MSP PMBus Library

API Guide

DOCNUM-1.00.00.00

Contents

Chapter 1

Copyright

Texas Instruments
13532 N. Central Expressway
Dallas, TX 75243
www.ti.com

Chapter 2

Introduction

This document describes the programming interface of the Power Management Bus (PMBus) protocol using the MSP430™ hardware as the master PMBus device.

This API document is broken out into the following sections:

- Modules
  - a list of the public API's designed to be called by user applications.
- Files
  - File List, an alphabetical list of API's avaialable in MSP430™ PMBus Library
  - Globals, an alphabetical list of MSP430™ PMBus Library global definitions

Chapter 3

Disclaimer

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

You can obtain information on other Texas Instruments products and application solutions from www.ti.com.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Module Documentation

5.1 Pmbus

Macros

- #define **PMB_PAGE** 0x00
- #define **PMB_OPERATION** 0x01
- #define **PMB_ON_OFF_CONFIG** 0x02
- #define **PMB_CLEAR_FAULTS** 0x03
- #define **PMB_PHASE** 0x04
- #define **PMB_PAGE_PLUS_WRITE** 0x05
- #define **PMB_ZONE_ACTIVE** 0x08
- #define **PMB_WRITE_PROTECT** 0x10
- #define **PMB_STORE_DEFAULT_ALL** 0x11
- #define **PMB_RESTORE_DEFAULT_ALL** 0x12
- #define **PMB_STORE_DEFAULT_CODE** 0x13
- #define **PMB_RESTORE_DEFAULT_CODE** 0x14
- #define **PMB_STORE_USER_ALL** 0x15
- #define **PMB_RESTORE_USER_ALL** 0x16
- #define **PMB_STORE_USER_CODE** 0x17
- #define **PMB_RESTORE_USER_CODE** 0x18
- #define **PMB_CAPABILITY** 0x19
- #define **PMB_VOUT_MODE** 0x20
- #define **PMB_VOUT_COMMAND** 0x21
- #define **PMB_VOUT_TRIM** 0x22
- #define **PMB_VOUT_CAL_OFFSET** 0x23
- #define **PMB_VOUT_MAX** 0x24
- #define **PMB_VOUT_MARGIN_HIGH** 0x25
- #define **PMB_VOUT_MARGIN_LOW** 0x26
- #define **PMB_VOUT_TRANSITION_RATE** 0x27
- #define **PMB_VOUT_DROOP** 0x28
- #define **PMB_VOUT_SCALE_LOOP** 0x29
- #define **PMB_VOUT_SCALE_MONITOR** 0x2A
- #define **PMB_VOUT_MIN** 0x2B
- #define **PMB_POUT_MAX** 0x31
- #define **PMB_MAX_DUTY** 0x32
- #define **PMB_FREQUENCY_SWITCH** 0x33
- #define **PMB_POWER_MODE** 0x34
- #define **PMB_VIN_ON** 0x35
- #define **PMB_VIN_OFF** 0x36

- #define **PMB_INTERLEAVE** 0x37
- #define **PMB_IOUT_CAL_GAIN** 0x38
- #define **PMB_IOUT_CAL_OFFSET** 0x39
- #define **PMB_FAN_CONFIG_1_2** 0x3A
- #define **PMB_FAN_COMMAND_1** 0x3B
- #define **PMB_FAN_COMMAND_2** 0x3C
- #define **PMB_FAN_CONFIG_3_4** 0x3D
- #define **PMB_FAN_COMMAND_3** 0x3E
- #define **PMB_FAN_COMMAND_4** 0x3F
- #define **PMB_VOUT_OV_FAULT_LIMIT** 0x40
- #define **PMB_VOUT_OV_FAULT_RESPONSE** 0x41
- #define **PMB_VOUT_OV_WARN_LIMIT** 0x42
- #define **PMB_VOUT_UV_WARN_LIMIT** 0x43
- #define **PMB_VOUT_UV_FAULT_LIMIT** 0x44
- #define **PMB_VOUT_UV_FAULT_RESPONSE** 0x45
- #define **PMB_IOUT_OC_FAULT_LIMIT** 0x46
- #define **PMB_IOUT_OC_FAULT_RESPONSE** 0x47
- #define **PMB_IOUT_OC_LV_FAULT_LIMIT** 0x48
- #define **PMB_IOUT_OC_LV_FAULT_RESPONSE** 0x49
- #define **PMB_IOUT_OC_WARN_LIMIT** 0x4A
- #define **PMB_IOUT_UC_FAULT_LIMIT** 0x4B
- #define **PMB_IOUT_UC_FAULT_RESPONSE** 0x4C
- #define **PMB_OT_FAULT_LIMIT** 0x4F
- #define **PMB_OT_FAULT_RESPONSE** 0x50
- #define **PMB_OT_WARN_LIMIT** 0x51
- #define **PMB_UT_WARN_LIMIT** 0x52
- #define **PMB_UT_FAULT_LIMIT** 0x53
- #define **PMB_UT_FAULT_RESPONSE** 0x54
- #define **PMB_VIN_OV_FAULT_LIMIT** 0x55
- #define **PMB_VIN_OV_FAULT_RESPONSE** 0x56
- #define **PMB_VIN_OV_WARN_LIMIT** 0x57
- #define **PMB_VIN_UV_WARN_LIMIT** 0x58
- #define **PMB_VIN_UV_FAULT_LIMIT** 0x59
- #define **PMB_VIN_UV_FAULT_RESPONSE** 0x5A
- #define **PMB_IIN_OC_FAULT_LIMIT** 0x5B
- #define **PMB_IIN_OC_FAULT_RESPONSE** 0x5C
- #define **PMB_IIN_OC_WARN_LIMIT** 0x5D
- #define **PMB_POWER_GOOD_ON** 0x5E
- #define **PMB_POWER_GOOD_OFF** 0x5F
- #define **PMB_TON_DELAY** 0x60
- #define **PMB_TON_RISE** 0x61
- #define **PMB_TON_MAX_FAULT_LIMIT** 0x62
- #define **PMB_TON_MAX_FAULT_RESPONSE** 0x63
- #define **PMB_TOFF_DELAY** 0x64
- #define **PMB_TOFF_FALL** 0x65
- #define **PMB_TOFF_MAX_WARN_LIMIT** 0x66
- #define **PMB_POUT_OP_FAULT_LIMIT** 0x68
- #define **PMB_POUT_OP_FAULT_RESPONSE** 0x69
- #define **PMB_POUT_OP_WARN_LIMIT** 0x6A
- #define **PMB_PIN_OP_WARN_LIMIT** 0x6B
- #define **PMB_STATUS_BYTE** 0x78
- #define **PMB_STATUS_WORD** 0x79
- #define **PMB_STATUS_VOUT** 0x7A
- #define **PMB_STATUS_IOUT** 0x7B
- #define **PMB_STATUS_INPUT** 0x7C

- #define **PMB_STATUS_TEMPERATURE** 0x7D
- #define **PMB_STATUS_CML** 0x7E
- #define **PMB_STATUS_OTHER** 0x7F
- #define **PMB_STATUS_MFR_SPECIFIC** 0x80
- #define **PMB_STATUS_FANS_1_2** 0x81
- #define **PMB_STATUS_FANS_3_4** 0x82
- #define **PMB_READ_KWH_CONFIG** 0x85
- #define **PMB_READ_EIN** 0x86
- #define **PMB_READ_EOUT** 0x87
- #define **PMB_READ_VIN** 0x88
- #define **PMB_READ_IIN** 0x89
- #define **PMB_READ_VCAP** 0x8A
- #define **PMB_READ_VOUT** 0x8B
- #define **PMB_READ_IOUT** 0x8C
- #define **PMB_READ_TEMPERATURE_1** 0x8D
- #define **PMB_READ_TEMPERATURE_2** 0x8E
- #define **PMB_READ_TEMPERATURE_3** 0x8F
- #define **PMB_READ_FAN_SPEED_1** 0x90
- #define **PMB_READ_FAN_SPEED_2** 0x91
- #define **PMB_READ_FAN_SPEED_3** 0x92
- #define **PMB_READ_FAN_SPEED_4** 0x93
- #define **PMB_READ_DUTY_CYCLE** 0x94
- #define **PMB_READ_FREQUENCY** 0x95
- #define **PMB_READ_POUT** 0x96
- #define **PMB_READ_PIN** 0x97
- #define **PMB_PMBUS_REVISION** 0x98
- #define **PMB_MFR_ID** 0x99
- #define **PMB_MFR_MODEL** 0x9A
- #define **PMB_MFR_REVISION** 0x9B
- #define **PMB_MFR_LOCATION** 0x9C
- #define **PMB_MFR_DATE** 0x9D
- #define **PMB_MFR_SERIAL** 0x9E
- #define **PMB_APP_PROFILE_SUPPORT** 0x9F
- #define **PMB_MFR_VIN_MIN** 0xA0
- #define **PMB_MFR_VIN_MAX** 0xA1
- #define **PMB_MFR_IIN_MAX** 0xA2
- #define **PMB_MFR_PIN_MAX** 0xA3
- #define **PMB_MFR_VOUT_MIN** 0xA4
- #define **PMB_MFR_VOUT_MAX** 0xA5
- #define **PMB_MFR_IOUT_MAX** 0xA6
- #define **PMB_MFR_POUT_MAX** 0xA7
- #define **PMB_MFR_TAMBIENT_MAX** 0xA8
- #define **PMB_MFR_TAMBIENT_MIN** 0xA9
- #define **PMB_MFR_EFFICIENCY_LL** 0xAA
- #define **PMB_MFR_EFFICIENCY_HL** 0xAB
- #define **PMB_MFR_PIN_ACCURACY** 0xAC
- #define **PMB_IC_DEVICE_ID** 0xAD
- #define **PMB_IC_DEVICE_REV** 0xAE
- #define **PMB_USER_DATA_00** 0xB0
- #define **PMB_USER_DATA_01** 0xB1
- #define **PMB_USER_DATA_02** 0xB2
- #define **PMB_USER_DATA_03** 0xB3
- #define **PMB_USER_DATA_04** 0xB4
- #define **PMB_USER_DATA_05** 0xB5
- #define **PMB_USER_DATA_06** 0xB6

- #define **PMB_USER_DATA_07** 0xB7
- #define **PMB_USER_DATA_08** 0xB8
- #define **PMB_USER_DATA_09** 0xB9
- #define **PMB_USER_DATA_10** 0xBA
- #define **PMB_USER_DATA_11** 0xBB
- #define **PMB_USER_DATA_12** 0xBC
- #define **PMB_USER_DATA_13** 0xBD
- #define **PMB_USER_DATA_14** 0xBE
- #define **PMB_USER_DATA_15** 0xBF
- #define **PMB_MFR_MAX_TEMP_1** 0xC0
- #define **PMB_MFR_MAX_TEMP_2** 0xC1
- #define **PMB_MFR_MAX_TEMP_3** 0xC2
- #define PMBUS_RET_OK (1)

  Return value when successful.
- #define PMBUS_RET_ERROR (-1)

  Return value when an error ocurred.
- #define PMB_MAX_PACKET_SIZE SMB_MAX_PACKET_SIZE

  Max packet size == SMBus max packet size.
- #define PMBUS_ARA (0x0C)

  Alert Response Address.

Functions

- void PMBus_init (uint16_t i2cAddr, uint32_t busClk)

  Initialize the PMBus Interface.
- void PMBus_processInt ()

  I2C Interrupt Service routine.
- int8_t PMBus_cmdRead (uint8_t slaveAddress, uint8_t commandId, uint8_t *rxData, uint8_t *rxSize, uint32_t timeout)

  Perform a PMBus read for the given PMBus command id.
- int8_t PMBus_cmdWrite (uint8_t slaveAddress, uint8_t commandId, uint8_t *txData, uint8_t txSize, uint32_t timeout)

  Perform a PMBus write for the given PMBus command id.
- int8_t PMBus_cmdSendByte (uint8_t slaveAddress, uint8_t txData, uint32_t timeout)

  Sends byte to the PMBus slave.
- int8_t PMBus_cmdWriteByte (uint8_t slaveAddress, uint8_t commandId, uint8_t *txData, uint32_t timeout)

  Writes a data byte to the PMBus slave.
- int8_t PMBus_cmdWriteWord (uint8_t slaveAddress, uint8_t commandId, uint8_t *txData, uint32_t timeout)

  Writes a word to the PMBus slave.
- int8_t PMBus_cmdBlockWrite (uint8_t slaveAddress, uint8_t commandId, uint8_t *txData, uint8_t txSize, uint32_t timeout)

  Writes a data block the PMBus slave.
- int8_t PMBus_cmdReceiveByte (uint8_t slaveAddress, uint8_t *rxData, uint32_t timeout)

  Receives a data byte from the PMBus slave.
- int8_t PMBus_cmdReadByte (uint8_t slaveAddress, uint8_t commandId, uint8_t *rxData, uint32_t timeout)

  Reads a data byte from the PMBus slave.
- int8_t PMBus_cmdReadWord (uint8_t slaveAddress, uint8_t commandId, uint8_t *rxData, uint32_t timeout)

  Reads a data word from the PMBus slave.
- int8_t PMBus_cmdBlockRead (uint8_t slaveAddress, uint8_t commandId, uint8_t *rxData, uint8_t *rxSize, uint32_t timeout)

  Reads a block of data from the PMBus slave.

- int8_t PMBus_cmdProcessCall (uint8_t slaveAddress, uint8_t commandByte, uint8_t *txData, uint8_t *rxData, uint32_t timeout)

  Sends a Process call command to the PMBus slave.
- int8_t PMBus_cmdBlockWriteBlockReadProcessCall (uint8_t slaveAddress, uint8_t commandId, uint8_t *txData, uint8_t txSize, uint8_t *rxData, uint8_t *rxSize, uint32_t timeout)

  Sends a Block-Write, Block-Read, Process Call to the PMBus slave.
- int8_t PMBus_ReceiveByteARA (uint8_t *rxData, uint32_t timeout)

  Sends a Receive Byte to Alert Response Address to request.

5.1.1 Detailed Description

5.1.2 Function Documentation

5.1.2.1 int8_t PMBus_cmdBlockRead (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *rxData*, uint8_t * *rxSize*, uint32_t *timeout*)

Reads a block of data from the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
rxData	RX data buffer
rxSize	Number of bytes received
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.2 int8_t PMBus_cmdBlockWrite (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *txData*, uint8_t *txSize*, uint32_t *timeout*)

Writes a data block the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
txData	TX data buffer
txSize	Number of bytes to transfer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.3 int8_t PMBus_cmdBlockWriteBlockReadProcessCall (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *txData*, uint8_t *txSize*, uint8_t * *rxData*, uint8_t * *rxSize*, uint32_t *timeout*)

Sends a Block-Write, Block-Read, Process Call to the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
txData	TX data buffer
txSize	Number of bytes to transfer
rxData	RX data buffer
rxSize	Number of bytes received
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.4 int8_t PMBus_cmdProcessCall (uint8_t *slaveAddress*, uint8_t *commandByte*, uint8_t * *txData*, uint8_t * *rxData*, uint32_t *timeout*)

Sends a Process call command to the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
txData	TX data buffer
rxData	RX data buffer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.5 int8_t PMBus_cmdRead (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *rxData*, uint8_t * *rxSize*, uint32_t *timeout*)

Perform a PMBus read for the given PMBus command id.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
rxData	RX data buffer
rxSize	Number of bytes received
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.6 int8_t PMBus_cmdReadByte (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *rxData*, uint32_t *timeout*)

Reads a data byte from the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
rxData	RX data buffer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.7 int8_t PMBus_cmdReadWord (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *rxData*, uint32_t *timeout*)

Reads a data word from the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
rxData	RX data buffer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.8 int8_t PMBus_cmdReceiveByte (uint8_t *slaveAddress*, uint8_t * *rxData*, uint32_t *timeout*)

Receives a data byte from the PMBus slave.

Parameters

slaveAddress	Slave address
rxData	RX data buffer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.9 int8_t PMBus_cmdSendByte (uint8_t *slaveAddress*, uint8_t *txData*, uint32_t *timeout*)

Sends byte to the PMBus slave.

Parameters

slaveAddress	Slave address
txData	TX data buffer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.10 int8_t PMBus_cmdWrite (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *txData*, uint8_t *txSize*, uint32_t *timeout*)

Perform a PMBus write for the given PMBus command id.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
txData	TX data buffer
txSize	Number of bytes to be transferred
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.11 int8_t PMBus_cmdWriteByte (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *txData*, uint32_t *timeout*)

Writes a data byte to the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
txData	TX data buffer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.12 int8_t PMBus_cmdWriteWord (uint8_t *slaveAddress*, uint8_t *commandId*, uint8_t * *txData*, uint32_t *timeout*)

Writes a word to the PMBus slave.

Parameters

slaveAddress	Slave address
commandId	One of the PMBus command id's #defined in pmbus.h
txData	TX data buffer
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**

5.1.2.13 void PMBus_init (uint16_t *i2cAddr*, uint32_t *busClk*)

Initialize the PMBus Interface.

Parameters

i2cAddr	Base address of I2C module. For MSP430G2xxx devices, this parameter is ignored.
busClk	SMCLK Frequency

5.1.2.14 void PMBus_processInt ( )

I2C Interrupt Service routine.

Handles the interrupts for PMBus. Should be called by application when USCI interrupt is detected

Returns

None

5.1.2.15 int8_t PMBus_ReceiveByteARA (uint8_t * *rxData,* uint32_t *timeout*)

Sends a Receive Byte to Alert Response Address to request.

Parameters

rxData	RX buffer returning address of device invoking the alert
timeout	Software timeout waiting for a response

Returns

PMBUS_RET_ERROR, or **PMBUS_RET_OK**