PAC193X

Microsoft® Windows® 10

드라이버 사용자 가이드

© 2018 Microchip Technology Inc.

Microchip 디바이스의 코드 보호 기능에 대한 자세한 내용은 다음과 같습니다.

- Microchip 제품은 특정 Microchip 데이터 시트에 포함된 사양을 충족합니다.
- Microchip은 자사 제품군을 의도한 방식에 따라 정상 조건에서 사용할 경우 현재 시장에 출시된 동종 제품 중 가장 안전한 제품군으로 봅니다.
- 코드 보호 기능을 훼손하는 데 부정 및 심지어 불법적인 수단까지 이용되고 있습니다. Microchip이 알고 있는 바에 따르면, 이러한 모든 수단을 이용하기 위해서는 Microchip 데이터 시트에 포함된 동작 사양 외의 방법으로 Microchip 제품을 사용해야 합니다. 대부분의 경우 이러한 행위를 하는 자는 지적 자산 도용과 연관되어 있습니다.
- Microchip은 코드 안전성에 대해 우려하는 고객과 협력하고자 합니다.
- Microchip이나 다른 반도체 제조업체도 코드 안전성을 보장할 수 없습니다. 코드 보호는 Microchip이 제품의 "침해 불가능성"을 보증한다는 의미가 아닙니다.

코드 보호는 지속적으로 발전하고 있습니다. Microchip은 당사 제품의 코드 보호 기능을 지속적으로 개선하기 위해 노력하고 있습니다. Microchip의 코드 보호 기능을 훼손하려는 시도는 디지털 밀레니엄 저작권법(디지털 Millennium Copyright Act) 위반에 해당할 수 있습니다. 이러한 행위에 의해 귀하의 소프트웨어 또는 기타 저작권을 소유한 작업물에 무단 액세스가 이루어졌다면 해당 법률을 근거로 소송을 제기할 수 있습니다.

디바이스 장치 및 유사 내용과 관련하여 본 출판물에 포함된 정보는 귀하의 편의를 위해 제공된 것이며 업데이트로 대체될 수 있습니다. 귀하의 장치가 해당 사양을 충족하는지 확인하는 것은 귀하의 책임입니다. Microchip은 정보와 관련하여 명시적, 묵시적, 서면, 구두, 법적 또는 다른 보증이나 진술을 하지 않으며, 여기에는 조건, 품질, 성능, 상업성, 특정 목적에의 적합성을 포함하되 이에 제한되지 않습니다. Microchip은 이 정보 및 이 정보의 사용으로 발생한 모든 책임을 부인합니다. Microchip 장치를 생명 유지 및/또는 안전 장치 용도로 사용하는 위험은 전적으로 구매자에게 있으며, 구매자는 그러한 사용으로 인해 발생한 모든 손해, 청구, 소송 또는 비용으로부터 Microchip을 옹호, 보호하며 배상을 약속하는 데 동의합니다. 다르게 명시되지 않는 한, Microchip 지적 재산권하에 어떠한 라이센스도 묵시적 또는 다른 방식으로 제공되지 않습니다.

Microchip은 세계 본사, 애리조나 주 챈들러 및 템피, 오리건 주 그레셤의 설계 및 웨이퍼 가공 설비와 캘리포니아 주 및 인도의 설계 센터에 대해 ISO/TS-16949:2009 인증을 획득했습니다. Microchip의 고품질 시스템 프로세스 및 절차는 PIC® MCU 및 dsPIC® DSC, KEELOQ® 코드 호핑 디바이스, 직렬 EEPROM, 마이크로 주변장치, 비휘발성 메모리 및 아날로그 제품에 적용되고 있습니다. Microchip의 개발 시스템의 설계 및 제조를 위한 품질 시스템도 ISO 9001:2000 인증을 획득했습니다.

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

상표

Microchip 이름과 로고, Microchip 로고, AnyRate, AVR, AVR 로고, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ 로고, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST 로고, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 로고, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST 로고, SuperFlash, tinyAVR, UNI/O 및 XMEGA는 미국과 기타 국가에서 Microchip Technology Incorporated의 등록 상표입니다.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge 및 Quiet-Wire는 미국에서 Microchip Technology Incorporated의 등록 상표입니다.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT 로고, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet 로고, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified 로고, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch 로고, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA 및 ZENA는 미국과 기타 국가에서 Microchip Technology Incorporated의 상표입니다.

SQTP는 미국에서 Microchip Technology Incorporated의 서비스 마크입니다.

Silicon Storage Technology는 기타 국가에서 Microchip Technology Incorporated의 등록 상표입니다.

GestIC는 기타 국가에서 Microchip Technology Inc.의 자회사인 Microchip Technology Germany II GmbH & Co. KG의 등록 상표입니다.

본 자료에 언급된 그 외의 상표는 해당 기업의 재산입니다.

© 2018, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-2700-1

목차

서문 **5**
소개 5
문서 레이아웃 5
이 가이드의 표기법 6
권장 참조 자료 7
Microchip 웹 사이트 7
고객 지원 7
문서 개정 내역 7

1장 제품 개요
1.1 소개 9

2장 드라이버 설치
2.1 사전 요구 사항 11
2.1.1 호스트 시스템에 PAC193X 통합 11
2.1.2 SpbCx 규격의 I2C 버스 컨트롤러 디바이스 드라이버 11
2.2 드라이버 설치 12
2.2.1 Microsoft® Windows® 업데이트 또는 장치 관리자를 사용하여 설치 12
2.2.2 드라이버 설치 관리자 애플리케이션 12
2.2.3 DevCon 유틸리티를 사용하여 설치 12

3장 디바이스 인스턴스화 및 초기화
3.1 시작하기 15
3.2 디바이스 초기 구성 15

4장 PAC193X 디바이스 인터페이스
4.1 소개 17
4.2 에너지 측정 인터페이스 17
4.3 디바이스 컨트롤 인터페이스 19

5장 PAC193X 드라이버 특징
5.1 드라이버 특징 25
5.2 복수 I/O 요청 25
5.3 EMI(에너지 측정 인터페이스) 및 디바이스 컨트롤 인터페이스 25
5.4 에너지 소프트웨어 누산기 26
5.5 저전력 모드 및 전력 상태 26
5.5.1 저전력 모드 26

부록 A. 드라이버 개정 내역
A.1 버그 수정 및 변경 내용 29

부록 B. PAC193X 디바이스 컨트롤 인터페이스
B.1 소개 31
B.2 PAC193X_HW.H 헤더 파일 31
B.3 PAC193X_INTF.H 헤더 파일 35
각국의 영업소 및 서비스 45

© 2018 Microchip Technology Inc.

서문

고객 안내문

모든 설명서는 시간 경과에 따라 최신 상태가 아니며 본 설명서도 예외는 아닙니다. Microchip 도구와 설명서는 고객의 요구에 맞게 지속적으로 발전하고 있으므로 대화 상자 및/또는 도구 설명이 본 문서와 다를 수 있습니다. 사용 가능한 최신 설명서를 받으려면 당사 웹 사이트(www.microchip.com)를 방문하십시오.

문서는 "DS" 번호로 식별됩니다. 이 번호는 각 페이지 아래쪽의 페이지 번호 앞에 있습니다. DS 번호 매기기 규칙은 "DSXXXXXXXXA"입니다. 여기에서 "XXXXXXXX"는 번호이고 "A"는 문서의 개정 수준입니다.

개발 도구에 대한 최신 정보는 MPLAB® IDE 온라인 도움말을 참조하십시오. Help 메뉴와 Topics를 차례로 선택하여 사용 가능한 온라인 도움말 파일 목록을 엽니다.

소개

이 장에는 PAC193X Microsoft® Windows® 10 드라이버를 사용하기 전에 알아두면 도움이 되는 일반 정보가 들어 있습니다. 이 장에서는 다음과 같은 항목을 다룹니다.

- 문서 레이아웃
- 이 가이드의 표기법
- 권장 참조 자료
- Microchip 웹 사이트
- 고객 지원
- 문서 개정 내역

문서 레이아웃

이 문서에서는 디바이스에서 보고하는 에너지 측정값에 액세스하기 위해 알아야 하는 PAC193X Microsoft® Windows® 10 드라이버 기술적 측면을 설명합니다. 설명서 레이아웃은 다음과 같습니다.

- **1장. "제품 개요"** – PAC193X Microsoft® Windows® 10 드라이버에 대한 중요 정보를 제공합니다.
- **2장. "드라이버 설치"** – PAC193X Microsoft® Windows® 10 드라이버 설치 및 시작에 대한 지침이 들어 있습니다.
- **3장. "디바이스 인스턴스화 및 초기화"** - PAC193X Microsoft® Windows® 10 드라이버 초기 구성에 대한 중요 정보를 제공합니다.
- **4장. "PAC193X 디바이스 인터페이스"** - PAC193X Microsoft® Windows® 10 드라이버 인터페이스에 대한 정보가 들어 있습니다.
- **5장. "PAC193X 드라이버 특징"** - PAC193X Microsoft® Windows® 10 드라이버 기능에 대한 정보가 들어 있습니다.
- **부록 A. "드라이버 개정 내역"** - 드라이버 버그 수정 및 변경 내용이 들어 있습니다.
- **부록 B. "PAC193X 디바이스 컨트롤 인터페이스"** – EMI의 C 정의와 컨트롤 인터페이스 데이터 구조가 들어 있습니다.

이 가이드의 표기법

이 설명서에는 다음과 같은 설명서 표기법이 사용됩니다.

설명서 표기법

설명	표시 내용	예
Arial 글꼴:		
기울임꼴 문자	참조 서적	*MPLAB® IDE 사용자 가이드*
	강조된 텍스트	...는 *유일한*컴파일러...
첫문자가 대문자	창	출력 창
	대화 상자	Settings 대화 상자
	메뉴 선택	Enable Programmer 선택
따옴표	창 또는 대화 상자의 필드 이름	"Save project before build"
닫는 꺾쇠괄호를 포함하며 밑줄이 그어진 기울임꼴 텍스트	메뉴 경로	*File>Save*
굵은 문자	대화 상자 단추	**OK** 클릭
	탭	**Power** 탭 클릭
N‘Rnnnn	verilog 형식의 숫자. 여기에서 N은 전체 자릿수이고, R은 기수이고, n은 자릿수임	4‘b0010, 2‘hF1
꺾쇠괄호 < >로 묶인 텍스트	키보드의 키	<Enter>, <F1> 누르기
Courier New 글꼴:		
일반 Courier New	샘플 소스 코드	`#define START`
	파일 이름	`autoexec.bat`
	파일 경로	`c:\mcc18\h`
	키워드	`_asm, _endasm, static`
	명령줄 옵션	`-Opa+, -Opa-`
	Bit 값	`0, 1`
	상수	`0xFF, ‘A’`
기울임꼴 Courier New	변수 인수	*`file`*`.o`. 여기에서 *`file`*은 유효한 파일 이름으로 지정할 수 있음
대괄호 [ ]	선택적 인수	`mcc18 [options]` *`file`* `[options]`
중괄호와 세로줄 문자: { | }	상호 배타적 인수 선택: OR 선택	`errorlevel {0|1}`
줄임표...	반복되는 텍스트 대체	`var_name [, var_name...]`
	사용자가 제공하는 코드를 나타냄	`void main (void) { ... }`

© 2018 Microchip Technology Inc.

권장 참조 자료

이 사용자 가이드에서는 PAC193X Microsoft® Windows® 10 드라이버 사용 방법을 설명합니다. 아래 나열된 문서도 유용합니다. 다음 Microchip 문서를 권장되는 보충 참조 자료로 사용할 수 있습니다.

- **PAC193X AN – *"PAC193X Microsoft® Windows® 10 드라이버 지원에 대한 통합 정보"*(DS00002534A)**
- **PAC1934 DS – *"PAC1934 누산기 데이터 시트를 통한 전원/에너지 모니터"* (DS20005850A)**
- **PAC1934 UG – *"PAC1934 E 평가 보드 사용자 가이드"*(DS50002673A)**

MICROCHIP 웹 사이트

Microchip은 당사 웹 사이트인 www.microchip.com을 통한 온라인 지원을 제공합니다. 이 웹 사이트는 고객이 파일과 정보를 쉽게 사용할 수 있도록 지원합니다. 좋아하는 인터넷 브라우저를 사용하여 액세스할 수 있으며 웹 사이트에는 다음 정보가 들어 있습니다.

- **제품 지원** – 데이터 시트와 정오표, 애플리케이션 설명 및 샘플 프로그램, 디자인 리소스, 사용자 가이드 및 하드웨어 지원 문서, 최신 소프트웨어 릴리스 및 보관된 소프트웨어
- **일반 지원** – FAQ(질문과 대답), 기술 지원 요청, 온라인 토론 그룹, Microchip 컨설턴트 프로그램 멤버 목록
- **Microchip의 비즈니스** – 제품 선택 및 주문 가이드, 최신 Microchip 보도 자료, 세미나 및 이벤트 목록, Microchip 영업소 목록, 판매 대리점 및 공장 담당자

고객 지원

Microchip 제품 사용자는 여러 채널을 통해 지원을 받을 수 있습니다.

- 판매 대리점 또는 담당자
- 지역 영업소
- FAE(필드 애플리케이션 엔지니어)
- 기술 지원

고객은 해당 판매 대리점, 담당자 또는 FAE에게 지원을 문의해야 합니다. 로컬 영업소를 통해 고객을 지원할 수도 있습니다. 영업소와 위치 목록은 이 문서 뒷면에 나와 있습니다.

다음 웹 사이트를 통해 기술 지원을 이용할 수 있습니다.
http://www.microchip.com/support

문서 개정 내역

개정 A(2017년 9월)

- 본 문서의 최초 릴리스입니다.
- 이 문서의 정보는 PAC193X 드라이버 릴리스 1.0, 1.1 및 1.2에 적용됩니다.

NOTES:

© 2018 Microchip Technology Inc.

1장. 제품 개요

1.1 소개

이 문서에서는 PAC193X 디바이스에서 보고하는 에너지 측정값에 액세스하기 위해 알아야 하는 PAC193X Microsoft® Windows® 10 드라이버 기술적 측면을 설명합니다.

Microsoft® Windows® 10 운영 체제부터 배터리 수명을 극대화하는 최상의 구성 옵션을 알아내기 위해 PC 랩톱이나 기타 이동식 디바이스에 있는 다양한 구성요소의 에너지 소비량을 모니터링할 수 있습니다. Microsoft® Windows® 10의 E3(Energy Estimation Engine) 서비스는 추정 알고리즘이나 시스템의 다양한 전원 레일에 연결된 계량 IC에서 측정한 실시간 에너지를 사용합니다.

계량 IC 디바이스 드라이버가 표준 소프트웨어 인터페이스인 EMI(에너지 측정 인터페이스)를 통해 에너지 측정값을 보고해야 합니다. E3 서비스나 사용자 애플리케이션에서 표준 EMI IOCTL 서비스 세트를 호출하여 에너지 데이터에 액세스할 수 있습니다.

표 1-1: 용어 및 머리글자어 설명

용어	설명
ACPI	Advanced Configuration and Power Interface: 고급 구성 및 전원 인터페이스
ASL	ACPI Source Language: ACPI 소스 언어
BIOS	Basic Input/Output System: 기본 입/출력 시스템
CPU	Central Processing Unit: 중앙 처리 장치
E3	Energy Estimation Engine
EC	Embedded Controller: 임베디드 컨트롤러
EMI	Energy Metering Interface: 에너지 계량 인터페이스
강제 셧다운	ACPI 전원 버튼 재정의를 통해 시스템 G0/S0에서 G2/S5로 전환(예약되지 않음)
GUID	Globally Unique Identifier
I^2C™	Inter-Integrated Circuit™
IC	Integrated Circuit: 집적 회로
IOCTL	I/O 컨트롤. *DeviceIoControl* 시스템 호출은 디바이스에 컨트롤 코드를 보낼 수 있는 범용 인터페이스입니다. 각 컨트롤 코드는 디바이스 드라이버가 수행할 동작을 나타냅니다.
OEM	Original Equipment Manufacturer: 주문자 상표 부착 방식
OS	Operating System: 운영 체제
SMBUS	System Management Bus: 시스템 관리 버스
SPB	Simple Peripheral Bus: 단순 주변 버스
SoC	System on Chip
SpbCx	SPB 프레임워크 확장
UUID	Universal Unique Identifier
WDF	Windows Driver Foundation: Windows 드라이버 파운데이션

NOTES:

© 2018 Microchip Technology Inc.

2장. 드라이버 설치

2.1 사전 요구 사항

대상 시스템에 드라이버를 설치하기 전에 다음 드라이버 종속성을 해결해야 합니다.

- PAC193X 디바이스가 호스트 시스템에 통합됩니다.
- I^2C 버스 컨트롤러 디바이스 드라이버가 SPB 프레임워크 확장(SpbCx) 규격입니다.

2.1.1 호스트 시스템에 PAC193X 통합

2.1.1.1 ACPI ASL

PAC193X 애플리케이션 노트의 지시 사항을 참조하여 Microsoft® Windows® 장치 관리자가 PAC193X 디바이스를 열거하도록 ACPI ASL을 변경합니다.

2.1.1.2 SLOW 핀

PAC 디바이스 VDD 핀은 시스템 대기 또는 전력 끄기 시퀀스로 종료되지 않는 시스템 전원 레일에 연결해야 합니다. SLOW 핀은 시스템 전원이 꺼질 때 HIGH로 구동되는 핀에 연결해야 합니다.

주의
시스템 정상 작동에는 SLOW 핀을 사용하여 디바이스 샘플링 속도를 변경하지 마십시오. 시스템이 S0 상태로 실행 중일 때는 Windows 드라이버가 SLOW 핀 전환을 모니터링하지 않습니다.

PAC193X Microsoft® Windows® 10 드라이버는 다음과 같은 SLOW 핀 사용 사례를 지원합니다.

- 시스템 작동이 중지되거나 운영자 또는 다른 유사 시 대기 메커니즘에 의해 시스템이 강제로 꺼지는 경우 유사 시 대기 메커니즘을 통해 PAC 디바이스가 Slow 샘플링 모드로 전환됩니다.
- 시스템 전원이 다시 켜지면 하이-로우 저장에서 읽기 가능한 디바이스로 느린 전환에 의해 시스템이 꺼져 있는 동안 축적된 데이터가 등록됩니다.

2.1.2 SpbCx 규격의 I2C 버스 컨트롤러 디바이스 드라이버

I_2C 버스에 액세스하고 PAC 디바이스와 통신하기 위해 PAC193X Microsoft® Windows® 10 드라이버는 Windows® SpbCx 프레임워크에 의해 구현된 표준 인터페이스와 프로토콜을 사용합니다. 따라서 I^2C 버스 컨트롤러에 SpbCx 프레임워크에 맞는 Microsoft® Windows® 디바이스 드라이버가 있어야 합니다. 일반적으로 I^2C 컨트롤러 제조업체에서 이러한 드라이버를 만들어 Microsoft® Windows® 설치 또는 Microsoft® Windows® 업데이트를 통해 제공합니다.

2.2 드라이버 설치

PAC193X Microsoft® Windows® 10 드라이버 설치 옵션이 여러 개 있습니다.

- Microsoft® Windows® 업데이트를 통해 설치
- 장치 관리자와 Microchip 웹 사이트에서 다운로드한 드라이버 패키지를 사용하여 설치
- Microchip 웹 사이트에서 다운로드한 드라이버 패키지에 포함된 드라이버 설치 관리자 애플리케이션인 `PAC193xDriverInstaller_x86` 또는 `PAC193xDriverInstaller_x64` 실행
- DevCon 유틸리티와 Microchip 웹 사이트에서 다운로드한 드라이버 패키지 사용

2.2.1 Microsoft® Windows® 업데이트 또는 장치 관리자를 사용하여 설치

Microsoft® Windows® 업데이트 또는 장치 관리자를 통한 드라이버 설치에 대한 자세한 내용은 Microsoft® 문서, "Microsoft® Windows 10의 드라이버 설치 및 업데이트 방법"을 참조하십시오.

2.2.2 드라이버 설치 관리자 애플리케이션

`PAC193xDriverInstaller_x86` 및 `PAC193xDriverInstaller_x64` 드라이버 설치 관리자 애플리케이션은 Microchip EULA(End User License Agreement) 대화 상자를 먼저 표시하는 클릭 방문 애플리케이션입니다. 사용 약관을 수락하면 드라이버가 자동으로 설치됩니다.

> **참고:** 대상 시스템에 설치된 운영 체제에 따라 32비트 (`PAC193xDriverInstaller_x86`) 또는 64비트 (`PAC193xDriverInstaller_x64`) 설치 관리자 애플리케이션을 선택하십시오.

2.2.3 DevCon 유틸리티를 사용하여 설치

DevCon 유틸리티는 WDK(Windows® 드라이버 키트)와 함께 제공됩니다.

1. 대상 시스템 OS(32비트 또는 64비트)에 따라 드라이버 패키지에서 다음 파일의 압축을 풉니다.

- `PAC193x.sys`
- `PAC193x.inf`
- `PAC193x.cat`

2. 대상 시스템에 드라이버 파일과 DevCon 유틸리티를 복사합니다.
3. 관리자 권한 명령 프롬프트를 열고 드라이버 파일을 복사한 폴더로 디렉터리를 바꾼 후 다음 명령을 실행합니다.

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

2.2.3.1 설치 확인

PAC193X Microsoft® Windows® 10 드라이버 설치가 성공적으로 완료되었는지 확인하려면 장치 관리자를 열고 PAC193X 디바이스가 "에너지 측정 장치" 범주 아래에 나열되고 제대로 작동하는 것으로 보고되는지 확인합니다. 그림 2-1을 참조하십시오.

© 2018 Microchip Technology Inc.

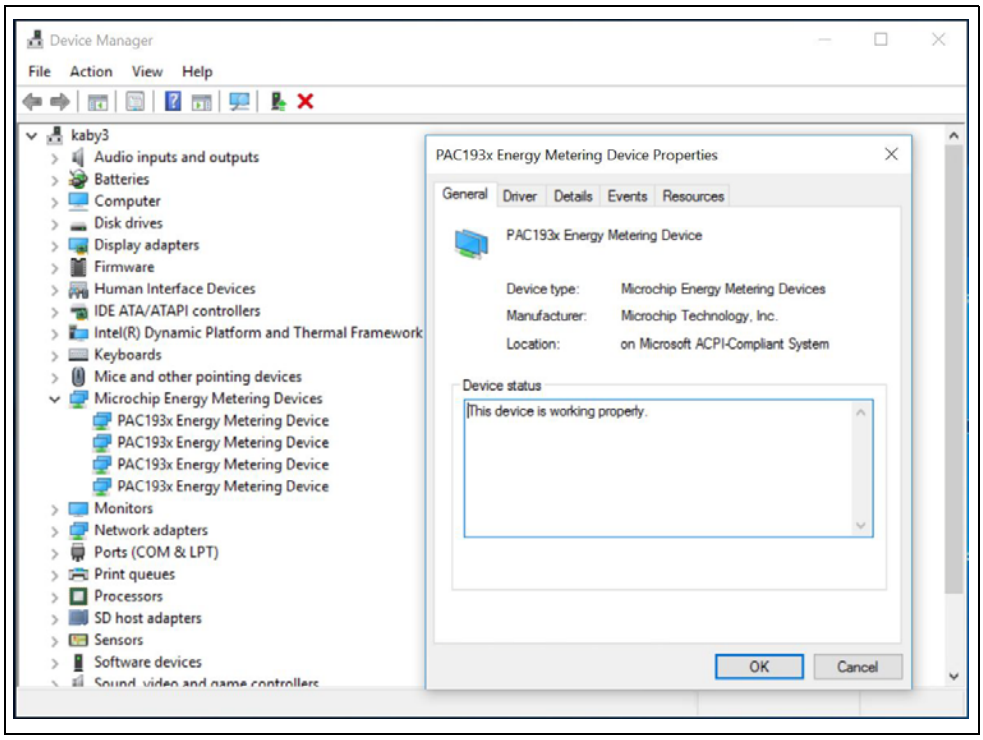

그림 2-1: *PAC193X 디바이스가 표시된 장치 관리자*

장치 관리자를 열려면

1. 바탕 화면 왼쪽 아래에 있는 **시작** 버튼을 클릭하고 검색 상자에 장치 관리자를 입력한 다음 메뉴에서 **장치 관리자**를 클릭합니다.
2. <Windows>+<X>를 눌러 바로 가기 메뉴를 열고 **장치 관리자**를 선택합니다.

NOTES:

© 2018 Microchip Technology Inc.

3장. 디바이스 인스턴스화 및 초기화

3.1 시작하기

시스템 I²C 버스에 연결된 PAC193X 디바이스는 버스 드라이버 역할을 하는 ACPI(Advanced Configuration and Power Interface)에 의해 `ACPI\MCHP1930\<_UID>`로 열거됩니다. 여기에서 `_UID`는 ACPI ASL 코드의 각 디바이스에 대해 정의된 값입니다. PAC193X에 대한 ACPI ASL 코드 정보가 PAC193x 애플리케이션 노트에 나열되어 있습니다.

PAC193X Microsoft® Windows 10® 드라이버는 버스 드라이버가 하드웨어 ID *ACPI\MCHP1930*의 디바이스를 열거할 때마다 INF에 따라 PnP 관리자에 의해 인스턴스화되는 표준 KMDF(Kernel-Mode Driver Framework) 기능 드라이버입니다.

디바이스 초기화 중 PAC193X Microsoft® Windows 10® 드라이버는 고유 ID(UUID) *033771E0-1705-47B4-9535-D1BBE14D9A09*에 의해 식별되는 디바이스의 ACPI BIOS_DSM 메서드를 평가합니다. _DSM 메서드는 다음과 같은 정보를 반환합니다.

- 각 채널에 대한 분로 저항기(R_{SENSE}) 값(밀리옴 단위)
  - 저항기 값은 에너지 계산에 사용되며 정수여야 합니다. 밀리옴의 분수는 현재 드라이버 버전에서 지원되지 않습니다.
  - R_{SENSE} = 0은 전원 레일에 연결되지 않은 채널을 나타냅니다. 드라이버가 이 채널에 대한 데이터를 보고하지 않습니다.
- 채널에서 모니터링하는 전원 레일에 대해 Windows에서 지정하는 이름
  - 드라이버가 명명된 모든 채널에 대한 EMI 디바이스 인터페이스를 생성합니다. 자세한 내용은 Microsoft 문서, "에너지 측정 인터페이스"를 참조하십시오. Windows 전원 레일 분류와 이름이 호환되는 경우 Windows E3 서비스가 EMI 인터페이스에서 내보낸 데이터를 사용할 수 있습니다.
  - 채널 이름이 빈 문자열인 경우 드라이버가 EMI를 생성하지 않습니다. 해당 채널에 대해 디바이스에 축적된 데이터가 PAC193X 컨트롤 인터페이스를 통해 계속 수집, 처리 및 보고됩니다. PAC 디바이스마다 인터페이스가 하나씩 있고 사용자는 각 디바이스 인터페이스를 만든 드라이버를 확인할 수 있습니다.

3.2 디바이스 초기 구성

PAC193X Microsoft® Windows 10® 드라이버 기본 디바이스 POR 구성과 동일한 구성을 사용합니다.

- I²C/SMBus 인터페이스가 I²C로 구성됩니다. SMBus는 현재 Microsoft® Windows®에서 지원되지 않습니다. 드라이버에서 통신 세팅 변경을 허용하지 않습니다.
- 샘플링 속도(SRN): 1024Hz. 모든 디바이스 채널이 EMI 채널이 아닌 경우에만 드라이버가 샘플링 속도를 변경하는 옵션을 사용자에게 제공합니다.
- SLOW/ALERT 핀이 SLOW로 구성됩니다. 드라이버에서 구성 변경을 허용하지 않습니다.

- SLOW 신호 전환이 제한된 REFRESH를 트리거합니다. 모든 결과 레지스터가 업데이트되고 누산기와 누산기 카운트가 리셋되어 SLOW 핀에 의해 트리거된 제한된 REFRESH가 정의됩니다. 그러나 디바이스 구성 변경은 허용되지 않습니다.
- 모든 디바이스 채널이 활성화됩니다. 이들 채널에 대해 열려 있는 EMI 인터페이스가 없는 경우 사용자가 상태를 변경할 수 있습니다.
- 모든 채널이 기본적으로 단방향입니다. 사용자가 V_{BUS}, V_{SENSE} 또는 둘 다에 대한 극성 구성을 변경할 수 있습니다. 그에 따라 드라이버 산술이 조정됩니다. 부호 있는 측정값을 변경하면 이러한 측정값에 대한 1비트 분해능도 손실됩니다.

또한 드라이버가 실리콘(개정 ID = 0 x 01)의 초기 개정에 영향을 주는 오류에 대한 해결 방법을 실행하기 위해 필요한 구성 작업을 수행합니다. 개정 ID에 대한 자세한 내용은 PAC193X 데이터 시트를 참조하십시오.

© 2018 Microchip Technology Inc.

4장. PAC193X 디바이스 인터페이스

4.1 소개

디바이스 인터페이스에는 사용자 애플리케이션이 디바이스로 I/O(입/출력) 요청을 제어하는 데 사용하는 심볼 링크 이름이 들어 있습니다. 드라이버에서 인터페이스를 만들 때 인터페이스 클래스 GUID(인터페이스 유형 식별자)도 연결합니다.

사용자 애플리케이션에서 "*SetupDiEnumDeviceInterfaces*" 함수 호출을 사용하여 특정 GUID 클래스에 속하는 인터페이스를 검색한 다음 "*SetupDiGetDeviceInterfaceDetail*" 함수 호출을 사용하여 디바이스의 심볼 링크 이름을 구합니다. 이 단락에서 설명하는 2개의 "SetupDi…" 함수는 `setupapi.lib`로 구현되었으며 Microsoft® Windows 2000부터 사용 가능합니다.

그 다음으로, 사용자 애플리케이션이 "*CreateFile*" 커널 함수에 심볼 링크 이름을 전달하여 I/O를 위한 디바이스를 열고 할당된 파일 핸들을 받습니다. 사용자 애플리케이션은 디바이스에서 요청된 동작을 식별하는 IOCTL 코드(입/출력 컨트롤 코드)를 지정하여 "*DeviceIoControl*" 커널 함수와 파일 핸들로 디바이스와 통신할 수 있습니다. 디바이스 드라이버는 요청을 디바이스별 동작으로 변환하고 결과를 다시 요청자에게 반환합니다.

이 장에서는 PAC193X Microsoft® Windows® 10 드라이버에서 생성하는 각 인터페이스 유형에 대해 구현되는 클래스 GUID 코드와 IOCTL 코드를 설명합니다.

- 에너지 측정 인터페이스
- 디바이스 컨트롤 인터페이스

4.2 에너지 측정 인터페이스

ACPI ASL에 비어 있지 않은 Windows 레일 이름이 할당된 각 디바이스 채널에 대해 EMI 인터페이스가 생성됩니다. EMI 인터페이스에 대한 자세한 내용은 Microsoft 문서, "에너지 측정 인터페이스"를 참조하십시오. Microsoft® Windows® 드라이버 키트와 함께 배포되는 `emi.h` 파일에서 정의 정보를 사용할 수 있습니다.

EMI 디바이스 인터페이스의 GUID는 다음과 같습니다.

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

표 4-1: EMI IOCTL

IOCTL 이름	설명
`IOCTL_EMI_GET_MEASUREMENT`	이 항목은 현재 에너지 측정값과 측정 시간을 검색합니다. • 입력: 없음 • 출력: `EMI_MEASUREMENT_DATA`
`IOCTL_EMI_GET_METADATA`	이 항목은 디바이스에서 EMI 메타데이터를 검색합니다. • 입력: 없음 • 출력: `EMI_METADATA`

표 4-1: EMI IOCTL (계속)

IOCTL 이름	설명
IOCTL_EMI_GET_METADATA_SIZE	이 항목은 IOCTL_EMI_GET_METADATA 요청을 보내 디바이스에서 가져오는 EMI 메타데이터 객체의 크기를 검색합니다. • 입력: 없음 • 출력: EMI_METADATA_SIZE
IOCTL_EMI_GET_VERSION	이 항목은 디바이스에서 지원하는 EMI 인터페이스의 현재 버전을 검색합니다. • 입력: 없음 • 출력: EMI_VERSION

표 4-2: EMI 열거 및 구조

항목	유형	설명
EMI_MEASUREMENT_DATA	출력 구조	IOCTL_EMI_GET_MEASUREMENT에서 보고하는 에너지 측정값과 측정 시간입니다. • 에너지 단위:1pWh • 에너지 값 및 타임스탬프 값: 64비트 무부호. • 타임스탬프 단위: 100ns.
EMI_MEASUREMENT_UNIT	열거	이 항목은 IOCTL_EMI_GET_MEASUREMENT에서 보고할 수 있는 사용 가능한 에너지 측정값 단위를 나타냅니다. • EMI V1은 1pWh 단위만 구현합니다.
EMI_METADATA	출력 구조	다음에 대한 정보: • 측정 디바이스: 하드웨어 모델 및 개정 • 측정된 전력 레일: 레일 이름 IOCTL_EMI_GET_METADATA에서 보고
EMI_METADATA_SIZE	출력 구조	다음에서 보고하는 EMI_METADATA 구조의 크기 IOCTL_EMI_GET_METADATA_SIZE
EMI_VERSION	출력 구조	이 디바이스에서 지원하는 EMI 버전, IOCTL_EMI_GET_VERSION에서 보고 • 현재 EMI V1 사양만 정의되고 지원합니다.

부록 B. "PAC193X 디바이스 컨트롤 인터페이스"에 EMI 인터페이스 데이터 구조의 C 정의가 들어 있습니다.

© 2018 Microchip Technology Inc.

4.3 디바이스 컨트롤 인터페이스

드라이버가 열거된 PAC193X 디바이스마다 하나씩 컨트롤 인터페이스를 생성합니다. 이 인터페이스는 IOCTL_PAC193x_* 요청을 수락하고, EMI 인터페이스보다 훨씬 더 많은 데이터와 디바이스 구성 옵션에 대한 액세스 권한을 사용자에게 제공합니다.

- 모든 측정값(예: 축적된 에너지와 함께 V_{BUS}, V_{SENSE})
- 비 EMI 채널을 포함한 모든 채널에 대한 데이터
- 각 디바이스 및 채널에 대한 구성 옵션(샘플링 속도, 채널 극성, 채널 ON/OFF)

주의

드라이버에서 일시적 또는 영구적으로 일부 디바이스 또는 채널 구성 옵션을 차단합니다.

- **사용자는 SMB/I²C 통신 옵션이나 SLOW/ALERT 핀 구성 옵션에 액세스할 수 없습니다.**
- **구성 변경의 영향을 받는 채널에 대해 정의된 EMI 인터페이스가 없는 경우에만 디바이스 샘플링 속도와 채널 ON/OFF 컨트롤 옵션이 허용됩니다.**

컨트롤 인터페이스의 GUID는 다음과 같습니다.

4166FE9F-A865-4314-8942-7C12ABA290F6

표 4-3: 디바이스 컨트롤 IOCTL

IOCTL 이름	설명
IOCTL_PAC193X_GET_DEVICE_INFO_SIZE	디바이스 정보 데이터 구조를 포함해야 하는 버퍼에 대한 크기 할당 요건을 반환합니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_GET_DEVICE_INFO_SIZE
IOCTL_PAC193X_GET_DEVICE_INFO	디바이스 버전과 채널 연결(레일 이름, 감지 저항기, 연결 상태)에 대한 정보를 반환합니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_GET_DEVICE_INFO
IOCTL_PAC193X_GET_CTRL	디바이스 레지스터 CTRL, CTRL_ACT, CTRL_LAT의 내용을 반환합니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_GET_CTRL
IOCTL_PAC193X_SET_CTRL	CTRL 레지스터에서 일부 디바이스 구성 bit인 SRN, SLEEP 및 SING에 대한 사용자 제한 컨트롤을 제공합니다. 디바이스에 열려 있는 EMI 채널이 있는 경우 드라이버에서 구성 요청을 거부합니다. • 입력 버퍼: PAC193X_SET_CTRL • 출력 버퍼: 없음(0바이트 반환됨)
IOCTL_PAC193X_GET_MEASUREMENTS	모든 채널에 대해 데이터 측정값, 타임스탬프 및 샘플링 시 디바이스 구성을 반환합니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_MEASUREMENTS
IOCTL_PAC193X_GET_CHANNEL_CFG	I²C/SMB와 채널 ON/OFF 구성 레지스터인 CHANNEL_DIS, CHANNEL_DIS_ACT, CHANNEL_DIS_LAT의 내용을 반환합니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_GET_CHANNEL_CFG

표 4-3: 디바이스 컨트롤 IOCTL (계속)

IOCTL 이름	설명
IOCTL_PAC193X_SET_CHANNEL_CFG	CHANNEL_DIS 레지스터인 CH1, CH2, CH3 및 CH4의 일부 구성 bit에 대한 사용자 제한 컨트롤을 제공합니다. 드라이버가 I²C/SMB 구성 bit인 TIMEOUT 및 BYTECOUNT, NOSKIP 변경 요청을 무시합니다. 드라이버가 포인터 스킵 구성 bit인 NOSKIP 변경 요청을 무시합니다. 드라이버가 열려 있는 EMI 채널이 있는 채널에 대한 변경 값을 무시합니다. • 입력 버퍼: PAC193X_SET_CHANNEL_CFG • 출력 버퍼: 없음(0바이트 반환됨)
IOCTL_PAC193X_GET_NEG_POWER	채널 극성 구성 레지스터 NEG_PWR, NEG_PWR_ACT, NEG_ PWR_LAT의 내용을 반환합니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_GET_NEG_POWER
IOCTL_PAC193X_SET_NEG_POWER	채널 극성 구성 레지스터인 NEG_PWR에 대한 사용자 컨트롤을 제공합니다. 채널 극성은 드라이버 초기화 시 단극으로 세팅되지만 EMI 상태에 관계없이 언제든지 사용자가 변경할 수 있습니다. • 입력 버퍼: PAC193X_SET_NEG_POWER • 출력 버퍼: 없음(0바이트 반환됨)
IOCTL_PAC193X_GET_OVERFLOW	CTRL_ACT 및 CTRL_LAT에서 OVF bit의 상태를 반환합니다. 입력 플래그의 값에 따라 이 요청이 CTRL 레지스터에서 OVF 플래그를 클리어(0으로 리셋됨)할 수도 있습니다. • 입력 버퍼: PAC193X_GET_OVERFLOW • 출력 버퍼: PAC193X_GET_OVERFLOW
IOCTL_PAC193X_DATA_REFRESH	디바이스가 누산기를 리셋하지 않고 측정값을 보고할 수 있습니다. REFRESH_V 디바이스 명령을 사용하여 소프트웨어 누산기 업데이트를 수행합니다. • 입력 버퍼: 없음 • 출력 버퍼: 없음(0바이트 반환됨)
IOCTL_PAC193X_DATA_REFRESH_RESET	REFRESH 디바이스 명령을 사용하여 소프트웨어 누산기 업데이트를 수행합니다. 입력 플래그의 값에 따라 이 요청이 열려 있는 EMI 인터페이스가 없는 채널에 대한 소프트웨어 누산기를 클리어(0으로 리셋됨)할 수도 있습니다. • 입력 버퍼: PAC193X_REFRESH_FLAGS • 출력 버퍼: 없음(0바이트 반환됨)

© 2018 Microchip Technology Inc.

표 4-3: 디바이스 컨트롤 IOCTL (계속)

IOCTL 이름	설명
IOCTL_PAC193X_DATA_REFRESH_RESET_GLOBAL	시스템의 모든 PAC193X 디바이스에 대한 소프트웨어 누산기의 글로벌 업데이트를 수행합니다. 입력 플래그의 값에 따라 이 요청이 열려 있는 EMI 인터페이스가 없는 채널에 대한 소프트웨어 누산기를 클리어(0으로 리셋됨)할 수도 있습니다. Microsoft® Windows® SpbCx 프레임워크는 I²C 일반 호출 주소(0 x 00)를 지원하지 않기 때문에 드라이버가 REFRESH_G 디바이스 명령을 사용하여 이 요청을 구현할 수 없습니다. 대신, 드라이버가 시스템의 모든 디바이스에 순차 REFRESH 명령을 보내는 이 요청을 구현합니다. • 입력 버퍼: PAC193X_REFRESH_FLAGS • 출력 버퍼: 없음(0바이트 반환됨)
IOCTL_PAC193X_GET_LAST_SLOW_DATA	모든 채널에 대해 다른 무전력 상태에서 S0으로 시스템 상태 전환에 의해 발생하는 마지막 SLOW 신호 HL 전환(SLOW에서 트리거하는 제한된 리프레시(refresh))에서 샘플링하는 모든 데이터 측정값을 반환합니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_LAST_SLOW_DATA
IOCTL_PAC193X_GET_SLOW	SLOW 레지스터의 값을 반환합니다. SLOW 레지스터의 구성은 드라이버만으로 컨트롤됩니다. • 입력 버퍼: 없음 • 출력 버퍼: PAC193X_GET_SLOW

표 4-4: 디바이스 컨트롤 구조

구조 이름	유형	설명
PAC193X_GET_DEVICE_INFO_SIZE	출력	다음에서 보고하는 PAC193X_GET_DEVICE_INFO 구조의 크기 IOCTL_PAC193X_GET_DEVICE_INFO
PAC193X_GET_DEVICE_INFO	출력	IOCTL_PAC193X_GET_DEVICE_INFO에서 반환하는 디바이스 및 채널 정보 • 제조업체 ID, 제품 ID, 실리콘 개정 • 연결된 채널 수 및 해당 ON/OFF 상태 • 채널 레일 이름, 감지 저항기 값, EMI 인터페이스 상태별(생성 여부) *주의*: 이 구조의 크기는 디바이스의 채널 수와 채널 이름의 크기에 따라 가변적입니다.
PAC193X_GET_CTRL	출력	다음에서 반환하는 컨트롤 레지스터 내용 IOCTL_PAC193X_GET_CTRL • CTRL • CTRL_ACT • CTRL_LAT
PAC193X_SET_CTRL	입력	사용자가 IOCTL_PAC193X_SET_CTRL에 제공해야 하는 CTRL 레지스터 값

표 4-4: 디바이스 컨트롤 구조

구조 이름	유형	설명
PAC193X_MEASUREMENTS	출력	다음에서 반환하는 데이터 구조 디바이스 레지스터 값을 포함하는 IOCTL_PAC193X_GET_MEASUREMENTS: • ACC_COUNT • VPOWERn_ACC • VBUSn, VBUSn_AVG • VSENSEn, VSENSEn_AVG • VPOWERn • NEG_PWR_LAT • CTRL_LAT • CHANNEL_DIS_LAT 및 디바이스 드라이버에서 구현하는 소프트웨어 레지스터: • 마지막 샘플 보고서 타임스탬프 • 소프트웨어 누산기 카운트 • 무부호 소프트웨어 누산기 • 부호 소프트웨어 누산기 *주의*: 특정 채널에 대해 양방향 전압 및/또는 전류가 활성화되면 보고된 값의 이진 형식은 2의 보수로 표시됩니다 (MSB는 부호 bit임). 후속 처리를 위해 필요한 데이터 유형으로 보고된 값을 캐스트합니다. 드라이버가 개정 ID = 0 x 01의 디바이스에 영향을 주는 VPOWER_ACC 및 VPOWERN 오류에 대한 해결 방법을 실행합니다.
PAC193X_GET_ CHANNEL_CFG	출력	IOCTL_PAC193X_GET_CHANNEL_CFG에서 반환하는 채널 컨트롤 레지스터 값: • CHANNEL_DIS, • CHANNEL_DIS_ACT, • CHANNEL_DIS_LAT
PAC193X_SET_ CHANNEL_CFG	입력	사용자가 다음에 제공해야 하는 CHANNEL_DIS 레지스터 값 IOCTL_PAC193X_SET_CHANNEL_CFG
PAC193X_GET_NEG_ POWER	출력	다음에서 반환하는 채널 극성 구성 레지스터 IOCTL_PAC193X_GET_NEG_POWER: • NEG_PWR • NEG_PWR_ACT • NEG_PWR_LAT
PAC193X_SET_NEG_ POWER	출력	사용자가 다음에 제공해야 하는 NEG_PWR 레지스터 값 IOCTL_PAC193X_SET_NEG_POWER
PAC193X_GET_OVERFLOW	입/출력	1개의 입력 및 2개의 출력 플래그를 포함하는 IOCTL_PAC193X_GET_OVERFLOW입/출력 구조: • 0이 아닌 경우 입력 플래그가 CTRL에서 OVF bit 리셋을 요청합니다. • 출력 플래그가 CTRL_ACT 및 CTRL_LAT에서 OVF bit를 반환합니다.

© 2018 Microchip Technology Inc.

표 4-4: 디바이스 컨트롤 구조

구조 이름	유형	설명
PAC193X_ REFRESH_FLAGS	입력	다음에 대한 입력 구조 IOCTL_PAC193X_DATA_REFRESH_RESET 및 IOCTL_PAC193X_DATA_REFRESH_RESET_ 1개의 입력 플래그를 포함하는 GLOBAL 요청. PAC193X_REFRESH_FLAG_RESET_ 누산기(1로 정의됨)로 세팅된 플래그 값은 요청에 소프트웨어 누산기도 클리어하라고 지시합니다.
PAC193X_LAST_SLOW _DATA	출력	모든 채널에 대해 다른 무전력 상태에서 S0으로 시스템 상태 전환에 의해 발생하는 SLOW 신호 HL 전환 (SLOW에서 트리거하는 제한된 리프레시(refresh))에서 샘플링하는 디바이스 레지스터 값을 포함하는 IOCTL_PAC193X_GET_LAST_SLOW_DATA에서 반환된 데이터 구조: • ACC_COUNT • VPOWERn_ACC • VBUSn, VBUSn_AVG • VSENSEn, VSENSEn_AVG • VPOWERn • NEG_PWR_LAT • CTRL_LAT • CHANNEL_DIS_LAT
PAC193X_GET_SLOW	출력	IOCTL_PAC193X_GET_SLOW에서 반환하는 SLOW 레지스터 값

부록 B. “PAC193X 디바이스 컨트롤 인터페이스”에는 컨트롤 인터페이스 데이터 구조의 C 정의가 들어 있습니다.

NOTES:

© 2018 Microchip Technology Inc.

5장. PAC193X 드라이버 특징

5.1 드라이버 특징

이 섹션에서는 드라이버에서 보고하는 데이터를 처리하는 사용자 애플리케이션에 대한 설계 결정에 영향을 줄 수 있는 드라이버 기능에 대한 정보를 제공합니다.

- 복수 I/O 요청
- EMI(에너지 측정 인터페이스) 및 디바이스 컨트롤 인터페이스
- 에너지 소프트웨어 누산기
- 저전력 모드 및 전력 상태

5.2 복수 I/O 요청

Windows® E3 모듈과 다른 여러 사용자 애플리케이션이나 동일한 애플리케이션의 여러 인스턴스가 데이터 보고서 요청 또는 디바이스 구성 명령을 동시에 발급하거나 실행할 수 있습니다. 드라이버가 모든 요청을 큐에 대기시키고 한 번에 하나씩 RTC(Run-To-Completion) 방식으로 순차 실행하여 I^2C 트랜잭션의 일관성을 유지합니다.

5.3 EMI(에너지 측정 인터페이스) 및 디바이스 컨트롤 인터페이스

각 PAC 디바이스에 대해 PAC193X Microsoft® Windows® 10 드라이버가 ACPI ASL 코드에 비어 있지 않은 EMI 전력 레일 이름이 할당되어 있는 연결된 각 채널마다 1개의 컨트롤 인터페이스와 0개 이상의 EMI 인터페이스를 생성합니다.

EMI 인터페이스는 양의 에너지 축적과 마지막 측정의 타임스탬프만 보고합니다.

컨트롤 인터페이스는 PAC193X 디바이스에서 수행한 모든 측정을 보고합니다. EMI 인터페이스가 정의되어 있지 않은 경우에도 EMI 인터페이스에서 보고하는 타임스탬프 수치도 여기에 포함됩니다. 이 경우 모든 EMI 레일 이름은 빈 문자열("")입니다.

Windows E3(Energy Estimation Engine)은 EMI 인터페이스에서 보고하는 에너지 수치만 수집합니다. E3이 제대로 데이터를 사용하려면 EMI 전력 레일 이름이 Microsoft® 명명 규칙을 준수해야 합니다.

사용자 애플리케이션은 EMI, 컨트롤 또는 두 인터페이스 유형 모두에서 데이터를 수집할 수 있습니다.

- EMI 인터페이스에는 Microsoft에서 정의한 컨트롤 코드와 데이터 구조가 있습니다. 이들을 통해 애플리케이션이 모든 EMI 규격의 측정 디바이스 공급업체로부터 데이터를 수집할 수 있습니다.
- 컨트롤 인터페이스는 Microchip에서 정의한 컨트롤 코드와 데이터 구조 사용에 대한 훨씬 더 많은 정보를 제공합니다.

또한 컨트롤 인터페이스는 디바이스 구성 레지스터에 대한 액세스도 제공합니다. 단, 다음과 같은 몇 가지 제약이 따릅니다.

- I^2C 인터페이스 파라미터, 포인터 스킵, ALERT 및 SLOW 핀 구성을 비롯한 몇몇 디바이스 구성 옵션은 드라이버로만 컨트롤됩니다.

- 구성 변경의 영향을 받는 채널에 대해 열려 있는 EMI 인터페이스가 없는 경우에만 샘플링 속도와 채널 활성화/비활성화 상태 변경이 수락됩니다.

5.4 에너지 소프트웨어 누산기

드라이버에서 각 PAC 디바이스에 대해 다음을 위한 에너지 소프트웨어 누산기 세트를 구현합니다.

- 디바이스 전력 제품 누산기 값, R_{SENSE} 값 및 샘플링 속도로 드라이버가 계산하는 실제 에너지 수치를 측정합니다. 이는 EMI 및 컨트롤 인터페이스를 통해 보고되는 에너지 값(pWh(pico-watt-hour) 단위)입니다.
- 디바이스 성능을 이상의 에너지 축적 시간 한도를 늘립니다. 소프트웨어 누산기의 64비트 분해능이 17분의 최소 축적 시간(레지스터 포화까지)을 약 2년(소프트웨어 누산기 오버플로까지)으로 늘려야 합니다.

디바이스 샘플 카운터 레지스터에는 드라이버에 의해 구현된 샘플 카운터 컴패니언이 있습니다.

드라이버는 에너지 보고서 요청을 받을 때마다 디바이스 누산기 레지스터를 읽고, 소프트웨어 누산기를 업데이트하고, 디바이스 누산기를 지웁니다.

예를 들어, Windows E3 모듈은 몇 분 간격으로 EMI 에너지를 수집합니다. 에너지 보고서 요청을 위한 EMI 인터페이스와 사용자 애플리케이션이 없더라도 드라이버에 내부 와치독 타이머가 있으며 드라이버가 주기적으로 소프트웨어 누산기를 업데이트하고 디바이스 누산기를 클리어할 수 있습니다. 현재 구현에서 와치독 기간은 4분으로 세팅되어 있습니다.

5.5 저전력 모드 및 전력 상태

PAC193X Microsoft® Windows® 10 드라이버에 대해 설명된 사용 사례에서는 전원이 시스템에서 완전히 제거되지 않는 한 호스트 운영 체제의 전력 상태에 관계없이 PAC 디바이스가 전력을 수신한다고 가정합니다. 자세한 내용은 **2장. "드라이버 설치"** 및 PAC193X 애플리케이션 노트를 참조하십시오. 전원이 제거되지 않는 한 PAC 디바이스에서 에너지를 축적할 수 있기 때문에 드라이버에서 처리해야 하는 여러 전력 모드 요소가 있습니다.

- PAC 디바이스가 POR(파워온 리셋)을 거쳤는지 확인하므로 디바이스가 초기화되지 않습니다. 이 경우 디바이스 레지스터 내용을 신뢰할 수 없고 소프트웨어 누산기에 데이터를 추가해야 합니다.
- 하이에서 로우로 SLOW 핀 전환 시 저장된 데이터를 수집합니다.
  - 시스템이 종료되거나(S5), 최대 절전 모드로 전환되거나(S4), 하이브리드 슬립 모드로 전환됩니다(전력은 유지됨).
- 시스템이 자체 전력 소모를 줄이기 위해 대기(S1-S3) 또는 연결된 대기 상태(CS)로 전환될 때마다 저전력 모드로 PAC 디바이스를 구성합니다.

5.5.1 저전력 모드

운영 체제가 PAC 디바이스를 비D0 전력 상태로 전환하면 드라이버가 샘플링 속도를 최저 주파수(8Hz)로 재구성하여 PAC 디바이스의 자체 전력 소모량을 줄입니다. 드라이버가 디바이스를 슬립 모드로 전환하지 않습니다. 따라서 PAC 디바이스 전력 상태가 대부분의 주변 디바이스에 대해 꺼짐을 의미하는 D3 상태일 때도 디바이스에서 데이터를 계속 축적합니다.

© 2018 Microchip Technology Inc.

CS 시스템에서 언제 저전력 모드로 전환할지 확인하기 위해 드라이버는 다음 전력 설정 콜백에도 등록합니다.

- GUID_CONSOLE_DISPLAY_STATE
- GUID_GLOBAL_USER_PRESENCE

드라이버가 저전력 모드를 끝내면 디바이스의 샘플링 주파수가 `IOCTL_PAC193X_SET_CTRL`에서 마지막으로 세팅된 샘플링 속도로 돌아갑니다. 마지막 POR 이후 샘플링 속도가 세팅되지 않은 경우에는 최고 속도(1024sps 샘플/초)로 돌아갑니다.

다음 조건에 맞으면 저전력 모드가 끝납니다.

- 디바이스가 D0 상태임
- 콘솔 디스플레이 상태가 **PowerMonitorOn**이거나 사용자 현재 상태가 **PowerUserActive**임

일반적으로 이는 D0 상태로 전환되어도 드라이버가 즉시 저전력 모드를 종료하지 않음을 의미합니다.

표 5-1에서는 드라이버가 저전력 모드로 있어야 하는 경우를 설명합니다.

표 5-1: 저전력 모드 진리표

D0 상태임(작동 중)	PowerMonitorOff	PowerUserInactive	저전력 모드
—	X	X	X
—	—	X	X
—	X	—	X
—	—	—	X
X	X	X	X
X	—	X	—
X	X	—	—
X	—	—	—

NOTES:

© 2018 Microchip Technology Inc.

부록 A. 드라이버 개정 내역

A.1 버그 수정 및 변경 내용

표 A-1: 드라이버 릴리스 내역

릴리스	날짜	설명
최초 공개	2017년 9월 7일	• LAST_SLOW_DATA에 래치 레지스터가 3개 추가됨 • 디바이스에 공개 채널이 있는지에 관계없이 SET_NEG_POWER IOCTL로 레지스터 세팅 가능 • SET_CHANNEL_CFG IOCTL로 개별 사설 채널 활성화 또는 비활성화 가능 (디바이스에 공개 채널이 없어도 됨) • 이제 사용자가 SET_CHANNEL_CFG IOCTL로 ByteCount 및 Timeout 필드를 세팅할 수 없음 • 이전 주기에서 TotalAccumulationCount가 캐시되어 REFRESH_V가 제대로 지원됨 • REFRESH_RESET IOCTL에 대해 클리어 옵션 지정 시 SoftwareAccumulatorCount가 클리어됨 • Single Shot 모드 지원됨 • IOCTL이 수신될 때가 아니라 하드웨어 누산기가 클리어될 때만 폴링 타이머가 초기화됨 • 리소스 균형 리밸런싱 문제가 수정됨(HLK 중 문제가 있었음) • 2번째 하드웨어 개정에 대한 지원 추가됨 • PACTest가 IOCTL 변경 내용을 수용하고 표시함

NOTES:

© 2018 Microchip Technology Inc.

부록 B. PAC193X 디바이스 컨트롤 인터페이스

B.1 소개

이 부록에 설명된 예제 C 헤더 파일에 PAC193X Microsoft® Windows 10® 드라이버에서 생성하는 PAC193X 디바이스 컨트롤 인터페이스가 정의되어 있습니다.

- PAC193x_HW.h: PAC193X 디바이스 관련 정의가 들어 있습니다.
- PAC193x_INTF.h: 인터페이스와 IOCTL 정의가 들어 있습니다.

B.2 PAC193X_HW.H HEADER FILE

```
////////////////////////////////////////////////////////////////////////////   //
//      (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
```


© 2018 Microchip Technology Inc.

```
#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)
```


© 2018 Microchip Technology Inc.

B.3 PAC193X_INTF.H HEADER FILE

```
////////////////////////////////////////////////////////////////////////////////
//      (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
```


© 2018 Microchip Technology Inc.

```
    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
```


© 2018 Microchip Technology Inc.

```
                    // 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)
```


© 2018 Microchip Technology Inc.

```
////////////////////////////////////////////////////////////////////////////
// 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.
```


© 2018 Microchip Technology Inc.

```
// - 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)
```


© 2018 Microchip Technology Inc.

각국의 영업소 및 서비스

미주

본사
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
기술 지원:
http://www.microchip.com/support
웹사이트:
www.microchip.com

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

오스틴, TX
Tel: 512-257-3370

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

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

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

디트로이트
Novi, MI
Tel: 248-848-4000

휴스턴, TX
Tel: 281-894-5983

인디애나폴리스
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

로스앤젤레스
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

롤리, NC
Tel: 919-844-7510

뉴욕, NY
Tel: 631-435-6000

새너제이, CA
Tel: 408-735-9110
Tel: 408-436-4270

캐나다, 토론토
Tel: 905-695-1980
Fax: 905-695-2078

아시아/태평양

아태평양 지사
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

홍콩
Tel: 852-2943-5100
Fax: 852-2401-3431

호주 - 시드니
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

중국 - 베이징
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

중국 -청두
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

중국 - 충칭
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

중국 - 둥관
Tel: 86-769-8702-9880

중국 - 광저우
Tel: 86-20-8755-8029

중국 - 항저우
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

중국 - 홍콩 SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

중국 - 난징
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

중국 - 칭다오
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

중국 - 상하이
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

중국 - 선양
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

중국 - 선전
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

중국 - 우한
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

중국 - 시안
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

아시아/태평양

중국 - 샤먼
Tel: 86-592-2388138
Fax: 86-592-2388130

중국 - 주하이
Tel: 86-756-3210040
Fax: 86-756-3210049

인도 - 방갈로르
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

인도 - 뉴델리
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

인도 - 푸네
Tel: 91-20-3019-1500

일본 - 오사카
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

일본 - 도쿄
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

한국 - 대구
Tel: 82-53-744-4301
Fax: 82-53-744-4302

한국 - 서울
Tel: 82-2-554-7200
Fax: 82-2-558-5932 또는
82-2-558-5934

말레이시아 - 쿠알라룸푸르
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

말레이시아 - 페낭
Tel: 60-4-227-8870
Fax: 60-4-227-4068

필리핀 - 마닐라
Tel: 63-2-634-9065
Fax: 63-2-634-9069

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

대만 - 신주
Tel: 886-3-5778-366
Fax: 886-3-5770-955

대만 - 가오슝
Tel: 886-7-213-7830

대만 - 타이페이
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

태국 - 방콕
Tel: 66-2-694-1351
Fax: 66-2-694-1350

유럽

오스트리아 - 벨스
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

덴마크 - 코펜하겐
Tel: 45-4450-2828
Fax: 45-4485-2829

핀란드 - 에스포
Tel: 358-9-4520-820

프랑스 - 파리
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

프랑스 - 셍끌루
Tel: 33-1-30-60-70-00

독일 - 가르힝
Tel: 49-8931-9700

독일 - 하안
Tel: 49-2129-3766400

독일 - 하일브론
Tel: 49-7131-67-3636

독일 - 카를스루에
Tel: 49-721-625370

독일 - 뮌헨
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

독일 - 로젠하임
Tel: 49-8031-354-560

이스라엘 - 라아나나
Tel: 972-9-744-7705

이탈리아 - 밀라노
Tel: 39-0331-742611
Fax: 39-0331-466781

이탈리아 - 파도바
Tel: 39-049-7625286

네덜란드 - 드루넨
Tel: 31-416-690399
Fax: 31-416-690340

노르웨이 - 트론헤임
Tel: 47-7289-7561

폴란드 - 바르샤바
Tel: 48-22-3325737

루마니아 - 부쿠레슈티
Tel: 40-21-407-87-50

스페인 - 마드리드
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

스웨덴 - 예테보리
Tel: 46-31-704-60-40

스웨덴 - 스톡홀름
Tel: 46-8-5090-4654

영국 - 워킹엄
Tel: 44-118-921-5800
Fax: 44-118-921-5820

© 2018 Microchip Technology Inc.