\
                                                    FILE_READ_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_SET_CTRL
//
// Provides user limited control over some device configuration bits from
// CTRL register: SRN, SLEEP and SING. The driver rejects the configuration
// requests if the device has open EMI channels.
// - input buffer: PAC193X_SET_CTRL
// - output buffer: none (zero bytes returned)
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_SET_CTRL {
    PAC193X_CTRL_REGISTER ControlRegister;  // value to be written in
                                            // CTRL register
} PAC193X_SET_CTRL, *PPAC193X_SET_CTRL;
```

```
#define IOCTL_PAC193X_SET_CTRL             CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1932,\
                                                    METHOD_BUFFERED,\
                                                    FILE_WRITE_DATA)

//////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_MEASUREMENTS
//
// Provides user limited control over some device configuration bits from CTRL
// register: SRN, SLEEP and SING. The driver rejects the configuration requests
// if the device has open EMI channels.
// - input buffer: PAC193X_SET_CTRL
// - output buffer: none (zero bytes returned)
//////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_MEASUREMENTS {
    ULONG     AccumulatorCount;                 // ACC_COUNT register value

    // unipolar or bipolar registers - section begin
    ULONGLONG AccumulatorOutput[PAC_MAXIMUM_CHANNELS];
                                                // VPOWERn_ACC registers values
    USHORT    VBus[PAC_MAXIMUM_CHANNELS];       // VBUSn register values
    USHORT    VSense[PAC_MAXIMUM_CHANNELS];     // VSENSEn registers values
    USHORT    VBusAvg[PAC_MAXIMUM_CHANNELS];    // VBUSn_AVG registers values
    USHORT    VSenseAvg[PAC_MAXIMUM_CHANNELS];  // VSENSEn_AVG registers values
    ULONG     VPower[PAC_MAXIMUM_CHANNELS];     // VPOWERn registers values

    // CAUTION: If the bidirectional voltage and/or current is enabled for
    //          certain channels the binary format of the reported values is in
    //          2's complement (the MSB is a sign bit).
    //          So, please make sure to cast the reported value to the necessary
    //          data type for the subsequent processing.

    // REMARK: the driver already implements the workaround for the VPOWER_ACC
    //         and VPOWERN errata which affects the devices with Revision ID = 0x01
    // unipolar or bipolar registers - section end

    // software accumulators - section begin
    // The driver accumulates in software accumulators the values reported by
    // devices and clears the device accumulators in order to avoid the
    // devices accumulators saturation
    ULONGLONG SofwareAccumulatorCount;      // software sample counter value

    ULONGLONG SofwareAccumulator[PAC_MAXIMUM_CHANNELS];
                      // Energy accumulator, as defined by EMI.
                      // If the channel is configured as bi-polar, only the
                      // positive energy increments are accumulated.
                      // The energy energy is reported in pico-watt-hours (pWh).
                      // Same values are also reported by
```

```
                    // EMI_MEASUREMENT_DATA.AbsoluteEnergy

    LONGLONG  SignedSoftwareAccumulator[PAC_MAXIMUM_CHANNELS];
                    // Signed energy accumulator (MSB is sign bit).
                    // Both positive and negative energy values are accumulated.
                    // There is one bit resolution loss due to sign bit but
                    // this allows bipolar energy measurements.
                    // The energy energy is reported in pico-watt-hours (pWh).
    // software accumulators - section end

    PAC193X_NEG_PWR_REGISTER     NegPowerLat;   // NEG_PWR_LAT register value
    PAC193X_CTRL_REGISTER        CtrlLat;       // CTRL_LAT register value
    PAC193X_CHANNEL_DIS_REGISTER ChannelDisLat; // CHANNEL_DIS_LAT register value

    ULONGLONG       Timestamp;      // timestamp of the last measurements
                                    // reported in multiples of 100ns.
                                    // The timestamp is created by the driver
                                    // by reading the system performance counter
                                    // (high resolution timer, <1us)
                                    // Same values are also reported by
                                    // EMI_MEASUREMENT_DATA.AbsoluteTime
}PAC193X_MEASUREMENTS, *PPAC193X_MEASUREMENTS;

#define IOCTL_PAC193X_GET_MEASUREMENTS     CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1933,\
                                                    METHOD_BUFFERED,\
                                                    FILE_READ_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_CHANNEL_CFG
//
// Returns the content of the I2C/SMB and channel ON/OFF configuration registers:
// CHANNEL_DIS, CHANNEL_DIS_ACT, CHANNEL_DIS_LAT
// - input buffer: none
// - output buffer: PAC193X_GET_CHANNEL_CFG
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_GET_CHANNEL_CFG {
    PAC193X_CHANNEL_DIS_REGISTER ConfigRegister;
                                                // CHANNEL_DIS register value
    PAC193X_CHANNEL_DIS_REGISTER ConfigAct;     // CHANNEL_DIS_ACT register value
    PAC193X_CHANNEL_DIS_REGISTER ConfigLat;     // CHANNEL_DIS_LAT register value
} PAC193X_GET_CHANNEL_CFG, *PPAC193X_GET_CHANNEL_CFG;

#define IOCTL_PAC193X_GET_CHANNEL_CFG      CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1934,\
                                                    METHOD_BUFFERED,\
                                                    FILE_READ_DATA)
```

```
//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_SET_CHANNEL_CFG
//
// Provides user limited control over some configuration bits from CHANNEL_DIS
// register: CH1, CH2, CH3 and CH4.
// Notes:
//  - The driver ignores the requests to change the I2C/SMB configuration bits:
//    TIMEOUT and BYTECOUNT, NOSKIP
//  - The driver ignores the requests to change the pointer skip configuration
//    bit: NOSKIP
//  - The driver ignores the change values for those channels which has
//    open EMI channels.
//
// - input buffer: PAC193X_SET_CHANNEL_CFG
// - output buffer: none (zero bytes returned)
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_GET_SET_CHANNEL_CFG {
    PAC193X_CHANNEL_DIS_REGISTER ConfigRegister;
} PAC193X_SET_CHANNEL_CFG, *PPAC193X_SET_CHANNEL_CFG;

#define IOCTL_PAC193X_SET_CHANNEL_CFG      CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1935,\
                                                    METHOD_BUFFERED,\
                                                    FILE_WRITE_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_NEG_POWER
//
// Returns the content of the channel polarity configuration registers NEG_PWR,
// NEG_PWR_ACT, NEG_PWR_LAT
// - input buffer: none
// - output buffer: PAC193X_GET_NEG_POWER
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_GET_NEG_POWER {
    PAC193X_NEG_PWR_REGISTER NegPowerRegister;      // NEG_PWR register value
    PAC193X_NEG_PWR_REGISTER NegPowerAct;           // NEG_PWR_ACT register value
    PAC193X_NEG_PWR_REGISTER NegPowerLat;           // NEG_PWR_LAT register value
} PAC193X_GET_NEG_POWER, *PPAC193X_GET_NEG_POWER;

#define IOCTL_PAC193X_GET_NEG_POWER        CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1936,\
                                                    METHOD_BUFFERED,\
                                                    FILE_READ_DATA)
```

```
////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_SET_NEG_POWER
//
// Provides user control over channel polarity configuration register: NEG_PWR.
// Channel polarity is set to unipolar at the driver initialization but can be
// changed by the user at any moment, regardless the EMI status.
// - input buffer: PAC193X_SET_NEG_POWER
// - output buffer: none (zero bytes returned)
////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_SET_NEG_POWER {
    PAC193X_NEG_PWR_REGISTER NegPowerRegister;      // value to be written in
                                                    // NEG_PWR register
} PAC193X_SET_NEG_POWER, *PPAC193X_SET_NEG_POWER;

#define IOCTL_PAC193X_SET_NEG_POWER        CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1937,\
                                                    METHOD_BUFFERED,\
                                                    FILE_WRITE_DATA)

////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_OVERFLOW
//
// Returns the status of the OVF bit from CTRL_ACT and CTRL_LAT.
// Depending on the value of the input flag, this request can also clear
// (reset to 0) the OVF flag from CTRL register.
// - input buffer: PAC193X_GET_OVERFLOW
// - output buffer: PAC193X_GET_OVERFLOW
////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_GET_OVERFLOW {
    UCHAR OvfClear;     // In: the user set nonzero value to request the driver
                        // to clear the OVF flag (in device CTRL register)

    UCHAR OvfActual;    // Out: the IOCTL_PAC193X_GET_OVERFLOW driver returns in
                        // the LSB the value of the OVF bit from CTRL_ACT

    UCHAR OvfLatch;     // Out: the IOCTL_PAC193X_GET_OVERFLOW driver returns in
                        // the LSB the value of the OVF bit from CTRL_ACT
} PAC193X_GET_OVERFLOW, *PPAC193X_GET_OVERFLOW;

#define IOCTL_PAC193X_GET_OVERFLOW      CTL_CODE(FILE_DEVICE_PAC193X,\
                                                 1938,\
                                                 METHOD_BUFFERED,\
                                                 FILE_READ_DATA | FILE_WRITE_DATA)
```

```
//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_DATA_REFRESH
//
// Performs the update of the software accumulators using the REFRESH_V
// device command
// - input buffer: none
// - output buffer: none (zero bytes returned)
//////////////////////////////////////////////////////////////////////////////
#define IOCTL_PAC193X_DATA_REFRESH         CTL_CODE(FILE_DEVICE_PAC193X,\
                                                    1939,\
                                                    METHOD_BUFFERED,\
                                                    FILE_READ_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_DATA_REFRESH_RESET
//
// Performs the update of the software accumulators using the REFRESH device
// command. Depending on the value of the input flag, this request can also clear
// (reset to 0) the software accumulators for the channels without open EMI
// interfaces.
// - input buffer: PAC193X_REFRESH_FLAGS
// - output buffer: none (zero bytes returned)
//////////////////////////////////////////////////////////////////////////////
#define PAC193X_REFRESH_FLAG_RESET_ACCUMULATORS (1 << 0)
typedef struct _PAC193X_REFRESH_FLAGS {
    ULONG Flags;    // In: the user set the value of this field to
                    // PAC193X_REFRESH_FLAG_RESET_ACCUMULATORS (1) in order to
                    // request the driver to clear the software accumulators
                    // - SofwareAccumulatorCount
                    // - SofwareAccumulator[PAC_MAXIMUM_CHANNELS]
                    // - SignedSoftwareAccumulator[PAC_MAXIMUM_CHANNELS]
                    //
                    // CAUTION: SofwareAccumulator and SignedSoftwareAccumulator
                    //          for the EMI channels are not reset
} PAC193X_REFRESH_FLAGS, *PPAC193X_REFRESH_FLAGS;

#define IOCTL_PAC193X_DATA_REFRESH_RESET  CTL_CODE(FILE_DEVICE_PAC193X,\
                                                   1940,\
                                                   METHOD_BUFFERED,\
                                                   FILE_READ_DATA | FILE_WRITE_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_DATA_REFRESH_RESET_GLOBAL
//
// Performs a global update of the software accumulators for all the PAC193x
// devices in the system. Depending on the value of the input flag, this request
// can also clear (reset to 0) the software accumulators for the channels without
// open EMI interfaces.
```

```
// - input buffer: PAC193X_REFRESH_FLAGS
// - output buffer: none (zero bytes returned)
//
// NOTE: because Windows SpbCx framework does not support the I2C general call
// address (0x00), the driver cannot implement this request using REFRESH_G
// device commands. Instead, the driver implements this request sending
// sequential REFRESH commands to all the devices in the system.
////////////////////////////////////////////////////////////////////////////
#define IOCTL_PAC193X_DATA_REFRESH_RESET_GLOBAL  CTL_CODE(FILE_DEVICE_PAC193X,\
                                                          1941,\
                                                          METHOD_BUFFERED,\
                                                          FILE_READ_DATA)

////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_LAST_SLOW_DATA
//
// Returns for all the channels all the data measurements sampled by the last
// SLOW signal HL transition (SLOW triggered "limited" refresh) caused by the
// system state transition to S0 from other non-powered state.
// - input buffer: none
// - output buffer: PAC193X_LAST_SLOW_DATA
////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_LAST_SLOW_DATA {
    ULONG       AccumulatorCount;               // ACC_COUNT register value

    // unipolar or bipolar registers - section begin
    ULONGLONG   AccumulatorOutput[PAC_MAXIMUM_CHANNELS];
                                                // VPOWERn_ACC registers values
    USHORT      VBus[PAC_MAXIMUM_CHANNELS];         // VBUSn register values
    USHORT      VSense[PAC_MAXIMUM_CHANNELS];       // VSENSEn registers values
    USHORT      VBusAvg[PAC_MAXIMUM_CHANNELS];      // VBUSn_AVG registers values
    USHORT      VSenseAvg[PAC_MAXIMUM_CHANNELS];    // VBUSn_AVG registers values
    ULONG       VPower[PAC_MAXIMUM_CHANNELS];       // VPOWERn registers values

    // CAUTION: If the bidirectional voltage and/or current is enabled for
    //          certain channels the binary format of the reported values is in
    //          2's complement (the MSB is a sign bit). So, please make sure
    //          to cast the reported value to the necessary data type for the
    //          subsequent processing.
    //
    // REMARK: the driver already implements the workaround for the VPOWER_ACC
    //         and VPOWERN errata which affects the devices with
    //         Revision ID = 0x01
    // unipolar or bipolar registers - section end

    PAC193X_NEG_PWR_REGISTER     NegPowerLat;   // NEG_PWR_LAT register value
    PAC193X_CTRL_REGISTER        CtrlLat;       // CTRL_LAT register value
```

```
    PAC193X_CHANNEL_DIS_REGISTER ChannelDisLat; // CHANNEL_DIS_LAT register value
}PAC193X_LAST_SLOW_DATA, *PPAC193X_LAST_SLOW_DATA;

#define IOCTL_PAC193X_GET_LAST_SLOW_DATA  CTL_CODE(FILE_DEVICE_PAC193X,\
                                                   1942,\
                                                   METHOD_BUFFERED,\
                                                   FILE_READ_DATA)

//////////////////////////////////////////////////////////////////////////////
// IOCTL_PAC193X_GET_SLOW
//
// Returns the value of the SLOW register.
// Note: the configuration of the SLOW register is controlled exclusively
//       by the driver.
// - input buffer: none
// - output buffer: PAC193X_GET_SLOW
//////////////////////////////////////////////////////////////////////////////
typedef struct _PAC193X_GET_SLOW {
    PAC193X_SLOW_REGISTER SlowRegister;     // SLOW register value
} PAC193X_GET_SLOW, *PPAC193X_GET_SLOW;

#define IOCTL_PAC193X_GET_SLOW            CTL_CODE(FILE_DEVICE_PAC193X,\
                                                   1943,\
                                                   METHOD_BUFFERED,\
                                                       FILE_READ_DATA)
```

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

奥斯汀Austin, TX
Tel: 1-512-257-3370

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

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

底特律Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453
Tel: 1-317-536-2380

洛杉矶Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608
Tel: 1-951-273-7800

罗利Raleigh, NC
Tel: 1-919-844-7510

纽约New York, NY
Tel: 1-631-435-6000

圣何塞San Jose, CA
Tel: 1-408-735-9110
Tel: 1-408-436-4270

加拿大多伦多Toronto
Tel: 1-905-695-1980
Fax: 1-905-695-2078

亚太地区

中国-北京
Tel: 86-10-8569-7000

中国-成都
Tel: 86-28-8665-5511

中国-重庆
Tel: 86-23-8980-9588

中国-东莞
Tel: 86-769-8702-9880

中国-广州
Tel: 86-20-8755-8029

中国-杭州
Tel: 86-571-8792-8115

中国-南京
Tel: 86-25-8473-2460

中国-青岛
Tel: 86-532-8502-7355

中国-上海
Tel: 86-21-3326-8000

中国-沈阳
Tel: 86-24-2334-2829

中国-深圳
Tel: 86-755-8864-2200

中国-苏州
Tel: 86-186-6233-1526

中国-武汉
Tel: 86-27-5980-5300

中国-西安
Tel: 86-29-8833-7252

中国-厦门
Tel: 86-592-238-8138

中国-香港特别行政区
Tel: 852-2943-5100

中国-珠海
Tel: 86-756-321-0040

台湾地区-高雄
Tel: 886-7-213-7830

台湾地区-台北
Tel: 886-2-2508-8600

台湾地区-新竹
Tel: 886-3-577-8366

亚太地区

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

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

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

印度India - Pune
Tel: 91-20-4121-0141

日本Japan - Osaka
Tel: 81-6-6152-7160

日本Japan - Tokyo
Tel: 81-3-6880-3770

韩国Korea - Daegu
Tel: 82-53-744-4301

韩国Korea - Seoul
Tel: 82-2-554-7200

马来西亚
Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

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

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

新加坡Singapore
Tel: 65-6334-8870

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

越南Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

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

芬兰Finland - Espoo
Tel: 358-9-4520-820

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

德国Germany - Garching
Tel: 49-8931-9700

德国Germany - Haan
Tel: 49-2129-3766400

德国Germany - Heilbronn
Tel: 49-7131-67-3636

德国Germany - Karlsruhe
Tel: 49-721-625370

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

德国Germany - Rosenheim
Tel: 49-8031-354-560

以色列Israel - Ra'anana
Tel: 972-9-744-7705

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

意大利Italy - Padova
Tel: 39-049-7625286

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

挪威Norway - Trondheim
Tel: 47-7289-7561

波兰Poland - Warsaw
Tel: 48-22-3325737

罗马尼亚
Romania - Bucharest
Tel: 40-21-407-87-50

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

瑞典Sweden - Gothenberg
Tel: 46-31-704-60-40

瑞典Sweden - Stockholm
Tel: 46-8-5090-4654

英国UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820