I²C/SMBus Controller

Features

- I²C/SMBus 2.0 Host Controller
- I²C Backward-compatibility Mode
- Collision Detect Hardware (Multi-Master Support)
- Slave Support:
- Support for multiple Slave addresses:
  - Programmable Slave Address Register for two generic slave addresses
  - SMBus Host Address (0001 000) for Host Notify Commands
  - SMBus Device Default Address (1100 001)
  - General Call Address (0000 000)
- Packet Error Checking (PEC) Support Option
- Selectable I²C/SMBus transaction speeds up to 1MHz
- ARP Support
- Support for 7-bit SMBus Addressing
- Host Notify command (received message) Support
- Input Noise Filter
- Support for Master/Slave Clock Stretching
- Support for SMBus Timeouts
- Bit-Bang Mode
- Support I²C and SMBus Timing.
- Hardware support for MCTP Fairness Protocol (see Ref [5])
- Up to 16 Selectable Bus Ports
- Hardware support for network layer protocols
  - Support for virtually all standard SMBus transactions
  - Support for other protocols that use I²C (e.g., PMBus; see Ref [6])
  - Combined with an external DMA, support for hardware transfer of up to 255 bytes of data between memory and the physical layer
  - No polling and no more than one or two interrupts required per network layer transaction

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include -literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com** to receive the most current information on all of our products.

Table of Contents

1.0 CONVENTIONS

The first table defines common terminology used in the documentation. The second table defines the register bit access type notation used in the documentation. These are the access types that are supported by CSL.

Term	Definition
Block	Used to identify or describe the logic or IP Blocks implemented in the device.
Reserved	Reserved registers and bits defined in the following table are read only values that return 0 when read. Writes to these reserved registers have no effect.
Microchip Reserved	Microchip Reserved locations should not be modified from their default value. Changing an Microchip register may cause unwanted results.
b	The letter ‘b’ following a number denotes a binary number.
h	The letter ‘h’ following a number denotes a hexadecimal number.

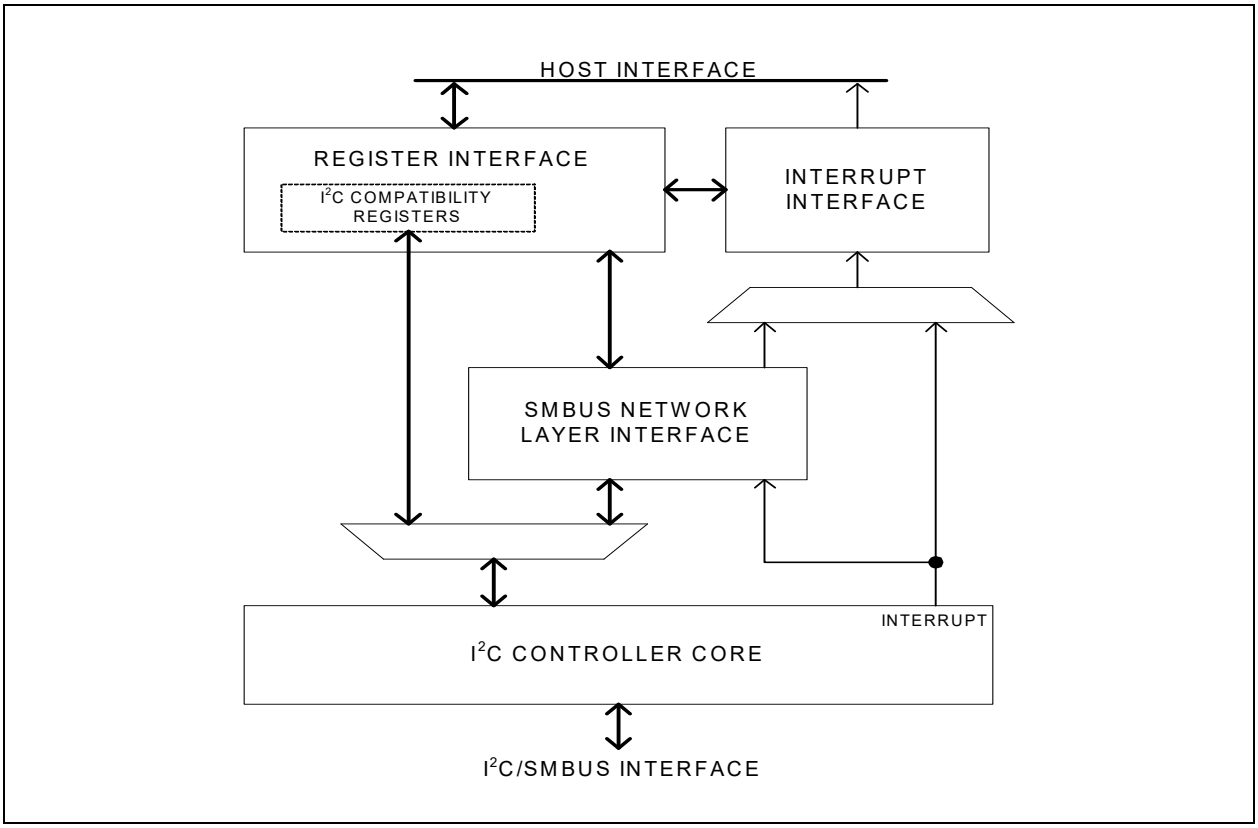
Register Bit Type Notation	Register Bit Description
R	**Read:** A register or bit with this attribute can be read only. Write to this bit has not effect.
R/W	**Write:** A register or bit with this attribute can be written and read back.
R/WC	**Write:** A register or bit with this attribute can be written and read back. Writing a ‘1’ clears the value. Writing a ‘0’ has no effect.

2.0 OVERVIEW

The I²C Bus protocol and the SMBus protocol are both used in many aspects of system internal communication. The I²C/SMBus Controller interface can handle standard SMBus 2.0 protocols as well as I²C interfaces. The I²C/SMB Controller is implemented on two levels: a low-level I²C core and a higher level network layer that uses the I²C core for lower-level data link and physical layer operations.

A block diagram of the I²C/SMBus Controller is shown in Figure 2-1.

FIGURE 2-1: I²C/SMB CONTROLLER BLOCK DIAGRAM



The I²C/SMB Controller can also be configured to execute in hardware the physical, data-link and network layer functions of the SMBus specification as described in Ref [1]. Network layer SMBus protocols (e.g, read byte, write byte, read process call, etc.) are not handled completely, and in most instances must be handled partially in low-level software. Full hardware implementations of network level protocols are inherently limiting, since new variants on SMBus network layer protocols continue to appear (for example, PMBus specified a multi-frame write protocol that is not part of the standard SMBus 2.0 specification).

The I²C/SMB Controller accelerates data movement in hardware but relies on software for the details of network layer protocol. Working in conjunction with an external DMA controller, an embedded controller can handle all current network layer protocols and most if not all future ones with minimal software intervention.

2.1 Reference Documents

1. System Management Bus (SMBus) Specification, Version 2.0, SBS Implementors Forum, August 3, 2000 (http://www.sbs-forum.org/).
2. The I²C-Bus Specification, Version 2.1, Philips Semiconductors, January 2000.
3. PCF8584 I²C-Bus Controller, Product specification, Philips Semiconductors, 1997 Oct 21.

4. Management Component Transport Protocol (MCTP) Base Specification, Version 1.0, DMTF DSP0236, June 20, 2007.
5. Management Component Transport Protocol SMBus/I²C Transport Binding Specification, Version 1.0, DMTF DSP0237, June 20, 2007.
6. PMBus™ Power System Management Protocol, Revision 1.1, 5 February 2007, System Management interface Forum.

3.0 HARDWARE INTERFACE

The hardware interface to the I²C/SMB Controller includes the 2-wire SDA and SCL bus pair. This interface also features Bus Port Selection, which allows assigning the two-wire bus pair to any one of 16 possible bus ports.

The I²C/SMB bus is manipulated directly by the controller. In addition, software can control the two bus wires directly through the Bit-Bang interface, described in Section 7.1.15, "Bit-Bang Control Register".

3.1 SCL

SCL represents the serial clock line in the I²C interface as defined in Ref [2] and referenced throughout this specification. As defined in Ref [2], SCL is a bi-directional signal connected to a positive supply voltage via an external current source or pull-up resistor.

3.2 SDA

SDAT represents the serial data line in the I²C interface as defined in Ref [2] and referenced throughout this specification. As defined in Ref [2], SDAT is a bi-directional signal connected to a positive supply voltage via an external current source or pull-up resistor.

3.3 Bus Port Selection

The I²C/SMB Controller can assign the 2-wire SDAT and SCLK bus pairs to any one of 16 possible bus ports using the PORT_SEL bits in the Configuration Register.

4.0 POWER, CLOCKS AND RESETS

This section defines the Power, Clock, and Reset parameters of the block.

4.1 Power Domains

Name	Description
EC_PWR	The I²C/SMB requires one supply voltage.

4.2 Clock Inputs

Name	Description
CORE_CLOCK	This is the main system clock for the part which incorporates the I²C/SMB
I2C_BAUD_CLOCK	The Baud Clock provides clocking for the I²C/SMB Controller transaction state machines and the physical I²C bus interface. Timing choices based on the current Baud Clock are shown in Table 7-2, "Recommended Programming Values" and Table 7-3, "Bus Clock Register vs. Frequency". The Baud Clock may be one of the following frequencies: • 10MHz • 16MHz • 20MHz
I2C_BUS_CLOCK	This is the clock on the external bus, carried on the SCL pin. When the I²C/SMB is acting as a slave, the external master controls the frequency of this clock. When the I²C/SMB is acting as a master, the frequency of this clock is defined by the Bus Clock Register.

4.3 Resets

Name	Description
RESET_EC	This is the reset signal used to reset the majority of the logic in a device.
RESET_I2C	This signal resets all the registers and logic in this block to their default state. I2C_RESET is asserted when: • MAIN_RESET is asserted • The I²C/SMB detects an error condition which requires resetting the block • The RESET bit in the Configuration Register is asserted

5.0 INTERRUPT INTERFACE

This section defines the Interrupt Sources generated from this block.

TABLE 5-1: EC INTERRUPTS

Source	Description
I2C_INT	Interrupt request to the Interrupt Aggregator. This interrupt is asserted any time any one of four status bits is asserted when its associated enable bit is high: • PIN (asserted low), enabled by ENI • IDLE, enabled by ENIDI • MDONE, enabled by ENMI • SDONE, enabled by ENSI
I2C_INT_WAKE	Wake-up request to the Interrupt Aggregator's wake-up interface. This interrupt is asserted when the START_BIT_DETECTION status bit is '1' when the START_DETECT_INT_EN bit is also '1'. See the following section.

5.1 I²C WAKE

As long as it is enabled, the I²C/SMB samples the SCL and SDA inputs asynchronously, even when its SLEEP_EN input is asserted. A wake event is generated if a START bit is detected on the bus and the I2C_INT_WAKE event is enabled. The wake capability permits the chip to enter low power modes without missing an I²C/SMBus command. The clock signal (SCL) will be stretched low immediately following the START bit until the CORE_CLOCK has attained its normal operating frequency. The clock stretching time is dependent on the sleep recovery time, which can vary depending on chip implementation and the state of the chip when the START bit is detected.

6.0 FUNCTIONAL DESCRIPTION

The I²C/SMB Controller contains a Core interface, described in Section 6.1, "I²C Controller Core", and a network layer interface that uses the Core interface to access the physical bus, described in Section 6.2, "SMBus Network Layer Interface".

6.1 I²C Controller Core

The I²C/SMB implements physical and data link layer support as defined in Ref [2] and Ref [3]. The core set of registers provide a byte-at-a-time interface to the external bus.

6.1.1 PIN OPERATION

The Pending Interrupt Not bit (PIN) is a status flag used to synchronize serial communication and is asserted ('0') whenever the I²C/SMB Controller requires servicing. The PIN bit is normally read in polled applications to determine when a transaction is complete (see Section 6.1.6, "Transaction Primitives").

When acting as a transmitter, PIN is de-asserted ('1') each time the Data Register is written. When receiving, the PIN bit is de-asserted each time the Data Register is read.

After transmission or reception of one byte on the bus (nine clock pulses, including acknowledge), the PIN bit will be automatically asserted ('0') (active) indicating a complete byte transmission/reception. When the PIN bit is subsequently set to logic '1' (inactive), all status bits in the Status Register will be reset to "0;" including, the BER bit, even if Bus Time-Outs have occurred. PIN is also asserted after the byte acknowledge (ACK) following Lost Arbitration. PIN is immediately asserted on a bus error (BER) condition.

When the ENI bit in the Control Register is asserted ('1'), the hardware PIN interrupt is enabled. In this case, the PIN bit triggers an external interrupt each time PIN is asserted ('0') (see also Section 5.0, "Interrupt Interface").

6.1.1.1 Clock Stretching

When acting as a slave transmitter or slave receiver, while PIN is asserted ('0'), the I²C/SMB Controller will suspend bus transmission by holding the SCL signal low until the PIN bit is de-asserted. Clock stretching prevents further data from being transmitted or received until the current byte in the Data Register has been read (when acting as slave receiver) or the next byte is written to the Data Register (when acting as slave transmitter).

6.1.2 SLAVE ADDRESS DECODING

When configured as a slave, the I²C/SMB can respond to up to five slave addresses:

TABLE 6-1: SLAVE ADDRESSES

Slave Address	Configuration Register		Status Register		
	GC_DIS	DSA	AAS	LRB/AD0	SAD
General Call Address (0000_000b)	0	x	1	1	0
Own Address 1	x	x	1	0	0
Own Address 2	x	x	1	0	0
SMBus Host Address (0001_000b)	x	1	1	0	1
SMBus Device Default Address (1100_001b)	x	1	1	1	1

6.1.2.1 Alert Response Address Decoding

A slave-only device can signal the host through the SMBALERT# sideband signal (beyond the scope of this specification) that service is required. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (ARA) (0001 100). Alert functionality using the ARA is optional and, as described above, is for slave-only devices. This specification assumes that the I²C/SMB Controller does not need to decode this address in hardware to support alert functionality because it can be configured as a master. If the I²C/SMB Controller is configured as a slave and asserts SMBALERT# (e.g., using a GPIO output dedicated for this purpose), software has the option to use one of the OWN addresses to decode the ARA.

6.1.3 START BYTE SUPPORT

The I²C/SMB Controller is required to provide START Byte Support as defined by the I²C Bus Specification (Ref [2]) and summarized in Table 6-2, "I2C/SMB Controller Addressing Behavior Summary". START Byte Support includes a unique slave response to the START Byte Address as described in Section 6.1.3.1, "START Byte Address".

TABLE 6-2: I²C/SMB CONTROLLER ADDRESSING BEHAVIOR SUMMARY

M/S (Note 1)	ADDR	R/W Bit (Note 2)	ACK/NACK	Description (Note 3)
M	ANY	0	ACK	Following slave address master sends DATA, STOP or SR.
			NACK	Following slave address master sends STOP or SR.
		1	ACK	Following slave address master receives DATA, sends STOP or SR.
		1	NACK	Following slave address master sends STOP or SR.
S	0000 000b	1	ACK	ACK is an illegal slave response to START Byte Address as defined in Ref [2]
			NACK	Correct slave response to START Byte Address
	>0000 000b	1	ACK	Following slave address slave transmits DATA to master.
			NACK	Slave unavailable.
	ANY	0	ACK	Following slave address slave receives DATA from master.
			NACK	Slave unavailable.

Note 1: M = Master (transmitting ADDR and R/W Bit), S = Slave (transmitting ACK/NACK as response to slave address).

2: 0 = Write (Master Transmitter), 1 = Read (Master Receiver).

3: DATA = next data byte, STOP = STOP Condition, SR = Repeated Start Condition.

6.1.3.1 START Byte Address

A START Byte Address is defined in Table 2 of Ref [2] as slave address 0000 000b with the R/nW bit set ('1'). The I²C/SMB Controller slave interface ignores a START Byte Address (i.e., NACK the address, discard the data and do not assert the PIN bit) to prepare for a Repeated Start Condition.

The I²C/SMB Controller when configured as a master depends on software to judge the slave response to the START Byte Address.

6.1.4 HARDWARE PEC SUPPORT

The I²C/SMB Controller includes hardware PEC support to help reduce software overhead for SMBus 2.0 packet error checking. User interface elements for PEC support include an 8-bit PEC Register and a PEC Enable Bit.

The PEC hardware calculates the PEC byte from the transmit and receive serial data stream as defined by the SMBus 2.0 specification (Ref [1]) when the PCEN, the PEC Enable bit, is asserted. Valid PEC data can be read by software from the PEC register whenever the PIN bit in the Status Register is asserted. Software can initialize the PEC byte to any arbitrary value by writing to the PEC Register when the PCEN is not asserted. PEC is cleared whenever the Master State Machine issues a STOP bit or when the Slave State Machine completes processing. PEC is also cleared if a Lost Arbitration condition occurs in the Master and the Slave State Machine is idle (that is, PEC is reset if the Master loses arbitration to an external master, but not if the external master was targeting the Slave)

When the Master State Machine is operating, the PEC register is copied to the Master Receive Buffer Register when the state machine completes a receive operation if READ_PEC is 1. When READ_PEC is 1, the Master State Machine copies the final PEC byte from the Slave into the Master memory receive buffer and then copies the PEC register as well. Software can verify that the incoming data stream was received correctly by checking to see that the last byte in the memory buffer, the copy of the PEC register, is 0.

When the Slave State Machine is operating, the PEC register is copied to the Slave Receive Buffer Register after the external Master has issued a STOP or repeated START while the state machine is receiving data. The Slave State Machine therefore always copies one more byte (the contents of the PEC register) into the Slave memory receive buffer than is received over the I²C bus. Software can determine what protocol was in use over the SMBus and therefore determine whether the last byte received over the I²C bus was a PEC byte. If the last received byte was a PEC byte, then the last byte in the memory buffer (the copy of the PEC register) will be 0 if the received data verified correctly.

When the Master and Slave state machines are disabled, software responsibilities for PEC support include initializing the PEC byte using the PEC register, enabling PEC where appropriate and inserting the PEC register value in, or comparing this value to, the Data Register when needed.

6.1.4.1 PEC Example

The following table shows an example PEC calculation for a thirty-two byte message where the message data for each byte equals the byte number in hex.

TABLE 6-3: CRC-8 EXAMPLE

Byte #	Message (Hex)	CRC (Hex)
	0	0
1	1	7
2	2	1B
3	3	48
4	4	E3
5	5	BC
6	6	2F
7	7	D8
8	8	3E
9	9	85
10	A	A4
11	B	44
12	C	FF
13	D	D0
14	E	14
15	F	41
16	10	B0
17	11	6E
18	12	73
19	13	27
20	14	99
21	15	AD
22	16	28
23	17	BD
24	18	72
25	19	16
26	1A	24
27	1B	BD
28	1C	6E
29	1D	5E
30	1E	C7

TABLE 6-3: CRC-8 EXAMPLE (CONTINUED)

Byte #	Message (Hex)	CRC (Hex)
31	1F	6
32	20	F2
	F2	0

6.1.5 INITIALIZATION

Following a reset, software is required to implement the initialization procedure illustrated in Table 6-4 to properly configure the I²C/SMB to successfully execute I²C compatible transactions. This sequence should be followed if the reset occurred because of an external reset signal or because the RESET bit in the Configuration Register was asserted. The procedure applies to both the I²C core layer and the network layer.

TABLE 6-4: INITIALIZATION SEQUENCE FOLLOWING RESET

	Operation	Description
1.	Write 80h to the Control Register.	Asserts PIN bit; turns off serial interface (ESO = '0') and disables Interrupts (ENI).
2.	Write a valid slave address to the Own Address Register.	Initialize Slave Address
3.	Write XXh to the Bus Clock Register.	Configures desired bus clocking. Suggested values are in shown in Table 7-3, "Bus Clock Register vs. Frequency".
4.	Write C1h to the Control Register.	PIN remains asserted; turns on serial interface (ESO = '1'); configures positive acknowledge (ACK = '1'; SDA and SCL are HIGH. (Note: the NBB is de-asserted '1' as a result of reset).
5.	Update other registers.	All other registers (Configuration Register, Idle Scaling Register, etc) should be set to the appropriate values
6.	Delay. (See Note 1)	Wait a time equal to the longest I²C message to synchronize the NBB bit in multi-master systems only (see Section 6.1.7, "I²C Multi-Master Support").
Note 1:	In order to synchronize properly, software must wait for a period equal to the longest interval another master might occupy the bus. If this period is unknown, or an external device violates the specified period by occupying the bus for a longer period, then the I²C cannot be initialized properly.	

6.1.6 TRANSACTION PRIMITIVES

All I²C and SMBus compatible protocols can be implemented using the Transaction Primitives of the I²C/SMB. These primitives include the Master Transmitter, Master Receiver, Master Transmitter Followed by Repeated START and Becoming Master Receiver and Slave Receiver/Transmitter transaction descriptions.

The transaction descriptions that follow do not include steps for detecting bus contention, as described in Section 6.1.7, "I²C Multi-Master Support", that must be incorporated into all software that is intended for use in multi-master environments. The description for initialization can be found in Section 6.1.5, "Initialization".

In all cases, the primitives described below assume that the ENAB bit in the Configuration Register is asserted and the RESET bit in the same register is not asserted.

6.1.6.1 Master Completion

In order to re-enable the Slave receiver after the Master issues a STOP command, the steps illustrated in Table 6-5 are required:

TABLE 6-5: RE-ENABLING THE SLAVE RECEIVER

	Operation	Description
1.	Read the Status Register.	Poll the NBB bit until de-asserted (‘1’); i.e., wait until the bus is idle
2.	Write C1h to the Control Register.	PIN remains asserted; turns on serial interface (ESO = ‘1’); configures positive acknowledge (ACK = ‘1’; SDA and SCL are HIGH.

6.1.6.2 Master Transmitter

It is assumed that the I²C/SMB Controller has been initialized as described in Section 6.1.5, "Initialization" and the ESO bit is asserted (‘1’).

TABLE 6-6: MASTER TRANSMITTER MODE

	Operation	Description
1.	Read the Status Register.	Poll the NBB bit until not-asserted (‘1’).
2.	Write the slave address w/D0 (R/nW bit) = ‘0’ to the Data Register.	Setup Master Transmitter mode.
3.	Write C5h to the Control Register.	PIN = ESO = ‘1,’ STA = ‘1,’ STO = ‘0,’ ACK = ‘1.’ The I²C/SMB Controller generates the 'START' condition and clocks out the slave address and the clock pulse for slave acknowledgment. The next byte(s) written to the Data Register will be immediately transferred over the I²C bus; i.e., the I²C/SMB Controller remains in Master Transmitter mode if the R/nW bit of the slave address = ‘0.’
4.	Read the Status Register.	Poll the PIN bit until asserted (‘0’); i.e., byte transmission done. If LRB/AD0 = ‘0’ (i.e., slave ACK received) and if more bytes need to be sent, go to Step 5; otherwise, go to Step 6 if no more bytes to send or slave NACKs (negative acknowledge) byte.
5.	Write next byte to Data Register.	Send next byte. Go to Step 4.
6.	Write C3h to the Control Register.	PIN = ESO = ‘1,’ STA = ‘0,’ STO = ‘1,’ ACK = ‘1.’ The I²C/SMB Controller generates the 'STOP' condition and goes into Slave Receiver/Transmitter mode. If a Repeated ‘START’ condition is required instead of ‘STOP,’ do not execute this step and go to Table 6-8, "Repeated Start" in Section 6.1.6.4, "Master Transmitter Followed by Repeated START and Becoming Master Receiver," on page 15.

6.1.6.3 Master Receiver

It is assumed that the I²C/SMB Controller has been initialized as described in Section 6.1.5, "Initialization" and the ESO bit is asserted (‘1’).

TABLE 6-7: MASTER RECEIVER MODE

	Operation	Description
1.	Write the slave address w/D0 (R/nW bit) = ‘1’ to the Data Register.	Setup Master Receiver mode.
2.	Read the Status Register.	Poll the NBB bit until not-asserted (‘1’).
3.	Write C5h to the Control Register.	PIN = ESO = ‘1,’ STA = ‘1,’ STO = ‘0,’ ACK = ‘1.’ The I²C/SMB Controller generates the 'START' condition and clocks out the slave address and the clock pulse for slave acknowledgment.
4.	Read the Status Register.	Poll the PIN bit until asserted (‘0’); i.e., byte reception done. If LRB/AD0 = ‘0’ (i.e., slave ACK received) and if more than two bytes need to be received, go to Step 5; otherwise, go to Step 6 if last byte is to be received, or go to Step 9 if slave NACKs (negative acknowledge) address.
5.	Read the Data Register.	Retrieve received data byte. Go to step 4. Note that the first byte read is a “dummy read” of the slave address. Subsequent reads are received data from the slave.
6.	Write 40h to the Control Register.	ESO = ‘1.’ De-assert ACK bit (‘0’) in preparation for negative acknowledgment to end the transfer.
7.	Read the Data Register.	Retrieve the received next-to-last byte. Receive and NACK (negative acknowledge) the final byte.
8.	Read the Status Register.	Poll the PIN bit until asserted (‘0’); i.e., final byte reception done.
9.	Write C3h to the Control Register.	PIN = ESO = ‘1,’ STA = ‘0,’ STO = ‘1,’ ACK = ‘1.’ The I²C/SMB Controller generates the 'STOP' condition and goes into Slave Receiver/Transmitter mode.
10.	Read the Data Register.	Retrieve the final received byte (or NACK’d slave address.

6.1.6.4 Master Transmitter Followed by Repeated START and Becoming Master Receiver

TABLE 6-8: REPEATED START

	Operation	Description
1.	Execute Master Transmitter mode, excluding final ‘STOP.’	See Section 6.1.6.2, "Master Transmitter". Do not execute step 6 in Table 6-6.
2.	Write 45h to the Control Register.	PIN = ‘0’, ESO = ‘1,’ STA = ‘1,’ STO = ‘0,’ ACK = ‘1.’ The I²C/SMB Controller generates the repeated 'START condition,' only; i.e., the current contents of the Data Register is NOT clocked out onto the I²C bus. The next byte sent to the Data Register should be the 'slave address' + ‘read’ bit (R/nW = ‘1’).
3.	Write the slave address w/D0 (R/nW bit) = ‘1’ to the Data Register.	Setup Master Receiver mode.
4.	Execute Master Receiver mode, excluding ‘START.’ generation.	See Section 6.1.6.3, "Master Receiver". Begin execution from step 4 in Table 6-7

6.1.6.5 Slave Receiver/Transmitter

It is assumed that the I²C/SMB Controller has been initialized as described in Section 6.1.5, "Initialization" and the ESO bit is asserted ('1').

TABLE 6-9: SLAVE RECEIVER/TRANSMITTER

	Operation	Description
1.	Read the Status Register.	Poll the Addressed As Slave (AAS) bit until asserted ('1').
2.	Read the Status Register.	Poll the PIN bit until asserted ('0'); i.e., address reception done.
3.	Read the Data Register.	Retrieve the received address byte to determine if the R/nW bit (D0) is '0' or '1' to differentiate between slave receiver (R/nW = '0,' go to step 1) or slave transmitter mode (R/nW = '1,' go to step 1). Note that the PIN bit will not be reset ('1') by reading the Data Register if R/nW = '1' because the slave is a transmitter.
SLAVE TRANSMITTER MODE		
1.	Read the Status Register.	Poll the PIN bit until asserted ('0') for slave transmitted data and master ACK. Note that the first time through this loop the PIN bit will remain asserted ('0') because the slave is a transmitter. If LRB/AD0 = '1' (master negative ACK), go to step 3.
2.	Write the Data Register.	Transmit a data byte on the bus. Go to step 1.
3.	Write the Data Register.	Perform a 'dummy write' to de-assert PIN bit ('1'). Note that the dummy write data is not sent on bus following a NACK and the clock is not held low. The I²C/SMB Controller becomes a slave receiver. **END SLAVE TRANSMITTER MODE.**
SLAVE RECEIVER MODE		
1.	Read the Status Register.	Poll the PIN bit until asserted ('0') for master transmitted data. If 'STOP' detected (STS = '1'), go to step 4. If this is not the next to last byte that the slave will receive, go to step 3.
2.	Write the Control Register.	Change ACK bit in the Control Register to '0' to NACK the last byte.
3.	Read the Data Register.	Retrieve the received data byte. Go to step 1.
4.	Read the Data Register.	Do a dummy read of the Data Register or write the PIN bit in the Control Register to de-assert the PIN bit in the Status Register. **END SLAVE RECEIVER MODE.**

6.1.7 I²C MULTI-MASTER SUPPORT

The I²C/SMB Controller includes Multi-Master support. Arbitration relies on the wired-AND connection of all I²C interfaces to this bus. If two or more masters attempt to put information onto the bus at the same time, the first to produce a 'one' when the other produces a 'zero' loses the arbitration.

Multi-Master support is automatically handled by the network layer hardware. For operations using the I²C core layer primitives only, Multi-Master support includes hardware as defined in Section 6.1.7.1, "Hardware Support" and heavily depends on software as described in the sections Accessing the Bus, Detecting Bus Contention and Lost Arbitration shown below.

6.1.7.1 Hardware Support

Hardware for I²C Multi-Master Support includes the LAB and NBB bits in the Status Register. The NBB bit is used by the master for Accessing the Bus. The LAB bit is used for Detecting Bus Contention. Definitions for both of these bits can be found in Section 7.1.2, "Status Register".

6.1.7.2 Accessing the Bus

Typically, an I²C master in a multi-master environment must not attempt to access the bus until the NBB bit in the Status Register is not asserted ('1'). The only exception to this is first enabled, where the possibility exists that the I²C/SMB Controller may power up slightly after another external I²C master has already begun a transaction and not detect the bus busy condition. To avoid bus contention in this case, a delay should be introduced in the initialization sequence of each master equal to the longest I²C-bus transaction time.

it is the responsibility of the software to identify bus irregularities like an unsynchronized NBB bit or stuck-at bus faults (e.g., using the Bit-Bang Control Register) that may obstruct or inhibit message transactions.

6.1.7.3 Bus Access Boundary Condition

There is a Bus Access Boundary Condition when the internal master verifies that NBB is not asserted and attempts to begin a transaction. Note that in this scenario the bus is idle and there are no bus synchronization issues; e.g., from a system reset or wake. During this attempt (i.e., immediately before the transaction is initiated) a transaction begins as a result of an external master.

The typical system-level response in this case requires that standard bus arbitration resolves the conflict and the losing master backs off gracefully, or a misplaced START condition is detected and a bus error (BER) is asserted which invalidates the transaction for both masters.

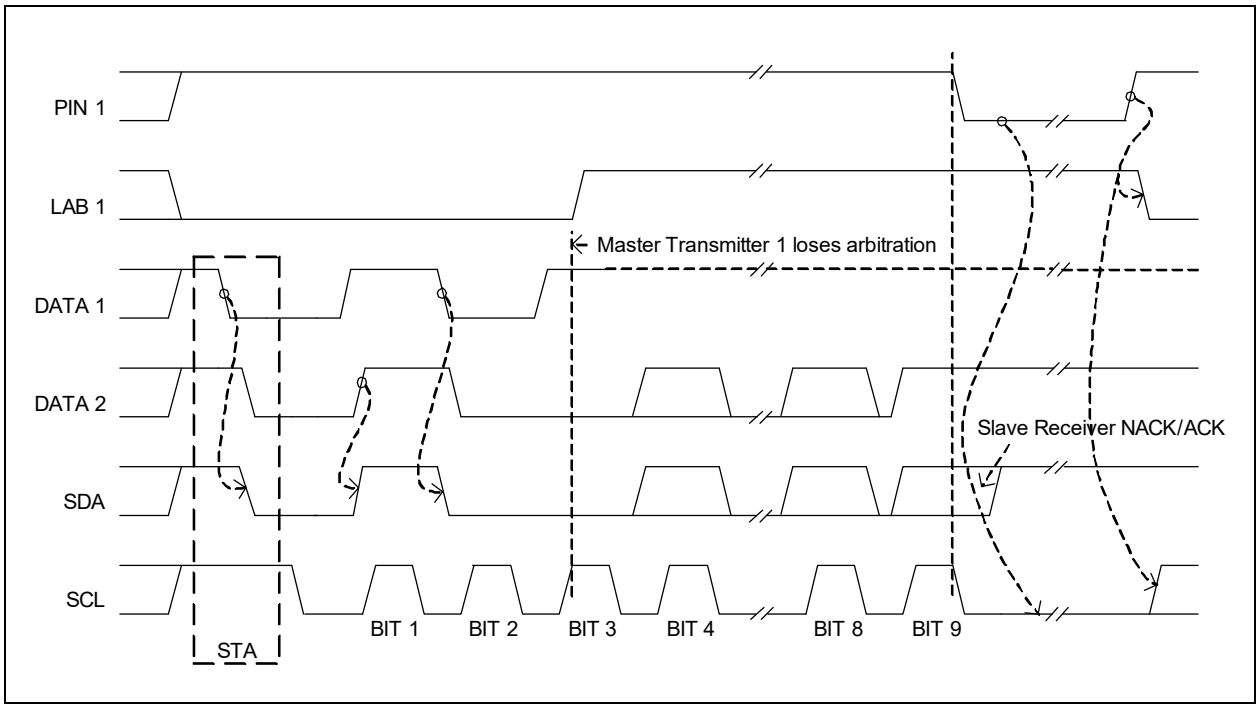
In the SMBus Controller Core if the internal master asserts the STA bit in the Control Register while NBB is asserted and the master controller is not already involved in the transaction (i.e., it is not a repeated START condition), the LAB bit will be asserted and the internal master bus drivers will be deactivated so as to not affect the transaction in progress; i.e., a bus error should not occur because message corruption is averted.

6.1.7.4 Detecting Bus Contention

Once a message transaction has begun, the only way for a master to know that another master has not also simultaneously begun a transaction is by using the LAB bit in the Status Register. Arbitration takes place on the SDA line in such a way that the master which transmits a HIGH level while another master is transmitting a LOW level will switch off its data output stage because the level on the bus doesn't correspond to its own level (see Figure 6-1, below). The I²C master driver is responsible for detecting Lost Arbitration and terminating the transaction before subsequent data is transferred.

Note that in the I²C/SMB Controller, bus contention detection occurs in hardware as a master and a slave during both the data and acknowledge phases of the data transfer protocol, as well as during START and repeated START conditions. Bus contention detection is not checked during a STOP.

FIGURE 6-1: DETECTING CONTENTION BETWEEN TWO MASTERS



6.1.7.5 Lost Arbitration

If Lost Arbitration occurs when detecting bus contention, the I²C controller becomes a slave receiver and holds the clock line (SCL) low indefinitely. At this point if the Master has lost arbitration to its own slave address; reading the data register for a master transmit command or writing the data register for a master receive command will clear the PIN bit.

Actions following arbitration loss are the responsibility of software. For example, if arbitration is lost before data is transferred (e.g., because of a slave address conflict), the master that lost the arbitration will typically retry the transaction at a later timer. If two masters simultaneously read data from the same slave and Master 1 requires less data than Master 2, Master 1 will lose arbitration during its last byte acknowledge (ACK) and can abandon the transaction (i.e., not generate a STOP condition) having received all its required data, while Master 2 successfully continues the transaction to completion.

If the internal master losses arbitration during START condition and the internal slave gets an address match, both the LAB and AAS bits will be asserted by the time PIN is asserted. This condition indicates that the internal master transaction should be abandoned because of arbitration loss to an external master that is addressing the internal slave.

For more detail about arbitration and clock synchronization see Ref [1].

6.1.8 INPUT FILTERING

The I²C/SMB includes an analog filter on both the SCLK and SDAT input pins. The filter is designed to reject pulses less than 50ns and always pass pulses greater than 140ns.

By default the filter is disabled. To enable the filter, set the FEN bit located in the Configuration Register on page 39.

6.2 SMBus Network Layer Interface

The I²C/SMB includes two network layer state machines that can automate Master and Slave transactions. Instead of requiring byte-by-byte control by software, the network layer state machines can complete most I²C or SMBus transfers with only one or two interrupts to software for service. The Master and Slave state machines are independent and can operate in parallel. Each is configured by its own register, the Master state machine by the Master Command Register and the Slave state machine by the Slave Command Register. The Master and Slave each use a DMA channel to move data between on-chip memory and the external bus.

Write transactions, in which the Master (either internal or external) claims the I²C bus, sends a Slave address followed by a sequence of data bytes, do not require turning around the I²C bus. The network layer controller, for either Master or Slave transfers, completes the transfer autonomously and can signal software that the transfer has completed with an interrupt. Read transfers require turning the I²C bus around. For these transfers, both the Master and Slave network layer state machines can temporarily halt the transfer, interrupting the EC while stretching the bus clock, until software reconfigures the DMA controller to handle the read phase of the transaction.

The network layer state machines can be configured for many different I²C transaction types, each of which can optionally include PEC calculation. The following section describe a number of examples.

6.2.1 MASTER EXAMPLES

TABLE 6-10: MASTER SMB WRITE BYTE EXAMPLE

	Operation	Description
1.	Software configures a memory buffer with 3 bytes: • The targeted Slave Address, with 0 in bit 0 • The command byte • The data byte	Setting bit 0 of the Slave Address byte to 0 makes the transfer a Write.
2.	Configure the Master DMA channel: • The source address is the buffer in Step 1 • The target address is the Master Transmit Buffer Register • The length is 3	
3.	Enable I²C/SMB interrupt: • Set ENMI in the Configuration Register • Enable I²C/SMB interrupt in the Interrupt Aggregator	This bit enables an interrupt to the EC upon completion of the Master Write transaction.
4.	Set the Master Command Register to: • WRITECOUNT = 3 • START0 = 1 • STOP = 0 • MPROCEED = 1 • MRUN = 1	All other fields in the Master Command Register are 0. This configuration of the Command register asserts a Start bit at the beginning of the transfer, before the slave address, and a Stop bit at the end. Setting MRUN and MPROCEED to 1 is required to start any Master transfer.
5.	Software waits for interrupt	If there are no errors, the Write Byte transaction terminated normally.

TABLE 6-11: MASTER SMB WRITE WORD WITH PEC EXAMPLE

	Operation	Description
1.	Software configures a memory buffer with 4 bytes: • The targeted Slave Address, with 0 in bit 0 • The command byte • The 2 data bytes	Setting bit 0 of the Slave Address byte to 0 makes the transfer a Write.
2.	Configure the Master DMA channel: • The source address is the buffer in Step 1 • The target address is the Master Transmit Buffer Register • The length is 4	
3.	Enable I²C/SMB interrupt: • Set ENMI in the Configuration Register • Enable I²C/SMB interrupt in the Interrupt Aggregator	This bit enables an interrupt to the EC upon completion of the Master Write transaction.
4.	Set the Master Command Register to: • WRITECOUNT = 4 • PEC_TERM = 1 • START0 = 1 • STOP = 0 • MPROCEED = 1 • MRUN = 1	All other fields in the Master Command Register are 0. This configuration of the Command register asserts a Start bit at the beginning of the transfer, before the slave address, and a Stop bit at the end. Setting MRUN and MPROCEED to 1 is required to start any Master transfer.
5.	Software waits for interrupt	If there are no errors, the Write Word transaction terminated normally.

TABLE 6-12: MASTER WRITE BLOCK EXAMPLE

	Operation	Description
1.	Software configures a memory buffer with 4 bytes: • The targeted Slave Address, with 0 in bit 0 • The command byte • The 2 data bytes	Setting bit 0 of the Slave Address byte to 0 makes the transfer a Write.
2.	Configure the Master DMA channel: • The source address is the buffer in Step 1 • The target address is the Master Transmit Buffer Register • The length is 4	
3.	Enable I²C/SMB interrupt: • Set ENMI in the Configuration Register • Enable I²C/SMB interrupt in the Interrupt Aggregator	This bit enables an interrupt to the EC upon completion of the Master Write transaction.

TABLE 6-12: MASTER WRITE BLOCK EXAMPLE (CONTINUED)

	Operation	Description
4.	Set the Master Command Register to: • WRITECOUNT = 4 • PEC_TERM = 1 • START0 = 1 • STOP = 0 • MPROCEED = 1 • MRUN = 1	All other fields in the Master Command Register are 0. This configuration of the Command register asserts a Start bit at the beginning of the transfer, before the slave address, and a Stop bit at the end. Setting MRUN and MPROCEED to 1 is required to start any Master transfer.
5.	Software waits for interrupt	If there are no errors, the Write Block transaction terminated normally.

TABLE 6-13: MASTER READ BLOCK WITH PEC EXAMPLE

	Operation	Description
1.	Software allocates two buffers: • A 3 byte buffer • An *n*+2 byte buffer	The 3 byte buffer is for the write phase of the Read Block operation. Buffer size *n* is the largest permitted block. The extra 2 bytes are for the PEC byte received from the Slave and the PEC calculated by the controller.
2.	Software fills the 3-byte buffer with the Write portion of the Read Block command • Byte 0 is the Slave Address, with bit 0=0 (for Write) • Byte 1 is the Command byte • Byte 2 is the same Slave Address, with bit 0=1 (for Read)	
3.	Configure the Master DMA channel: • The source address is the buffer in Step 1 • The target address is the Master Transmit Buffer Register • The length is 3	
4.	Enable I²C/SMB interrupt: • Set ENMI in the Configuration Register • Enable I²C/SMB interrupt in the Interrupt Aggregator	The I²C/SMB interrupt will occur when the Write phase of the command is complete.
5.	Set the Master Command Register to: • WRITECOUNT = 3 • READCOUNT = 1 • READ_PEC = 1 • READM = 1 • START0 = 1 • STARTN = 1 • STOP = 0 • MPROCEED = 1 • MRUN = 1	All other fields in the Master Command Register are 0. The READCOUNT field will be overwritten by the Slave. The value 1 is sufficient to start the Read phase of the operation. This configuration of the Command register asserts a Start bit at the beginning of the transfer, before the slave address, a repeated Start bit before the second Slave address, and a Stop bit at the end. Setting MRUN and MPROCEED to 1 is required to start any Master transfer.
6.	Software waits for interrupt	If there are no errors, the Write phase of the command has completed normally when the interrupt occurs.

TABLE 6-13: MASTER READ BLOCK WITH PEC EXAMPLE (CONTINUED)

	Operation	Description
7.	Software checks the last 2 bytes of the receive buffer	If the last 2 bytes are the same, then the PEC was calculated correctly
8.	On receipt of the interrupt, software re-configures the Master DMA channel: • The source address is the Master Receive Buffer Register • The target address is the *n*+2 byte buffer in Step 1 • The length is *n* + 2	The Master state machine is still active, but suspended waiting for software to re-configure the DMA channel. The bus clock is stretched until software tells the state machine to proceed.
9.	Enable I²C/SMB interrupt: • Set ENMI in the Configuration Register • Enable I²C/SMB interrupt in the Interrupt Aggregator	The I²C/SMB interrupt will occur when the Read phase of the command is complete.
10.	Set MPROCEED bit in the Master Command Register to 1	All other bits in the Command Register are unchanged. Setting MPROCEED lets the Master state machine complete the transaction.
11.	Software waits for interrupt	If there are no errors, the Read phase of the command has completed normally

6.2.2 SLAVE EXAMPLE

TABLE 6-14: SLAVE CONFIGURATION

	Operation	Description
1.	Set the desired slave address in the Own Address Register	
2.	Software allocates an *n*-byte buffer in memory large enough to accommodate the longest Write phase of a transaction.	If PEC is enabled for Slave transactions, the buffer size should include two bytes for the received and calculated PEC bytes.
3.	Configure the Slave DMA channel: • The source address is the Slave Receive Buffer Register • The target address is the buffer allocated in step 2. • The length is *n*	The DMA length is *n*+2 if PEC is enabled.
4.	Enable I²C/SMB interrupt: • Set ENSI in the Configuration Register • Enable I²C/SMB interrupt in the Interrupt Aggregator	
5.	Set the Slave Command Register to: • SLAVE_READCOUNT = *n* • SPROCEED = 1 • SRUN = 1	If PEC is required, also set SLAVE_PEC to 1.
6.	Software waits for interrupt	If there are no errors, the Write phase of the command has completed normally when the interrupt occurs.
Slave State Machine Terminated when interrupt occurred (Write Transaction, SRUN = 0)		
7.	Software processes received command from Master	

TABLE 6-14: SLAVE CONFIGURATION (CONTINUED)

	Operation	Description
8.	Go to Step 2.	Re-enable the Slave state machine for the next transaction from the Master.
Slave State Machine Active when interrupt occurred (Read Transaction, SRUN = 1)		
7.	Software processes received command from Master	
8.	Software allocates an *m*-byte buffer in memory large enough to accommodate read phase of a transaction.	Software prepares the data to send to the Master
9.	Configure the Slave DMA channel: • The source address is the buffer allocated in step 8. • The target address is the Slave Transmit Buffer Register. • The length is *m*	
10.	Enable I²C/SMB interrupt: • Set ENSI in the Configuration Register • Enable I²C/SMB interrupt in the Interrupt Aggregator	
11.	Set the Slave Command Register to: • SLAVE_WRITECOUNT = *m* • SPROCEED = 1 • SRUN = 1	Set SLAVE_PEC to 1 if PEC calculation is required.
12.	Software waits for interrupt	If there are no errors, the Read phase of the command has completed normally when the interrupt occurs.
13.	Go to Step 2.	Re-enable the Slave state machine for the next transaction from the Master.

6.2.3 ERROR CONDITIONS

If either the LAB or the BER bits in the Status Register become 1 while the Master state machine is processing a transaction, the State Machine clears MRUN, sets the MDONE bit in the Completion Register and halts. An interrupt is generated, if enabled. The I²C Controller Core is reset.

If the Master state machine terminated because the Master lost arbitration for the bus, the LAB bit in the Completion Register is set in addition to the LAB bit in the Status Register. Software should flush the Master Transmit Buffer Register and the Master Receive Buffer Register (using FLUSH_MXBUF and FLUSH_MRBUF) and reconfigure the Master transaction.

6.3 DMA Interface

Both the Master Network Layer state machine and the Slave Network Layer state machine have an interface to an external DMA controller. The DMA Channel assigned to the Master can be used to transfer data between memory and the Master Transmit Buffer Register and the Master Receive Buffer Register, while the Channel assigned to the Slave can be used to transfer data between memory and the Slave Transmit Buffer Register and the Slave Receive Buffer Register.

The DMA Channel and the controller interlocked, so the assigned DMA Channel will terminate the transfer when the I²C/SMB terminates, and the I²C/SMB will terminate a transaction if the DMA Channel halts, due to a bus error or data overrun.

6.4 MCTP Fairness Protocol Support

Support is included for the MCTP Fairness Protocol. The Fairness Protocol is enabled by the FAIR bit in the Configuration Register. Figure 6-2, "Fairness Arbitration Timing Measurements for SMBus and I2C" and Table 6-15, "Fairness Arbitration Timing Values for 100KHz/400KHz SMBus/I2C", taken from Reference [5], illustrate the MCTP Fairness Protocol. The table includes timings for both a 100kHz SMBus and a 400kHz I²C bus:

FIGURE 6-2: FAIRNESS ARBITRATION TIMING MEASUREMENTS FOR SMBUS AND I²C

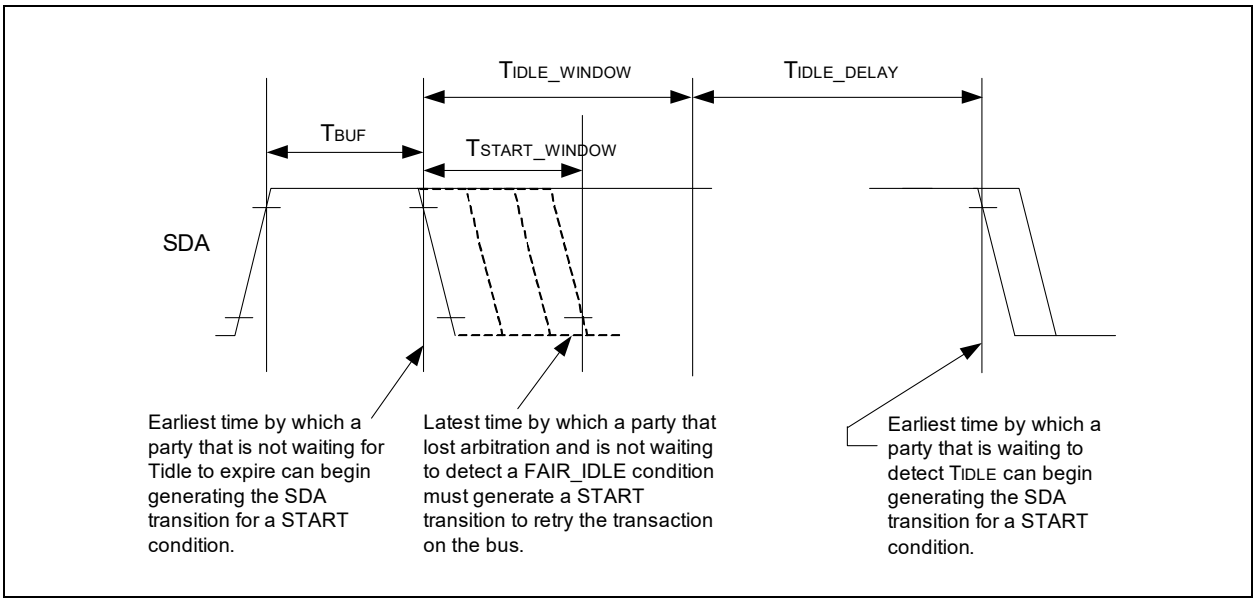


TABLE 6-15: FAIRNESS ARBITRATION TIMING VALUES FOR 100KHZ/400KHZ SMBUS/I²C

Symbol	MIN	MAX	Unit	Notes
T_{BUF}	4.7/1.3	-	µs	Per SMBus 100 kHz/I²C 400 kHz specification
T_{START_WINDOW}	-	20/4	µs	Window of time within which a device that is not waiting to detect a FAIR_IDLE condition must generate START if the device is retrying to gain bus access after losing arbitration.
T_{IDLE_WINDOW}	30/5	60/20	µs	Window of time within which a device that is waiting to detect a FAIR_IDLE condition must not detect a Bus Busy condition. A FAIR_IDLE condition exists when Bus Busy is not detected within this interval.
T_{IDLE_DELAY}	31/16	-	µs	A device that detects FAIR_IDLE condition must wait this delay before attempting to generate START. This delay accommodates the difference between the TIDLE_WINDOW intervals implemented by different devices on the bus, plus additional time to accommodate bus skews between devices that are generating START and devices that are monitoring for it. This guarantees that one party that has detected TIDLE_WINDOW does not generate START before other devices that are detecting FAIR_IDLE have completed checking for their TIDLE window. Otherwise, the other devices would not see a FAIR_IDLE condition even though one occurred. (Therefore TIDLE_DELAY must be greater than the difference between the TIDLE_WINDOW maximum and minimum.)

When Fairness is enabled, an internal state (Fair_Wait) is set whenever the Master state machine wins SMBus arbitration. If Fair_Wait is not true, the bit nBB in the Status register will be de-asserted after the bus is idle for a period determined by the value of BUS_IDLE_MIN in the Time-Out Scaling Register. This value should be set to be the T_{BUF} period.

If Fair_Wait is true, NBB will be de-asserted after a period determined by BUS_IDLE_MIN plus an additional period of FAIR_BUS_IDLE_MIN plus FAIR_IDLE_DELAY. If the bus remains idle after the BUS_IDLE_MIN + FAIR_BUS_IDLE_MIN period expires, the Fair_Wait state is cleared. although the bus must be idle for a period of FAIR_IDLE_DELAY afterwards before NBB is de-asserted.

6.5 Bus Time-Outs

Bus Time-Outs allow the I²C/SMB to implement hardware "Timeouts" as defined in Table 1 and Section 3.1.1.2 of the SMBus Specification (Ref [1]). Time out detection is enabled by the TCEN bit in the Configuration Register. Time-Outs must not be enabled when Bit-Banging is enabled (the BBEN bit in the Bit-Bang Control Register is asserted). Time-Outs are summarized in the following table:

TABLE 6-16: BUS TIME-OUTS

Time-Out	Enable	Status	Scaling Parameter	Symbol in Ref[1]
Bus Idle (SDA = '1')	BIDEN	CHDH	CLOCK_HIGH_TIME_OUT	$T_{HIGH\ (MAX)}$
Data Stuck Low	BIDEN	CHDL	CLOCK_HIGH_TIME_OUT	$T_{HIGH\ (MAX)}$
Slave Cumulative Time-Out	SCEN	SCTO	SLAVE_CUM_TIME_OUT	$T_{LOW:SEXT}$
Master Cumulative Time-Out	MCEN	MCTO	MASTER_CUM_TIME_OUT	$T_{LOW:MEXT}$
Clock Low Timeout	DTEN	DTO	SLAVE_CUM_TIME_OUT	$T_{TIMEOUT}$

If both the Enable bit and the Status bit are both '1' for any of the Time-Outs listed in Table 6-16 the TIMERR bit in the Completion Register is set to '1'. If the controller was actively involved in a transaction as either a Master or a Slave when the Time-Out occurred, the BER bit in the Completion Register is also set.

Typically, a Bus Time-Out indicates a serious problem. If the controller is involved in a transaction when a Time-Out occurs the controller must be reset.

6.5.1 DEVICE TIME-OUT

The controller, when active as a Master in a transaction, can detect the Clock Low Timeout defined in Ref [1], also known ad a Device Time-Out. Both TCEN and DTEN must be '1' for detection to be enabled. The time-out period is defined by the SLAVE_CUM_TIME_OUT value. When the Device Time-Out is enabled and a time-out occurs (i.e., any device, including the master, holds SCL low for $T_{TIMEOUT,\ MIN}$), the Master holds SCL low for an additional period defined by MASTER_CUM_TIME_OUT, in order to guarantee that the error is detected by all devices on the bus.

6.5.2 BUS BUSY

The controller always guarantees a minimum idle time between a STOP bit and a subsequent START bit, independent of the state of the TCEN bit. The idle time is set by the BUS_IDLE_MIN field of the Time-Out Scaling Register and corresponds to the timing parameter T_{BUF} in both Ref [1] and Ref [2]. The NBB bit in the Status Register is de-asserted when he idle time has elapsed after a STOP bit. The NBB bit is de-asserted for transactions initiated by the controller as a master, as well as for transactions initiated by external masters. In addition, the controller, when acting as a Master, will not assert a START bit until at least the idle time period has elapsed after a STOP bit appeared on the bus.

Time-outs that are required for NBB synchronization following a power-on-reset or soft reset are not included as hardware-implemented time-outs and are the responsibility of firmware.

7.0 REGISTER INTERFACE

TABLE 7-1: I²C/SMBUS CONTROLLER REGISTER SUMMARY

Offset (HEX)	Register Name
00h	Control Register
	Status Register
04h	Own Address Register
08h	Data Register
0Ch	Master Command Register
10h	Slave Command Register
14h	PEC Register
18h	Repeated START Hold Time Register
1Ch	Microchip Reserved
20h	Completion Register
24h	Idle Scaling Register
28h	Configuration Register
2Ch	Bus Clock Register
30h	Block ID Register
34h	Revision Register
38h	Bit-Bang Control Register
3Ch	Microchip Reserved Register
40h	Data Timing Register
44h	Time-Out Scaling Register
48h	Slave Transmit Buffer Register
4Ch	Slave Receive Buffer Register
50h	Master Transmit Buffer Register
54h	Master Receive Buffer Register
58h	Microchip Reserved
5Ch	Microchip Reserved
60h	Wake Status Register
64h	Wake Enable Register
68h	Microchip Reserved Register

Registers marked “Microchip Reserved” must not be modified.

TABLE 7-2: RECOMMENDED PROGRAMMING VALUES

20MHz BAUD Clock:				
Register	**Default**	**Value for 100K**	**Value for 400K**	**Value for 1M (Note 1:)**
Bus Clock Register	00_00_65_65h	00_00_65_65h	00_00_13_20h	00_00_09_0Ah
Data Timing Register	0E_54_62_0Ah	0E_54_62_0Ah	0E_15_15_08h	06_08_08_01h
Repeated START Hold Time Register	00_00_00_56h	00_00_00_56h	00_00_00_0Eh	00_00_00_09h
Idle Scaling Register	02_80_02_6Ch	02_80_02_6Ch	00_14_00_64h	00_14_00_64h

Note 1: MCTP is defined at 100kHz and 400kHz and is not yet defined to support I²C Fast-mode Plus mode operation per Reference [5]. Therefore this value may not be meaningful for I²C Fast-mode Plus at 1MHz.

TABLE 7-2: RECOMMENDED PROGRAMMING VALUES (CONTINUED)

Time-Out Scaling Register	30_C6_F5_FBh	30_C6_F5_FBh	0D_C6_F5_FBh	06_C6_F5_FBh
Microchip Reserved Register	7h			
16MHz BAUD Clock:				
Register	**Default**	**Value for 100k**	**Value for 400k**	**Value for 1M (Note 1)**
Bus Clock Register	00_00_4F_4Fh	00_00_4F_4Fh	00_00_0F_17h	00_00_05_09h
Data Timing Register	0C_4D_50_06h	0C_4D_50_06h	04_0A_0A_06h	04_06_06_01h
Repeated START Hold Time Register	00_00_00_4Dh	00_00_00_4Dh	00_00_00_0Ah	00_00_00_06h
Idle Scaling Register	01_FC_01_EDh	01_FC_01_EDh	01_00_00_50h	01_00_00_50h (See Note 1)
Time-Out Scaling Register	4B_9C_C2_C7h	4B_9C_C2_C7h	15_9C_C2_C7h	08_9C_C2_C7h
Microchip Reserved Register	5h			
10MHz BAUD Clock:				
Register	**Default**	**Value for 100k**	**Value for 400k**	
Bus Clock Register	00_00_31_31h	00_00_31_31h	00_00_0A_0Dh	N/A
Data Timing Register	07_2A_31_04h	07_2A_31_04h	03_08_08_04h	N/A
Repeated START Hold Time Register	00_00_00_2Bh	00_00_00_2Bh	00_00_00_07h	N/A
Idle Scaling Register	01_40_01_36h	01_40_01_36h	00_A0_00_32h	N/A
Time-Out Scaling Register	30_C6_F5_FBh	30_C6_F5_FBh	0D_C6_F5_FBh	N/A
Microchip Reserved Register	2h			

Note 1: MCTP is defined at 100kHz and 400kHz and is not yet defined to support I²C Fast-mode Plus mode operation per Reference [5]. Therefore this value may not be meaningful for I²C Fast-mode Plus at 1MHz.

TABLE 7-3: BUS CLOCK REGISTER VS. FREQUENCY

Bus Frequency (KHz)	Bus Clock Register Setting		
	10MHz Baud Clock	**16MHz Baud Clock**	**20MHz Baud Clock**
1000	N/A	0509h	090Ah
400	0A0Dh	0F17h	1320h
333	0A12h	0F1Fh	1626h
100	3131h (default)	4F4Fh (default)	6565h (default)
80	3D3Eh	6363h	7C7Dh
40	7C7Ch	C7C7h	F9F9h

7.1 Register Description

7.1.1 CONTROL REGISTER

Access to this register is not required if Network Layer functions (using the Master Command Register and the Slave Command Register) are used.

Offset	00h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7	PIN The Pending Interrupt Not bit serves as a software reset function. Writing the PIN bit to a logic '1' de-asserts all status bits except for the HBB bit which is not affected by the PIN bit. The PIN bit is a self-clearing bit. Writing this bit to a logic '0' has no effect.	W	N/A	RESET_I2C
6	ESO The Enable Serial Output bit enables and disables the serial data output (SDA). When ESO is asserted ('1'), SDA is enabled. When ESO is not asserted ('0') SDA is disabled. The ESO bit does not affect access to registers in the controller. The ESO bit must not be de-asserted when the I²C/SMB Controller is actively involved in a bus transaction.	W	N/A	RESET_I2C
5:4	Reserved	R	-	-
3	ENI The Enable Interrupt bit controls the Interrupt Interface as described in Section 5.0, "Interrupt Interface".	W	N/A	RESET_I2C
2	STA Transmit START. The STA and STO bits control the generation of the I²C Start condition and the transmission of the Slave Address and R/nW bit (from the Data Register), generation of repeated Start condition, and generation of the Stop condition as described in Table 7-4.	W	N/A	RESET_I2C
1	STO Transmit STOP. The STA and STO bits control the generation of the I²C Start condition and the transmission of the Slave Address and R/nW bit (from the Data Register), generation of repeated Start condition, and generation of the Stop condition as described in Table 7-4.	W	N/A	RESET_I2C
0	ACK Acknowledge. The ACK must normally be asserted ('1'). This causes the controller to send an acknowledge automatically after each byte (this occurs during the 9th clock pulse). The ACK bit must be de-asserted ('0') when the controller is operating in master/receiver mode and requires no further data to be sent from the slave transmitter. This causes a negative acknowledge on the I²C bus, which halts further transmission from the slave device.	W	N/A	RESET_I2C

TABLE 7-4: INSTRUCTION TABLE FOR SERIAL BUS CONTROL

STA	STO	Mode	Function	Operation
1	0	SLV/REC	START	Transmit START+address, remain MST/TRM if Data Register bit 0 (R/nW) = 0; go to MST/REC if Data Register bit 0 (R/nW) =1.
		MST/TRM	REPEAT START	Same as for SLV/REC
0	1	MST/REC; MST/TRM	STOP READ; STOP WRITE	Transmit STOP go to SLV/REC mode. In Master Receive mode, the last received byte must be terminated with a NACK.
1	1	Reserved		Reserved
0	0	ANY	NOP	No operation

All other STA and STO mode combinations not mentioned in Table 7-4 are NOPs.

7.1.2 STATUS REGISTER

Use of this register is only required for legacy I²C operation. Access to this register is not required if Network Layer functions are used.

Offset	00h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7	PIN The operation of PIN (Pending Interrupt) is described in Section 6.1.1, "PIN Operation".	R	1h	RESET_I2C
6	SAD SAD (SMBus Address Decoded) is asserted as described in Section 6.1.2, "Slave Address Decoding".	R	0h	RESET_I2C
5	STS This bit is asserted (‘1’) when an externally generated STOP condition is detected. It is used only in slave receiver mode.	R	0h	-
4	BER When BER (Bus Error) is asserted, a misplaced START or STOP condition or Bus Time-outs have been detected. If this bit is asserted, NBB (‘1’) and PIN (‘0’) are de-asserted. After BER is asserted, the controller must be reset before taking any further action.	R	0h	-

Offset	00h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
3	LRB/AD0 LBR/AB0 (“Last Received Bit” or “Address 0”) is valid only while the PIN bit is asserted (‘0’). When the AAS bit is not asserted (‘0’) (i.e., the controller has not addressed as a slave), this bit holds the value of the last received bit over the bus. Normally this will be the value of the slave acknowledgment; thus, checking for slave acknowledgment is done via testing this bit. When the AAS bit is asserted (‘1’) (i.e., the controller has been addressed as a slave), this bit will be set to logic ‘1’ if the slave address received was the ‘general call’ (00h) address, or logic ‘0’ if the received slave address matches the value programmed in the Own Address Register.	R	0h	RESET_I2C
2	AAS AAS (Addressed As Slave) is valid only when PIN is asserted (‘0’). When acting as a slave, this bit is set when an incoming address over the bus matches the value in the Own Address Register (shifted by one bit) or if the ‘general call’ address (00h) has been received (‘general call’ is indicated by the LRB/AD0 bit)	R	0h	RESET_I2C
1	LAB LAB (Lost Arbitration) is set when, in multi-master operation, arbitration is lost to another master on the bus as described in Section 6.1.7, "I²C Multi-Master Support".	R	0h	RESET_I2C
0	NBB The Bus Busy bit indicates when the bus is in use. A zero indicates that the bus is busy and access is not possible. This bit is asserted (‘0’) by a START condition and de-asserted (‘1’) following a STOP condition. See Section 6.5.2, "Bus Busy".	R	1h	RESET_I2C

7.1.3 OWN ADDRESS REGISTER

This register provides the ability for the I²C/SMB to respond to two different slave addresses. See Section 6.1.2, "Slave Address Decoding".

Note: The internal master should not attempt transactions to a slave with the same address as one that is programmed in the **Own Address Register**. This represents an illegal operation. If it occurs the controller must be reset.

Offset	04h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:15	Reserved	R	-	-
14:8	OWN_ADDRESS_2 This field configures one of the two addresses to which the controller will respond when addressed as a slave. The Own Address fields is offset by one bit, so that programming this field with a value of 55h will result in the value AAh being recognized as the slave address. The AAS bit in the Status Register is set when this field matches an incoming slave address. The address '0000000b' is the General Call Address. An Own_Address of 0h will match the General Call Address unless this match is disabled by setting GC_DIS in the Configuration Register.	R/W	0h	RESET_I2C
7	Reserved	R	-	-
6:0	OWN_ADDRESS_1 This field configures one of the two addresses to which the controller will respond when addressed as a slave. The Data Register and Own Address fields are offset by one bit, so that programming this field with a value of 55h will result in the value AAh being recognized as the slave address. The AAS bit in the Status Register is set when this field matches an incoming slave address. The address '0000000b' is the General Call Address. An Own_Address of 0h will match the General Call Address unless this match is disabled by setting GC_DIS in the Configuration Register.	R/W	0h	RESET_I2C

7.1.4 DATA REGISTER

Offset	08h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	DATA This register holds the data that are either shifted out to or shifted in from the I²C port.	R/W	0h	RESET_I2C

7.1.5 MASTER COMMAND REGISTER

Offset	0Ch			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
31:24	READ_COUNT This field is a count of the number of bytes to read in from the I²C port to the Master Receive Buffer Register and must be greater than 0 in order for the Master State Machine to initiate a read phase. It is decremented by 1 for each byte read into the Master Receive Buffer Register. It can be overwritten by the first byte read in if the READM bit is set. The Read Count is a hard limit for block reads. Any data received beyond this value will be ignored. READ_COUNT does not include a PEC byte, so the number of bytes read from the Slave is (READ_COUNT + READ_PEC). The number of bytes copied into to memory is (READ_COUNT + 2XREAD_PEC), since the state machine copies the PEC Register into memory after it copies a PEC byte from the Slave.	R/W	0h	RESET_I2C
23:16	WRITE_COUNT This field is a count of the number of bytes to transmit to the I²C/SMBus port from the Master Transmit Buffer Register It is decremented by 1 for each byte written to the port from the Master Transmit Buffer Register.	R/W	0h	RESET_I2C
15:14	Reserved	R	-	-
13	READ_PEC If this bit is '0', reading from the I²C/SMBus port stops when READ_COUNT reaches '0'. If this bit is '1', reading continues after READ_COUNT is '0' for one more byte. This enables the controller to read a PEC returned from an external device after it has read an M-byte block. Once the PEC byte is read this field is cleared to '0'.	R/W	0h	RESET_I2C
12	READM If this bit is '1', the READ_COUNT field is replaced by the byte that is read from the I²C/SMBus port when READ_COUNT is '1'. After READ_COUNT is updated, this bit is cleared to '0'.	R/W	0h	RESET_I2C
11	PEC_TERM If this bit is '1', a copy of the PEC Register is transmitted when WRITE_COUNT is 0. After the PEC Register data are transmitted, both the PEC Register and this bit are cleared to '0'.	R/W	0h	RESET_I2C
10	STOP If this bit is '1', send a Stop bit after the transaction completes.	R/W	0h	RESET_I2C
9	STARTN If this bit is '1', send a Start bit before the last byte of the WRITE_COUNT data is sent.	R/W	0h	RESET_I2C
8	START0 If this bit is '1', send a Start bit before the first byte of the WRITE_COUNT data is sent.	R/W	0h	RESET_I2C
7:2	Reserved	R	-	-

Offset	0Ch			
Bits	**Description**	**Type**	**Default**	**Reset Event**
1	MPROCEED In order to start a Master I²C/SMB transaction, both this bit and the MRUN bit in this register must be set to '1'. This bit is cleared by hardware when the Master state machine stops, either because the transaction has completed, or because the transaction has reached a point where it requires software assistance. If the state machine clears this bit but leaves MRUN set (for example, after a repeated START in a Read transfer, which requires software to reconfigure the Master DMA channel), software re-starts the transaction by writing a '1' to this bit. MRUN must remain '1' for the transaction to complete. Software must never write this bit with a '0'.	R/W	0h	RESET_I2C
0	MRUN In order to start a Master I²C/SMB transaction, both this bit and the MPROCEED bit in this register must be set to '1'. Hardware clears this bit to '0' when the Master transaction completes, either successfully or due to an error condition. Software must never write this bit with a '0'.	R/W	0h	RESET_I2C

7.1.6 SLAVE COMMAND REGISTER

The Slave Command register configures how the controller responds as a Slave device.

Offset	10h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:24	Reserved	R	-	-
23:16	SLAVE_READCOUNT This field is decremented each time a byte is copied from the DATA register to the Slave Receive Buffer Register. When the field is decremented to '0' the state machine will NACK any additional received data. If a value in this field is one less than the number of bytes and external Master is transmitting, it is possible that a repeated start address will be NACK'd. SLAVE_READ_COUNT should always be large enough for the largest expected transfer from an external Master. Once the external Master issues a STOP bit, a bad count will not cause the Slave to NACK its own slave address on a subsequent START.	R/W	0h	RESET_I2C

Offset	10h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
15:8	SLAVE_WRITECOUNT This field is set to the number of bytes software expects to send to the Master. If if is greater than '0' when the external Master requests data and the Slave Transmit Buffer Register is empty, the state machine stretches the I²C/SMBus clock until the buffer is not empty. If this field is greater than '0' when the Master requests data and the Buffer Register is full, the register is copied to the DATA register and this field is decremented by 1. If both SLAVE_WRITE_COUNT and SLAVE_PEC are '0' when the Master requests data, the slave resends the contents of the Slave Transmit Buffer Register to the Master. If this occurs, the SPROT bit in the Completion Register is set to '1'.	R/W	0h	RESET_I2C
7:3	Reserved	R	-	-
2	SLAVE_PEC If SLAVE_WRITE_COUNT is '0' and this bit is '1' when the Master requests data, the PEC Register is copied to the DATA register. After the PEC Register is sent to the Master, the register is cleared and this bit is set to '0'.	R/W	0h	RESET_I2C
1	SPROCEED In order to start a Slave I²C/SMB transaction, both this bit and the SRUN bit in this register must be set to '1'. This bit is cleared by hardware when the Master state machine stops, either because the transaction has completed, or because the transaction has reached a point where it requires software assistance. If the state machine clears this bit but leaves SRUN set (for example, after a repeated START in a Read or Write transfer, which requires software to reconfigure the Slave DMA channel), software re-starts the transaction by writing a '1' to this bit. SRUN must remain '1' for the transaction to complete. Software must never write this bit with a '0'.	R/W	0h	RESET_I2C
0	SRUN In order to enable a Slave I²C/SMB transaction, both this bit and the SPROCEED bit in this register must be set to '1'. Hardware clears this bit to '0' when the Slave transaction completes, either successfully or due to an error condition. Software must never write this bit with a '0'.	R/W	0h	RESET_I2C

7.1.7 PEC REGISTER

Offset	14h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	PEC The SMBus Packet Error Check. See Section 6.1.4, "Hardware PEC Support" for details.	R/W	0h	RESET_I2C

7.1.8 REPEATED START HOLD TIME REGISTER

Offset	18h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	RPT_START_HOLD_TIME This is the value of the timing requirement $t_{Hd:Sta}$ in the I²C specification for a repeated START bit. This is used to hold the clock until the Hold Time for the repeated Start Bit has been satisfied. See Table 7-2, "Recommended Programming Values" for recommended values for this field for different clock rates.	R/W	See Table 7-2	RESET_I2C

7.1.9 COMPLETION REGISTER

Offset	20h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31	SDONE Slave Done. 1=Slave transaction completed 0=Slave transaction not completed	R/WC	0h	RESET_I2C
30	MDONE 1=Master transaction completed 0=Master transaction not completed	R/WC	0h	RESET_I2C
29	IDLE This bit is set to '1' when: • The bus becomes idle (on the rising edge of NBB in the Status Register) • The bus was idle when the Idle Interrupt was enabled • The bus was idle when the controller was enabled (the ENAB bit in the Configuration Register was set to '1')	R/WC	0h	RESET_I2C
28:26	Reserved	R	-	-
25	MTR Master Transmit/Receive. This bit reports the phase of the Master State Machine when it asserts MDONE. 1=Master has just finished the transmit phase of a transaction 0=Master has just finished the receive phase of a transaction	R	0h	RESET_I2C
24	MNAKX Master received a NACK while Transmitting. 1=The external Slave sent a NACK to the Master while the Master was transmitting data over the I²C/SMBus port 0=No NACK was received during the Master transmission	R/WC	0h	RESET_I2C
23:22	Reserved	R	-	-

Offset	20h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
21	REPEAT_WRITE 1=The Slave state machine paused because it detected a Repeat START Write (bit[0] of the byte containing the Slave address was '0') 0=No Repeat Write detected	R/WC	0h	RESET_I2C
20	REPEAT_READ 1=The Slave state machine paused because it detected a Repeat START Read (bit[0] of the byte containing the Slave address was '1') 0=No Repeat Read detected	R/WC	0h	RESET_I2C
19	SPROT Slave Protocol error in write count. 1=The WRITE_COUNT field in the Slave Command Register either counted down to 0 before the external Master sent a NACK signal, or the Slave received a NACK signal before the counter reached 0 0=No error in write count occurred	R/WC	0h	RESET_I2C
18	Reserved	R	-	-
17	STR Slave Transmit/Receive 1=The Slave just finished the receive phase of a transaction 0=The Slave just finished the transmit phase of a transaction	R	0h	RESET_I2C
16	SNAKR Slave NACK sent while Receiving 1=The Slave state machine sent a NACK to the external transmitting Master while the Slave was receiving data from the I²C/SMBus port 0=A NACK was not sent to the Master	R/WC	0h	RESET_I2C
15	Reserved	R	-	-
14	LAB Lost Arbitration Status. This bit is set to '1' if the LAB bit in the Status Register was set to '1' while either a Master or a Slave transaction was in progress.	R/WC	0h	RESET_I2C
13	BER Bus Error Status. This bit is set to '1' if the BER bit in the Status Register was set to '1' while either a Master or a Slave transaction was in progress.	R/WC	0h	RESET_I2C
12	CHDH Clock High Data High time-out status.This is the bus idle time-out status. See.Section 6.5, "Bus Time-Outs". 1=Time-Out occurred 0=Time-Out did not occur	R/WC	0h	RESET_I2C

Offset	20h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
11	CHDL Clock High Data Low time-out status. This status is asserted if the SCL remains high while SDA remains low. See.Section 6.5, "Bus Time-Outs". 1=Time-Out occurred 0=Time-Out did not occur	R/WC	0h	RESET_I2C
10	SCTO Slave Cumulative Time-Out status. See Section 6.5, "Bus Time-Outs". 1=Time-Out occurred 0=Time-Out did not occur	R/WC	0h	RESET_I2C
9	MCTO Master Cumulative Time-Out status. See.Section 6.5, "Bus Time-Outs". 1=Time-Out occurred 0=Time-Out did not occur	R/WC	0h	RESET_I2C
8	DTO Device Time-Out status. See Section 6.5, "Bus Time-Outs". 1=Time-Out occurred 0=Time-Out did not occur	R/WC	0h	RESET_I2C
7	Reserved	R	-	-
6	TIMERR Time Out Error Detected. This bit is ‘1’ if any of the Time-Out status bits (CHDH, CHDL, MCTO and DTO) are both asserted ‘1’ and enabled by their respective Time Out Enable bits.	R	0h	RESET_I2C
5	BIDEN This bit enables Bus Idle Time-Out checking, as described in.Section 6.5, "Bus Time-Outs". 1=Time-Out checking is enabled 0=Time-Out checking is disabled	R/W	0h	RESET_I2C
4	SCEN This bit enables Slave Cumulative Time-Out checking, as described in Section 6.5, "Bus Time-Outs". 1=Time-Out checking is enabled 0=Time-Out checking is disabled	R/W	0h	RESET_I2C
3	MCEN This bit enables Master Cumulative Time-Out checking, as described in Section 6.5, "Bus Time-Outs". 1=Time-Out checking is enabled 0=Time-Out checking is disabled	R/W	0h	RESET_I2C

Offset	20h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
2	DTEN This bit enables Device Time-Out checking, as described in Section 6.5, "Bus Time-Outs". 1=Time-Out checking is enabled 0=Time-Out checking is disabled	R/W	0h	RESET_I2C
1:0	Reserved	R	-	-

7.1.10 IDLE SCALING REGISTER

Offset	24h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:28	Reserved	R	-	-
27:16	FAIR_IDLE_DELAY This field establishes the MCTP T_{IDLE_DELAY} period. When the MCTP Fairness protocol is enabled, this field causes the start of master transaction to be delayed by an additional period of T_{IDLE_DELAY} after the state machine has detected an idle period of FAIR BUS IDLE MIN. This field defines the number of ticks of the baud clock required to program the delay. See Table 7-2, "Recommended Programming Values" for recommended values for this field for different clock rates.	R/W	See Table 7-2	RESET_I2C
15:12	Reserved	R	-	-
11:0	FAIR_BUS_IDLE_MIN The bus idle period should be programmed to the T_{IDLE_WINDOW} time as defined in Table 6-15, "Fairness Arbitration Timing Values for 100KHz/400KHz SMBus/I2C". This field defines the number of ticks of the baud clock required to satisfy the fairness protocol. See Table 7-2, "Recommended Programming Values" for recommended values for this field for different clock rates.	R/W	See Table 7-2	RESET_I2C

7.1.11 CONFIGURATION REGISTER

Offset	28h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31	ENSI Enable Slave Done Interrupt. 1=Slave Done Interrupt enabled 0=Slave Done Interrupt disabled	R/W	0h	RESET_I2C
30	ENMI Enable Master Done Interrupt. 1=Master Done Interrupt enabled 0=Master Done Interrupt disabled	R/W	0h	RESET_I2C
29	ENIDI Enable Idle Interrupt. This interrupt should not be used if either MRUN in the Master Command Register or SRUNin the Slave Command Register are set. 1=Idle Interrupt enabled 0=Idle Interrupt disabled	R/W	0h	RESET_I2C
28	ENABLE_AAS Interrupt Enable for the Addressed As Slave interrupt. The interrupt is asserted if the AAS bit in the Status Register is asserted. 1=AAS Interrupt enabled 0=AAS Interrupt disabled	R/W	0h	RESET_I2C
27:20	Reserved	R	-	-
19	FLUSH_MRBUF A write of '1' to this bit clears the Master Receive Buffer Register and marks it empty. It is self-clearing. A write of '0' has no effect.	W	0h	RESET_I2C
18	FLUSH_MXBUF A write of '1' to this bit clears the Master Transmit Buffer Register and marks it empty. It is self-clearing. A write of '0' has no effect.	W	0h	RESET_I2C
17	FLUSH_SRBUF A write of '1' to this bit clears the Slave Receive Buffer Register and marks it empty. It is self-clearing. A write of '0' has no effect.	W	0h	RESET_I2C
18	FLUSH_SXBUF A write of '1' to this bit clears the Slave Transmit Buffer Register and marks it empty. It is self-clearing. A write of '0' has no effect.	W	0h	RESET_I2C
15	Reserved	R	-	-
14	GC_DIS General Call disable 1=The response to the General Call address (0h) as a slave is enabled. 0=The response to the General Call address (0h) as a slave is disabled.	R/W	0h	RESET_I2C
13	Microchip Reserved Must be always written with 0.	R/W	0h	RESET_I2C

Offset	28h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
12	FAIR 1=MCTP Fairness protocol is enabled 0=MCTP Fairness protocol is disabled	R/W	0h	RESET_I2C
11	DSA Decode SMBus address. 1=The Slave controller will match the slave addresses in the Own Address Register, the General Call Address (if enabled by GC_DIS) and two SMBus-defined addresses, the SMBus Host Address (0001_000b) and the SMBus Device Default Address (110_001b) 0=The Slave controller will match only the slave addresses in the Own Address Register and the General Call Address (if enabled by GC_DIS)	R/W	0h	RESET_I2C
10	ENAB I²C/SMBus controller enable. 1=Controller is enabled 0=Controller disabled and in its lowest power state	R/W	0h	RESET_I2C
9	RESET When RESET is asserted (‘1’), all logic and registers except for the RESET bit itself are initialized to the power-on default state. RESET is not self-clearing. It must remain asserted for at least one baud clock period. Register reads while RESET is asserted return default register values. The RESET bit is itself reset on the main system reset and not on the block reset signal, RESET_I2C.	R/W	0h	RESET_EC
8	FEN Input filtering enable. See Section 6.1.8, "Input Filtering". Input filtering is required by the I²C specification if external filtering is not available. 1=Input filtering is enabled 0=Input filtering is disabled	R/W	0h	RESET_I2C
7	PCEN PEC Enable bit. PCEN should only be changed when the PIN bit in the Status Register is de-asserted or when the I²C/SMB Controller is not involved in a transaction. 1=Hardware PEC support is enabled 0=Hardware PEC support is disabled	R/W	0h	RESET_I2C
6	Microchip Reserved Must be always written with 0.	R/W	0h	RESET_I2C

Offset	28h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
5	SLOW_CLOCK 1=The base period for the Bus Clock Register is multiplied by 4, and thus the frequency is divided by 4. It does not affect other timing calculations (such as those in the Data Timing Register or the Time-Out Scaling Register 0=The base period for the Bus Clock Register is the controller's baud clock	R/W	0h	RESET_I2C
4	TCEN Timing Check Enable. 1=Bus Time-Outs, as described in Section 6.5, "Bus Time-Outs", are enabled 0=Bus Time-Outs are disabled	R/W	0h	RESET_I2C
3:0	PORT_SEL This field selects which of the 16 possible bus ports is the currently active I²C/SMBus port. Unselected bus ports remain inactive. See the specific part Data Sheet for the configuration of ports per I²C/SMBus controller.	R/W	0h	RESET_I2C

Note:	All fields in this register should be set to their desired values before setting the ENAB bit in this register to '1b'. Failure to do so could result in glitches on the signal interface, and incorrect state machine behavior.

7.1.12 BUS CLOCK REGISTER

The Bus Clock Register is used to determine I²C/SMBus bus clock when the controller is a master. The bus clock period is determined by the following equation:

EQUATION 7-1: $\mathbf{BUS\_CLOCK_{PERIOD} = ((LOW\_PERIOD+1) + (HIGH\_PERIOD+1)) \times I2C\_BAUD\_CLOCK_{PERIOD}}$

Examples of appropriate settings for the Bus Clock Register for several desired bus frequencies are shown in Table 7-3, "Bus Clock Register vs. Frequency".

Changes to the Bus Clock Register must only occur when the ESO bit in the Control Register is not asserted; i.e., the register must not be changed in the middle of a transaction. In order to comply with the I²C, SMBus and MCTP specifications, the registers listed in Table 7-2, "Recommended Programming Values" may have to be adjusted when changing bus speeds.

If the SLOW_CLOCK bit in the Configuration Register is set to '1', the $I2C\_BAUD\_CLOCK_{PERIOD}$ term is multiplied by 4, and the frequencies shown in Table 7-3, "Bus Clock Register vs. Frequency" are divided by 4.

Offset	2Ch			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:16	Reserved	R	-	-
15:8	HIGH_PERIOD This field defines the number of I2C_BAUD_CLOCK periods that make up the high phase of the I²C/SMBus bus clock. The number of clock periods is one greater than the contents of this field. When both HIGH_PERIOD and LOW_PERIOD are 00h, the bus clock is disabled and master transactions cannot occur.	R/W	See Table 7-2	RESET_I2C
7:0	LOW_PERIOD This field defines the number of I2C_BAUD_CLOCK periods that make up the low phase of the I²C/SMBus bus clock. The number of clock periods is one greater than the contents of this field	R/W	See Table 7-2	RESET_I2C

7.1.13 BLOCK ID REGISTER

Offset	30h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	ID Block ID	R	11h	RESET_I2C

7.1.14 REVISION REGISTER

Offset	34h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	REVISION Block Revision Number	R	Current Revision	RESET_I2C

7.1.15 BIT-BANG CONTROL REGISTER

Offset	38h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:7	Reserved	R	-	-
6	BBDATI Bit-Bang Data In. This bit always returns the state of the SDA pin.	R	1h	RESET_I2C
5	BBCLKI Bit-Bang Clock In. This bit always returns the state of the SCL pin.	R	1h	RESET_I2C

Offset	38h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
4	BBDAT Bit-Bang Mode Data.state. 1=The SDA pin is tri-stated 0=If BBEN='1' and DADIR='1', the SDA pin is driven low. If either BBEN='1' or DADIR='1', the SDA pin is tri-stated	R/W	0h	RESET_I2C
3	BBCLK Bit-Bang Mode Clock.state. 1=The SCL pin is tri-stated 0=If BBEN='1' and DADIR='1', the SCL pin is driven low. If either BBEN='1' or DADIR='1', the SCL pin is tri-stated	R/W	0h	RESET_I2C
2	DADIR Bit-Bang Mode Data direction. 1=If BBEN='1', the SDA pin is an output 0=If BBEN='1', the SDA pin is an input	R/W	0h	RESET_I2C
1	CLDIR Bit-Bang Mode Clock direction. 1=If BBEN='1', the SCL pin is an output 0=If BBEN='1', the SCL pin is an input	R/W	0h	RESET_I2C
0	BBEN Bit-Bang Mode Enable. 1=Bit-Bang Mode Enabled 0=Bit Bang Mode Disabled	R/W	0h	RESET_I2C

7.1.16 MICROCHIP RESERVED REGISTER

Offset	3Ch			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	MICROCHIP_RSVD This register must not be written, or undesirable results may occur.	R/W	See Table 7-2	RESET_I2C

7.1.17 DATA TIMING REGISTER

Recommended values for the Data Timing Register are shown in Table 7-2, "Recommended Programming Values".

Offset	40h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:24	FIRST_START_HOLD This field determines the SCL hold time following SDA driven low during the first START bit in a transfer. It is the parameter $T_{HD:STA}$ in the I²C Specification for an initial START bit. Repeated START hold time is determined by the Repeated START Hold Time Register.	R/W	See Table 7-2	RESET_I2C
23:16	STOP_SETUP This field determines the SDA setup time from the rising edge of SCL for a repeated START condition. It is the parameter $T_{SU:STOP}$ in the I²C Specification.	R/W	See Table 7-2	RESET_I2C
16:8	RESTART_SETUP This field determines the SDA setup time from the rising edge of SCL for a repeated START condition. It is the parameter $T_{SU:STA}$ in the I²C Specification.	R/W	See Table 7-2	RESET_I2C
7:0	DATA_HOLD This field determines the SDA hold time following SCL driven low. It is the parameter $T_{HD:DAT}$ in the I²C Specification.	R/W	See Table 7-2	RESET_I2C

7.1.18 TIME-OUT SCALING REGISTER

Time-Outs and Bus Busy calculation are described in Section 6.5, "Bus Time-Outs". Recommended values for the Time-Out Scaling Register are shown in Table 7-2, "Recommended Programming Values".

The time-outs MASTER_CUM_TIME_OUT, SLAVE_CUM_TIME_OUT and CLOCK_HIGH_TIME_OUT are only defined in the SMBus Specification, and not in the I²C Specification. Since SMBus is only defined for 100KHz, these time-outs are only defined by specification for bus speeds of 100 KHz or less.

The Baud_Clock_Period is the cycle time of the I2C_BAUD_CLOCK.

Offset	44h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:24	BUS_IDLE_MIN This field determines the minimum bus idle time, also defined as the time between a STOP and START condition. It is the parameter T_{BUF} in the I²C Specification. It is defined here as BUS_IDLE_MIN × Baud_Clock_Period	R/W	See Table 7-2	RESET_I2C
23:16	MASTER_CUM_TIME_OUT This field determines the Master Cumulative Time-Out Duration, or the parameter $T_{LOW:MEXT}$ parameter in the SMBus specification. It is defined here as MASTER_CUM_TIME_OUT × Baud_Clock_Period × 512 This Time-Out is only detected when the controller is acting as a Master. The active transaction is not abandoned when this Time-Out is detected; software can chose to do so.	R/W	See Table 7-2	RESET_I2C

Offset	44h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
16:8	SLAVE_CUM_TIME_OUT This field determines the Slave Cumulative Time-Out Duration, or the parameter $T_{LOW:SEXT}$ parameter in the SMBus specification. It is defined here as SLAVE_CUM_TIME_OUT × Baud_Clock_Period × 1024 If this Time-Out is detected while the controller is acting as a Slave, the transaction is abandoned and the bus is released. If this Time-Out is detected while the controller is acting as a Master, the transaction is not abandoned, to accommodate slave devices that may not implement the Slave Cumulative Time-Out.	R/W	See Table 7-2	RESET_I2C
7:0	CLOCK_HIGH_TIME_OUT The Clock High Time Out period, or the parameter T_{HIGH} parameter in the SMBus specification. It is defined here as CLOCK_HIGH_TIME_OUT × Baud_Clock_Period × 2	R/W	See Table 7-2	RESET_I2C

7.1.19 SLAVE TRANSMIT BUFFER REGISTER

Offset	48h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	SLAVE_TRANSMIT_BUFFER	R/W	0h	RESET_I2C

7.1.20 SLAVE RECEIVE BUFFER REGISTER

Offset	4Ch			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	SLAVE_RECEIVE_BUFFER	R/W	0h	RESET_I2C

7.1.21 MASTER TRANSMIT BUFFER REGISTER

Software should make the Master Transmit Buffer empty (using FLUSH_MXBUF in the Configuration Register) if a Master transaction is terminated because of Lost Arbitration.

Offset	50h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	MASTER_TRANSMIT_BUFFER	R/W	0h	RESET_I2C

7.1.22 MASTER RECEIVE BUFFER REGISTER

Offset	54h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	MASTER_RECEIVE_BUFFER	R/W	0h	RESET_I2C

7.1.23 WAKE STATUS REGISTER

Offset	60h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:1	Reserved	R	-	-
0	START_BIT_DETECTION This bit is set to '1' when a START bit is detected while the controller is enabled. This bit is cleared to '0' when written with a '1'. Writes of '0' have no effect.	R/WC	0h	RESET_I2C

7.1.24 WAKE ENABLE REGISTER

Offset	64h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:1	Reserved	R	-	-
0	START_DETECT_INT_EN Enable Start Bit Detection Interrupt. The Start Bit Detection Interrupt is wake-capable. 1=Start Bit Detection Interrupt enabled 0=Start Bit Detection Interrupt disabled	R/W	0h	RESET_I2C

8.0 SMBUS INTERFACE TIMINGS

FIGURE 8-1: I²C/SMBUS TIMING

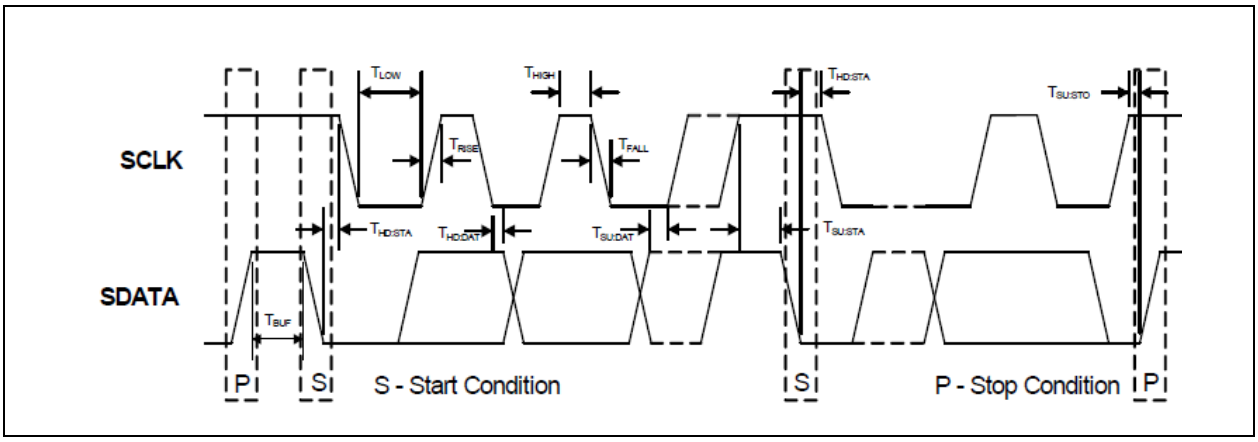


TABLE 8-1: I²C/SMBUS TIMING PARAMETERS

Symbol	Parameter	Standard-Mode		Fast-Mode		Units
		MIN	MAX	MIN	MAX	
T_{BUF}	Bus Free Time	4.7		1.3		µs
$T_{SU;STA}$	START Condition Set-Up Time	4.7		0.6		µs
$T_{HD;STA}$	START Condition Hold Time	4.0		0.6		µs
T_{LOW}	SCL LOW Time	4.7		1.3		µs
T_{HIGH}	SCL HIGH Time	4.0		0.6		µs
T_{RISE}	SCL and SDA Rise Time		1.0		0.3	µs
T_{FALL}	SCL and SDA Fall Time		0.3		0.3	µs
$T_{SU;DAT}$	Data Set-Up Time	0.25		0.1		µs
$T_{HD;DAT}$	Data Hold Time	0		0		µs
$T_{SU;STO}$	STOP Condition Set-Up Time	4.0		0.6		µs

APPENDIX A: DATA SHEET REVISION HISTORY

TABLE A-1: REVISION HISTORY

Revision	Section/Figure/Entry	Correction
DS00002379B (02-01-21)	Section 8.0, "SMBus Interface Timings"	Page header modified; updated Sales Listing and Trademarks pages.
DS00002379A (02-24-17)	Document Release	

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

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

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

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

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

ISBN: 9781522474630

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

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

Austin, TX
Tel: 512-257-3370

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

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

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

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

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

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

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

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

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

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

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

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

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

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

EUROPE

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

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

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

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

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

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

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

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

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