Freescale Semiconductor
Application Note

Document Number: AN4471
Rev. 1, 08/2012

SMBus Quick Start Guide

by: Roger Fan
Field Application Engineer

1 Introduction

The System Management Bus (SMBus) is a two-wire interface through which various system component chips can communicate with each other and with the rest of the system. It is based on the principles of operation of I^2C. SMBus provides a control bus for the system to pass messages to and from devices instead of using individual control lines, helping to reduce pin count and system wires.

With SMBus, a device can:

- Provide manufacturer information
- Tell the system its model/part number
- Save its state for a suspend event
- Report different types of errors
- Accept control parameters
- Return its status

SMBus, first proposed by Intel in 1995, was designed to allow a battery to communicate with the charger, the

Contents

freescale™ semiconductor

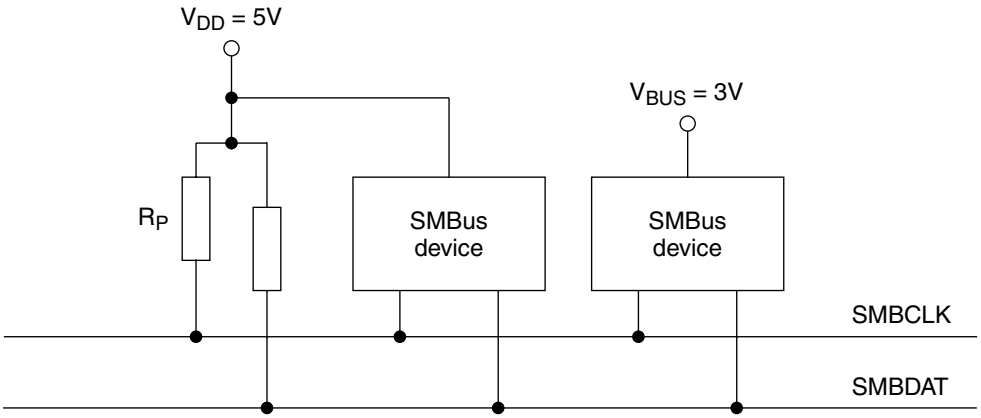
system host, and/or other power-related components in the system. It was developed to enable an inexpensive, yet powerful method for controlling and getting information from devices attached to a notebook motherboard. One of the goals of SMBus was to give digital capabilities to devices based on analog semiconductor technologies, thus creating a hybrid of the two. Many of the commands defined by SMBus are for simple logic implementations. SMBus devices do not need to implement all the commands defined in the SMBus specification. This makes the implementation of a SMBus driver for a SMBus support system much easier.

The current SMBus specification is version 2.0. The earlier version SMBus 1.0 and 1.1 specification was designed primarily with Smart Batteries in mind, though it could be used for other low-power devices. The main difference between the earlier and current versions of the specification is that SMBus 2.0 defines electrical characteristic classes for both low- and high-power devices.

1.1 SMBus topology

SMBus devices in a system may be powered by the system bus V_{DD} or by another power source (for example, the Smart Batteries powered by themselves). The following diagram shows such an implementation.

The 5V V_{DD} is the main power of this system and devices are powered by it. At the same time, there is another device that is powered by a 3V V_{BUS} attached to the SMBus lines. These devices will inter-operate as long as they adhere to the SMBus electrical specifications. An example of this implementation can be found in a system with a Smart Charger, powered by 5V and a Smart Battery, powered by the device itself.



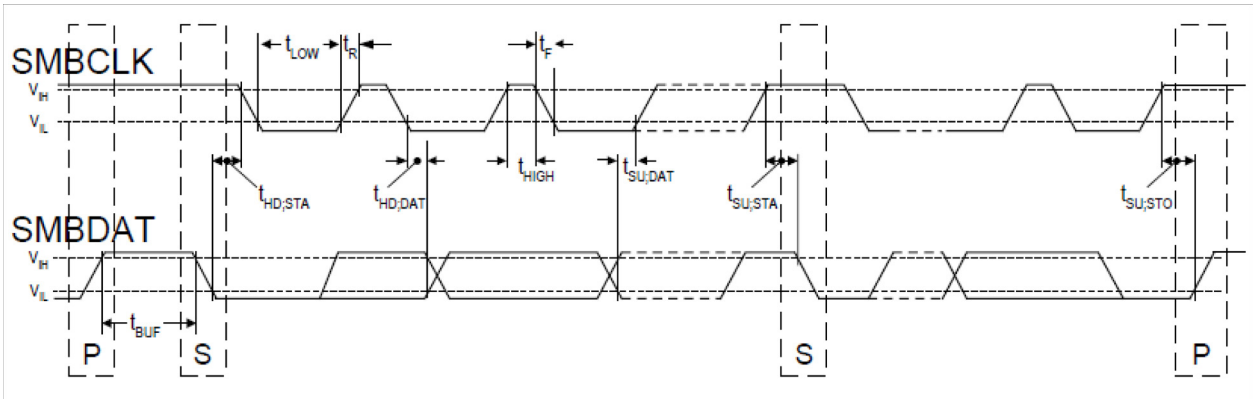
Source: *System Management Bus Specification,* version 2.0, Figure 2-1

Figure 1. SMBus topology

It is generally known that, as with I^2C, devices connected to the bus, the SMBCLK and SMBDAT lines, must have an open drain or open collector in order to perform the wired-AND function. Care should be taken in the design of both the input and output stages of SMBus devices in order not to load the bus when their power is turned off (that is, powered down devices must provide no leakage path to the ground).

2 SMBus electrical specifications

Although the speed of the SMBus is specified from 10 KHz to 100 KHz, but most current implementations are in the range of 50 KHz to 100 KHz. Do not reduce the operating frequency to F_{SMB} minimum even due to periodic clock extensions by slave devices. The device needs to be in the operational state within 500 ms after it is powered on. For a self-powered or always-powered device, this ready-for-operation criteria can be replaced by detecting the active state of SMBus (that is, the clock and data lines have gone high from low for more than 2.5 seconds. Below is the timing diagram of SMBus and its AC and DC specifications.



Source: *System Management Bus Specification,* version 2.0, Figure 3-1

Figure 2. SMBus timing diagram

Table 1. SMBus AC specification [1]

Symbol	Parameter	Limits		Units
		Min	Max	
f_{SMB}	SMBus operating frequency	10	100	KHz
t_{BUF}	Bus free time between Stop and Start condition	4.7	—	µs
$t_{HD:STA}$	Hold time after (repeated) Start condition. After this period, the first clock is generated.	4.0	—	µs
$t_{SU:STA}$	Repeated Start condition setup time	4.7	—	µs
$t_{SU:STO}$	Stop condition setup time	4.0	—	µs
$t_{HD:DAT}$	Data hold time	300	—	ns
$t_{SU:DAT}$	Data setup time	250	—	ns
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
t_{LOW}	Clock low period	4.7	—	µs
t_{HIGH}	Clock high period	4.0	50	µs
$t_{LOW:SEXT}$	Cumulative clock low extend time (slave device)	—	25	ms

Table 1. SMBus AC specification (continued)[1]

Symbol	Parameter	Limits		Units
		Min	Max	
$t_{LOW:MEXT}$	Cumulative clock low extend time (master device)	—	10	ms
t_F	Clock/data fall time	—	300	ns
t_R	Clock/data rise time	—	1000	ns
t_{POR}	Time in which a device must be operational after power-on reset		500	ms

1 Source: *System Management Bus Specification,* version 2.0, Table 1

Table 2. SMBus DC specification[1]

Symbol	Parameter	Limits		Units	Comments
		Min	Max		
V_{IL}	Data, clock input low voltage	—	0.8	V	
V_{IH}	Data, clock input high voltage	2.1	VDD	V	
V_{OL}	Data, clock output low voltage	—	0.4	V	at I_{PULLUP} ,max
I_{LEAK}	Input leakage	—	±5	µA	2
I_{PULLUP}	Current through pullup resistor or current source	100	350	µA	3
V_{DD}	Nominal bus voltage	2.7	5.5	V	3V to 5V ± 10%

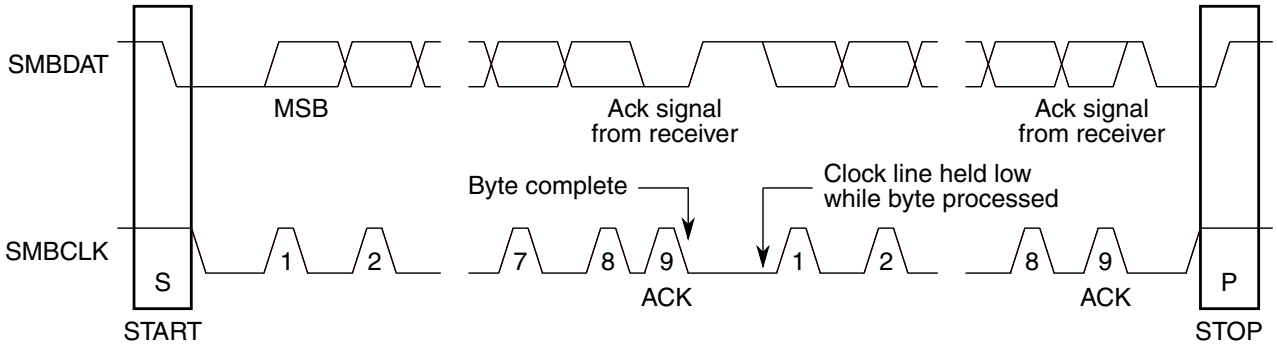
1 Source: *System Management Bus Specification,* version 2.0, Table 2

2 Devices must meet this specification whether powered or unpowered. However, a microcontroller acting as an SMBus host may exceed I_{LEAK} by no more than 10 µA.

3 The I_{PULLUP}, MAX specification is determined primarily by the need to accommodate a maximum of 1.1K equivalent series resistor of removable SMBus devices, such as the Smart Battery, while maintaining the V_{OL}, MAX of the bus.

3 Data transfers on SMBus

The SMBus uses fixed voltage levels to define the logic 0 (max 0.8V) and logic 1 (min 2.1V) on the bus, respectively. The data that appears on the SMBDAT line must be stable during the “high” period of the clock, and the data can only change state in the “low” period of the clock.



Source: *System Management Bus Specification,* version 2.0, Figure 3-1

Figure 3. SMBus data transfer format

The SMBus uses the ACK signal to detect the presence of detachable devices on the bus, so a device must always ACK its own address when the host accesses it. For other data bytes, the device can select ACK or NACK when receiving them.

4 SMBus usage model

The SMBus specification refers to three types of devices: *host*, *master*, and *slave*.

- A host is a specialized master that provides the main interface to the system's CPU.
- A master is a device that issues commands, generates the clocks, and terminates the transfer.
- A slave is a device that receives or responds to a command.

A system may not include a host. For example, a simple battery charging system is a hostless system. In an SMBus system, a device can be master only, slave only, or it may act as a slave most of the time, but in special instances it becomes a master.

Table 3. Reserved SMBus addresses [1]

Slave address Bits 7–1	R/W# bit Bit 0	Comment
0000 000	0	General call address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future use
0000 1XX	X	Reserved for future use
0101 000	X	Reserved for ACCESS.bus host
0110 111	X	Reserved for ACCESS.bus default address
1111 0XX	X	10-bit slave addressing

Table 3. Reserved SMBus addresses (continued)[1]

Slave address Bits 7–1	R/W# bit Bit 0	Comment
1111 1XX	X	Reserved for future use
0001 000	X	SMBus host
0001 100	X	SMBus alert response address
1100 001	X	SMBus device default address

[1] Source: *System Management Bus Specification,* version 2.0, Table 4

Each device that exists as a slave on the SMBus has one unique seven bit address called the slave address. Each address is seven bits long with a read/write bit appended in bit position 0. When a device "sees" its address, it wakes up and responds to the rest of the command. Besides the General Call Address, SMBus systems can have 127 devices. SMBus version 2.0 introduces the concept of dynamically assigned addresses, and the SMBus Device Default Address is reserved for this purpose. A process called SMBus Address Resolution Protocol (ARP) uses this address. When the host detects two devices with the same slave address, the ARP process resolves the slave address conflict by dynamically assigning a new unique address to slaves. For reference, some addresses in the table below are reserved and must not be used or assigned to any SMBus slave device unless otherwise detailed by the SMBus specification. The following table lists the current assigned device addresses.

Table 4. Assigned SMBus addresses [1]

Slave Address	Description	Specification
0001 000	SMBus Host	*System Management Bus Specification*, version 1.1 December 1998
0001 001	Smart Battery Charger	*Smart Battery Charger Specification*, version 1.1 December 1998
0001 010	Smart Battery Selector Smart Battery System Manager	*Smart Battery Selector Specification*, version 1.1 December 1998 *Smart Battery System Manager Specification*, version 1.0B August 1999
0001 011	Smart Battery	*Smart Battery Data Specification*, version 1.1 December 1998
0001 100	SMBus Alert Response	*System Management Bus Specification*, version 1.1 December 1998
0101 000	ACCESS.bus host	
0101 100	Reserved by previous versions of the SMBus specification for LCD Contrast Controller. This address may be reassigned in future versions of the SMBus specification.	
0101 101	Reserved by previous versions of the SMBus specification for CCFL Backlight Driver. This address may be reassigned in future versions of the SMBus specification.	
0110 111	ACCESS.bus default address	

Table 4. Assigned SMBus addresses (continued)[1]

Slave Address	Description	Specification
1000 0XX	Reserved by previous versions of the SMBus specification for PCMCIA Socket Controllers (four addresses). These addresses may be reassigned in future versions of the SMBus specification.	
1000 100	Reserved by previous versions of the SMBus specification for (VGA) Graphics Controller. This address may be reassigned in future versions of the SMBus specification.	
1001 0XX	Unrestricted addresses	*System Management Bus Specification*, version 1.1 December 1998
1100 001	SMBus Device Default Address	*System Management Bus Specification*, version 1.1 December 1998

1 Source: *System Management Bus Specification,* version 2.0, Table 11

5 Using an SMBus device

A typical SMBus device will have a set of commands by which data can be read and written. All commands are one byte long while their arguments and return values can vary in length. Accessing a command that does not exist or is not supported may cause an error condition. In accordance with the SMBus specification, the most significant bit (MSB) is transferred first. There are eleven possible command protocols for any given device. These commands are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write, and Block Write-Block Read Process Call. A slave device may use any or all of the eleven protocols to communicate.

Many of the commands defined by SMBus are geared for simple logic implementations; therefore, SMBus devices do not need to implement all of the commands defined in the SMBus specification. They can implement only those commands that they need for their simple system. For example, SMBus devices like Smart Batteries, Smart Battery Chargers, and Selectors may use only Read/Write Word and Block Read commands. This makes implementing a SMBus driver for these devices much easier. One interesting thing is that Smart Batteries are free to use any SMBus commands such as Block Write, to implement custom functions, but it is not required by the Smart Battery Data specification.

The Block Write command protocol is typically used during the manufacturing process to initialize the battery data (for example, manufacturing date, serial number, electronics trimming, scaling values, and so on).

6 Packet error checking

SMBus version 1.1 introduced a Packet Error Checking mechanism to improve communication reliability and robustness. Implementation of Packet Error Checking is optional for SMBus devices, but it is required for devices participating in the ARP process. Devices that implement Packet Error Checking must be capable of communicating with the host and other devices that do not implement the Packet Error Checking mechanism. Packet Error Checking, whenever applicable, is implemented by appending a

Packet Error Code (PEC) at the end of each message transfer. Most command protocol have two variants: one with the PEC byte and one without. The PEC is a CRC-8 error-checking byte, which is calculated on all the message bytes.

7 Bus protocols

All the commands first put a start condition on the bus, then begin the transmission by transmitting the command/data, wait for an acknowledge from the slave (receiving) device during command/data transmission, and then put a stop condition on the bus.

Following is a description of some basic SMBus command protocols with and without a Packet Error Code. In each figure below, *S* is the START bit, *A* is the ACK/NACK bit, and *P* is the STOP bit. Detailed descriptions of these command protocols can be found in the SMBus specification.

Quick command—The R/W bit in the slave address denotes the commands. An example of using it is to turn it on/off, or to enable/disable a device function. There is no data sent or received.

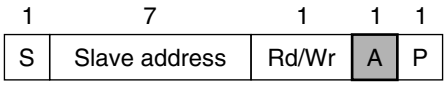


Figure 4. Quick command protocol

Send byte—The data byte sent contains the features to be accessed and actions for this particular feature to execute.

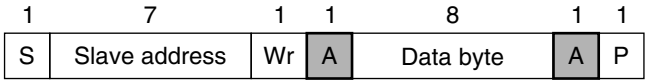


Figure 5. Send byte protocol

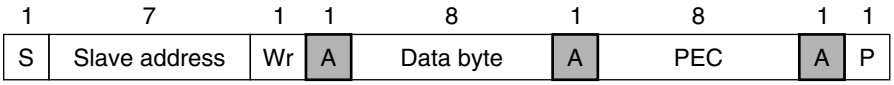


Figure 6. Send byte protocol with PEC

Receive byte—The Receive byte is similar to a Send byte; the only difference is the direction of data transfer. The Receive byte is used where the host accesses the device for some information.

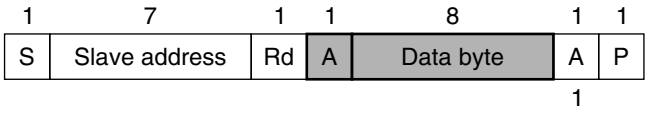


Figure 7. Receive byte protocol

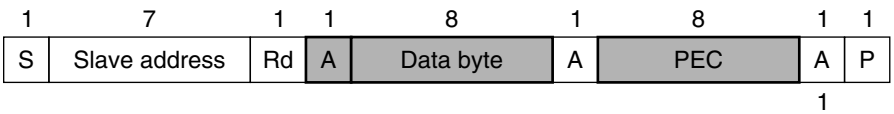


Figure 8. Receive byte protocol with PEC

Write byte/word—The first byte of a Write byte/word access is the command code. The next one or two bytes are the data to be written.

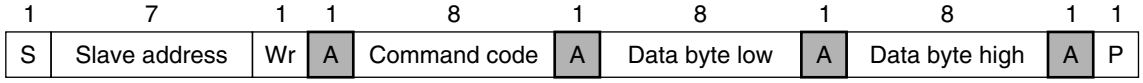


Figure 9. Write word protocol

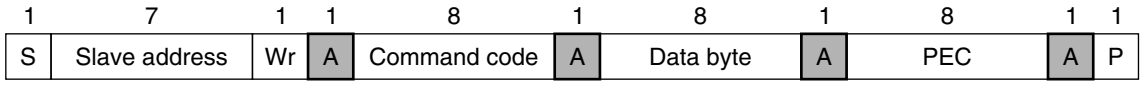


Figure 10. Write byte protocol with PEC

Read byte/word—The host first writes a command to the slave device, then immediately follows that command with a repeated START condition to denote a read from that device's address. The slave then returns one or two bytes of data.

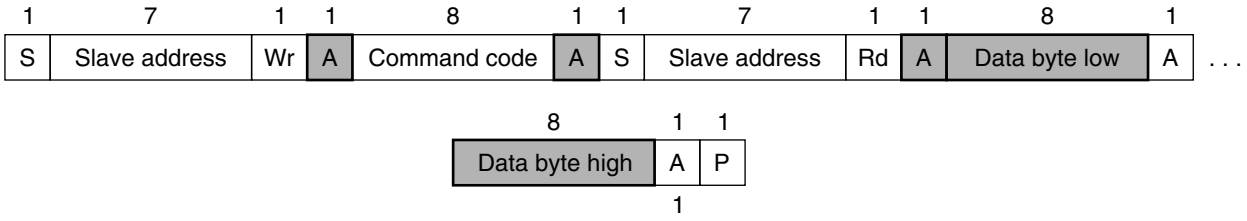


Figure 11. Read word protocol

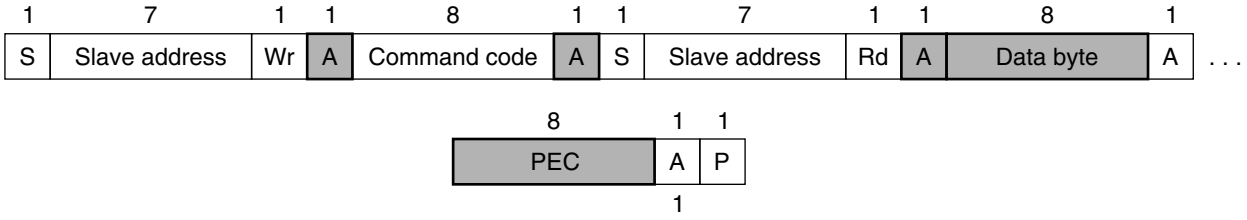


Figure 12. Read byte protocol with PEC

Block Read or **Block Write**—Block Read or Write begins with a slave address and then a R/W condition. After the command code, the host issues a data byte which describes how many more bytes will follow in the message. Block Read differs from Block Write in that a repeat START exists for the requirement of transfer direction change.

For a Smart Battery application, Block Write is typically used during the manufacturing process to initialize the battery data (for example, manufacturing date, serial number, electronics trimming, scaling values, and so on).

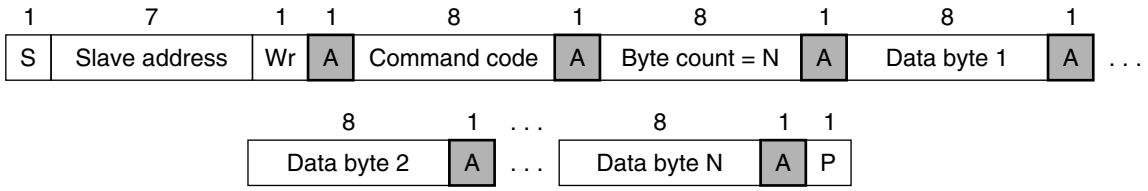


Figure 13. Block read protocol

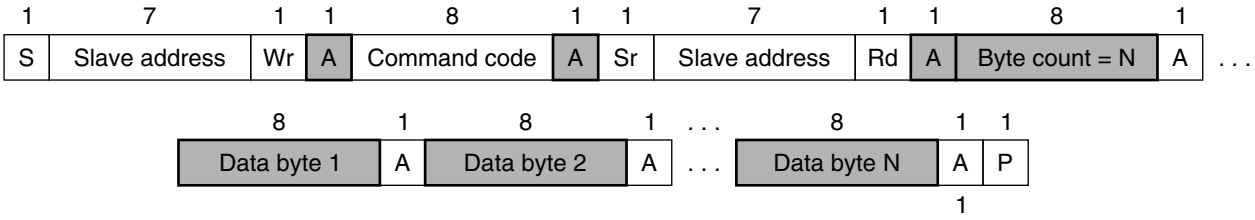


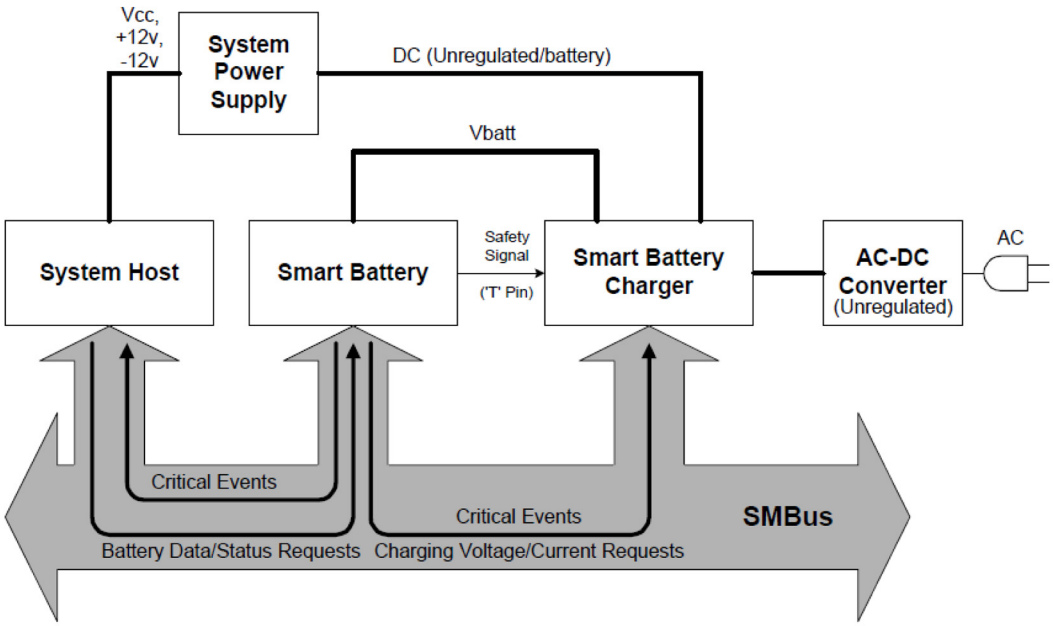
Figure 14. Block write protocol

8 Where SMBus is used

The Smart Battery System (SBS) is the system that uses SMBus technology. An SBS system usually consists of a Host, a Smart Charger, and a Smart Battery. It is the easiest and most efficient way to implement a battery management system for portable electronic devices such as laptop computers, cellular phones, video cameras, and so on.

The SBS design uses SMBus technology; therefore, the Smart Battery and Smart Charger can communicate with each other and with the rest of the system. In the SBS, the battery can tell the charger its battery chemistry type, its capacity, how it wants to be charged, its optimal charging current, maximum charging time, and so on. The battery, if it is smart, can be operated optimally. It doesn't have to be recharged as frequently as those using systems that only "guess" how much power is available. This benefits the end user by providing longer battery life and accurate battery energy capacity information down to the last 1%.

In fact, an SBS-compliant design can increase battery cycle life up to 30%. In addition, users are not tied to one type of battery technology; different types of battery chemistries can be used interchangeably. Smart Chargers can choose the battery chemistry they support, and use the correct charging profile for that chemistry. This reduces potential damage and avoids risk that may be caused by an non-smart charger.



Source: *Smart Battery Data Specification,* Revision 1.1, p. 4

Figure 15. Typical Smart Battery/Charger system using SMBus

The previous diagram is a typical Smart Battery system. It shows data/commands such as critical events, charging currents, charging requests, and so on, that flow across the SMBus between the Smart Battery, SMBus Host, Smart Battery Charger, and other devices.

Some data or commands supported by Smart Battery and Smart Charger are listed below.

For Smart Battery:

- *ManufacturerAccess() (0x00)*
- *RemainingCapacityAlarm() (0x01)*
- *RemainingTimeAlarm() (0x02)*
- *BatteryMode() (0x03)*
- *AtRate() (0x04)*
- *AtRateTimeToFull() (0x05)*
- *AtRateTimeToEmpty() (0x06)*
- *Temperature() (0x08)*
- *Voltage() (0x09)*
- *Current() (0x0a)*
- *AverageCurrent() (0x0b)*
- *MaxError() (0x0c)*
- *RelativeStateOfCharge() (0x0d)*

For Smart Charger:

- *ChargingCurrent() (0x14)*
- *ChargingVoltage() (0x15)*
- *AlarmWarning() (0x16)*
- *ChargerMode() (0x12)*
- *ChargerStatus() (0x13)*
- *ChargerSpecInfo() (0x11)*

A example of one of these commands—AverageCurrent—excerpted from the *Smart Battery Data Specification*, revision 1.1 is shown below. 0x0b is the command code and Read Word is the SMBus protocol being used.

AverageCurrent() **(0x0b)**

Description:
Returns a one-minute rolling average based on the current being supplied (or accepted) through the battery's terminals (mA). The AverageCurrent() function is expected to return meaningful values during the battery's first minute of operation.

Purpose:
The AverageCurrent() function provides the average current flowing into or out of the battery for the power management system.

SMBus Protocol: Read Word

Output: signed int -- charge/discharge rate in mA increments - positive for charge, negative for discharge

Units:	mA
Range:	0 to 32,767 mA for charge or 0 to -32,768 mA for discharge
Granularity:	0.2% of the DesignCapacity() or better
Accuracy:	±1.0% of the DesignCapacity()

A more detailed description of the data can be found in the *Smart Battery Data Specification* and *Smart Battery Charger Specification*.

9 Differences between SMBus and I2C

SMBus is derived from I2C, but there are several major differences between the specifications of the two buses in the areas of timing, protocols, operation modes, and electrical characteristics.

Timing:

- SMBus defines a minimum (10 KHz) and maximum (100 KHz) bus clock frequency, while I2C does not specify minimum bus frequency. I2C provides 100 KHz for Standard mode and 400 KHz for Fast mode.
- SMBus defines clock low time out, cumulative clock low extend time for slave and master, rise and fall time of bus signals, and so on while I2C does not.
- SMBus defines data hold time to 300 ns, while I2C defines it as zero.

ACK and NACK usage:

- I^2C devices allow the slave not to ACK its slave address, but SMBus requires it to always ACK it as a mechanism to detect a detachable device's presence on the bus (battery, docking station, and so on).
- I^2C uses NACK to indicate that it cannot receive any more data bytes after it acknowledges its slave address. SMBus uses NACK to indicate the reception of an invalid command or data.

Protocol:

- SMBus specifies the protocols that a device is allowed to use when communicating with other SMBus devices on the bus.

Electrical:

The following table lists the main DC parameter differences between I^2C and SMBus.

Table 5. DC parameter comparison between standard I^2C, fast I^2C, and SMBus devices[1]

Symbol	Parameter	Std. I^2C mode device		Fast I^2C mode device		SMBUs device		Units
		Min	Max	Min	Max	Min	Max	
V_{IL}	Fixed input level	–0.5	1.5	–0.5	1.5	—	0.8	V
	V_{DD} related input level	–0.5	$0.3V_{DD}$	–0.5	$0.3V_{DD}$	N/A	N/A	V
V_{IH}	Fixed input level	3.0	V_{DD}max + 0.5	3.0	V_{DD}max + 0.5	2.1	5.5	V
	V_{DD} related input level	$0.7V_{DD}$	V_{DD}max + 0.5	$0.7V_{DD}$	V_{DD}max + 0.5	N/A	N/A	V
V_{HYS}	V_{IH}–V_{IL}	N/A	N/A	$0.05V_{DD}$	—	N/A	N/A	V
V_{OL}	V_{OL} @ 3mA	0	0.4	0	0.4	N/A	N/A	V
	V_{OL} @ 6mA	N/A	N/A	0	0.6	N/A	N/A	V
	V_{OL} @ 350µA	N/A	N/A	N/A	N/A	—	0.4	V
I_{PULLUP}		N/A	N/A	N/A	N/A	100	350	µA
I_{LEAK}		–10	10	–10	10	–5	5	µA

[1] Source: *System Management Bus Specification*, version 2.0, Table 10

10 MC9S08MP12/16 introduction

The MC9S08MP16 and MC9S08MP12 are members of the low-cost, high-performance HCS08 family of 8-bit microcontrollers (MCUs) from Freescale. All MCUs in the family use the enhanced HCS08 core and are available with a variety of peripheral modules, memory sizes, memory types, and package types. MC9S08MP16 supports SMBus version 2.0 with the help of the I^2C peripheral modules. The following table summarizes the feature set available in the MC9S08MP16 series MCUs.

Table 6. MC9S08MP16 series feature set

Feature	MC9S08MP16			MC9S08MP12
Flash size (bytes)	16384			12288
RAM size (bytes)	1024			512
Pin quantity	48	32	28	
ADC channels	13	12	8	
CRC	yes			
DAC	3			
DBG	yes			
FTM1 channels	2			
FTM2 channels	6			
HSCMP	3			
ICS	yes			
IIC	yes			no
MTIM	yes			
KBI pins	24	15	14	
PDB	2			
PGA	yes			no
Pin I/O[1]	40	25	22	
RTC	yes			
SCI	yes			
SPI	yes			
XOSC	yes			

[1] Port I/O count does not include the output-only pins PTF0/BKGD/MS and PTF1/$\overline{\text{RESET}}$.

11 References

- *MC9S08MP16 Series Data Sheet*
- *System Management Bus Specification*, Revision 2.0, SBS-Implementers Forum, August, 2000
- *Smart Battery Charger Specification*, Revision 1.1, SBS-Implementers Forum, December, 1998
- *Smart Battery Data Specification*, Revision 1.1, SBS-Implementers Forum, December, 1998
- *The IC-bus and How to Use It*, Philips Semiconductors document #98-8080-575-01

12 Summary

This technical guide serves as a quick guide for people who are new to the System Management Bus (SMBus). It describes SMBus history, its DC and AC electrical characteristic, command protocols, and key differences between SMBus and I^2C. This document also introduces the MC9S08MP16, which supports SMBus 2.0.

13 Revision history

Revision	Description of changes
0	Initial version
1	• Section 10, “MC9S08MP12/16 introduction”: Removed reference to PEC and CRC in the following sentence: “MC9S08MP16 supports SMBus version 2.0 with PEC through the on-chip CRC and the I^2C peripheral modules.”

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4471
Rev. 1
08/2012