Smart Battery System Specifications

System Management Bus Specification

Revision 1.0
February 15, 1995

Please forward questions and comments regarding this specification to:

Intel Corporation
Phone Support (8am-5pm PST) 1-800-628-8686
FAX Support 1-916-356-6100
International Phone Support 1-916-356-3551
Email IAL_Support@ccm.hf.intel.com

TABLE OF CONTENTS

Revision History

Revision Number	Date	Notes
1.0	2/15/95	General Release

(This page left intentionally blank.)

1.Overview

1.1. What is System Management Bus?

The System Management Bus (SMB) is a two-wire interface through which simple power-related chips can communicate with the rest of the system. It uses I²C as its backbone.

A system using SMB passes messages to and from devices instead of tripping individual control lines. Removing the individual control lines reduces pin count. Accepting messages ensures future expandability.

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

The System Management Bus may share the same host device and physical bus as ACCESS.bus components provided that an appropriate electrical bridge is provided between the internal SMB devices and external ACCESS.bus devices.

1.2. Audience

The target audience for this document includes:

- System designers implementing the System Management Bus Specification in their systems
- VLSI engineers designing chips to connect to the System Management Bus
- Software engineers writing support code for System Management Bus chips

1.3. Scope

This document describes the communications protocols available for use by devices on SMB. Its original purpose was to define the communication link between an intelligent battery, a charger for the battery, and a microcontroller that communicates with the rest of the system. However, it can also be used to connect a wide variety of power-related devices.

The specification allows for multiple devices to attach to the System Management Bus through standard slave addresses. Information is exchanged through a simple index set specific to each device.

The SMBCLK and SMBDATA pins are similar to the clock and data pins found on an I²C bus. The SMB electrical characteristics differ from those of I²C.

1.4. Supporting Documents

This specification assumes that the reader is familiar with or has access to the following documents:

- *The I²C-bus and how to use it*, Philips Semiconductors document #98-8080-575-01.
- *ACCESS.bus Specifications* -- *Version 2.2*, ACCESS.bus Industry Group, 370 Altair Way Suite 215, Sunnyvale, CA 94086 Tel (408) 991-3517

1.5. Main Differences Between System Management Bus and I²C

The major differences between SMB and I²C fall into several categories including electrical, timing, protocols, and operating modes.

- SMB is based on fixed voltage levels, the I²C levels are scaleable. However, the SMB logic levels are easily met using standard 5 volt components.
- SMB specifies a minimum operational clock speed.
- SMB specifies device timeouts.
- SMB allows a slave device to stretch the cumulative clock (low) time, in a single message, up to $T_{LOW:SEXT}$. This allows, for example, a low-power microprocessor-based slave device, such as a Smart Battery, sufficient time to "wake-up" and/or "marshal data."
- SMB allows a master device to stretch the cumulative clock (low) time, in any single byte, up to $T_{LOW:MEXT}$. This allows, for example, a keyboard controller-based SMB emulation sufficient time to service keyboard interrupts while hosting the SMBus.
- SMB specifies the protocol that an SMB device is allowed to use when communicating with the SMB Host operating as a slave device.

1.6. Main Differences Between System Management Bus and ACCESS.bus

The major differences between ACCESS.bus and SMB also fall into several categories including electrical, protocols and operating modes.

- SMB has fixed voltage levels, ACCESS.bus uses .3 and .7 V_{CC} (presently defined at 5 volts) for logic levels.
- SMB does not specify a maximum bus capacitance.
- SMB specifies a maximum sink current I_{PULLUP} (350 µa), which is considerably less than the 6 ma specified by ACCESS.bus.
- SMB specifies a maximum V_{OL} (0.4 volts) less than the 0.6 volts specified by ACCESS.bus.
- SMB specifies fixed addresses for SMB devices as opposed to the assignable addressing scheme specified by ACCESS.bus. However, there is a reserved SMB address which is intended for use by future SMB devices that may offer a limited form of assignable addressing.
- SMB requires SMB devices to respond directly as opposed to ACCESS.bus that requires a device to respond independently to a request within 40 ms. All SMB devices are required to reset themselves in such a manner as to return the SMBus to an idle state whenever any SMB device does not respond within $T_{TIMEOUT}$ ms.
- SMB uses both the read and write modes of I²C. ACCESS.bus uses only the write mode.
- SMB does not specify a connector.

2. Electrical Characteristics

SMB voltages depart from the original I²C specification in order to allow the same chips to work on a future SMB with a much lower operating voltage.

The protocol deviates from the original I²C electrical characteristics in the following ways:

2.1. AC Specifications

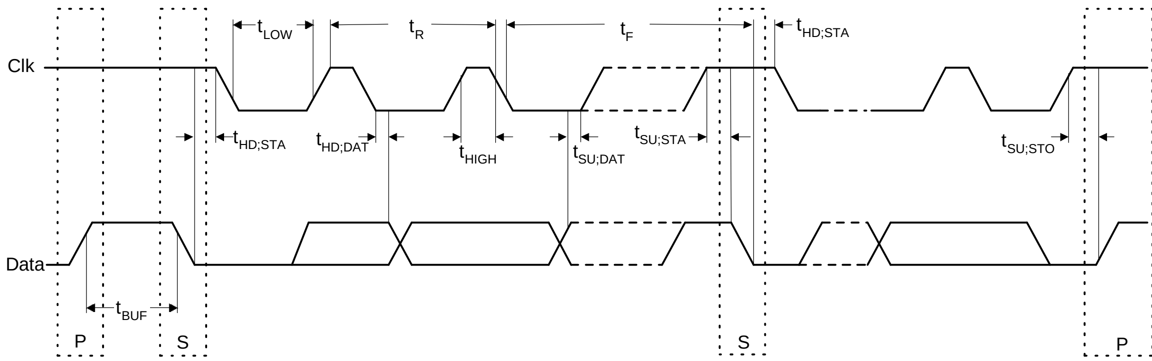
Symbol	Parameter	Limits		Units	Comments
		Min	Max		
F_{SMB}	SMB Operating Frequency	10	100	KHz	
T_{BUF}	Bus free time between Stop and Start Condition	4.7		µs	
$T_{HD:STA}$	Hold time after (Repeated) Start Condition. After this period, the first clock is generated.	4.0		µs	
$T_{SU:STA}$	Repeated Start 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	
$T_{TIMEOUT}$		25	35	ms	see note 1
T_{LOW}	Clock low period	4.7		µs	
T_{HIGH}	Clock high period	4.0	50	µs	see note 2
$T_{LOW:SEXT}$	Cumulative clock low extend time (slave device)		25	ms	see note 3
$T_{LOW:MEXT}$	Cumulative clock low extend time (master device)		10	ms	see note 4
T_F	Clock/Data Fall Time		300	ns	
T_R	Clock/Data Rise Time		1000	ns	

Note 1: A device will timeout when any clock low exceeds this value.

Note 2: T_{HIGH} Max provides a simple guaranteed method for devices to detect bus idle conditions.

Note 3: $T_{LOW:SEXT}$ is the cumulative time a slave device is allowed to extend the clock cycles in one message from the initial start to the stop. If a slave device exceeds this time, it is expected to release both its clock and data lines and reset itself.

Note 4: $T_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from start-to-ack, ack-to-ack, or ack-to-stop.



Timing Measurements

2.1.1. General timing conditions

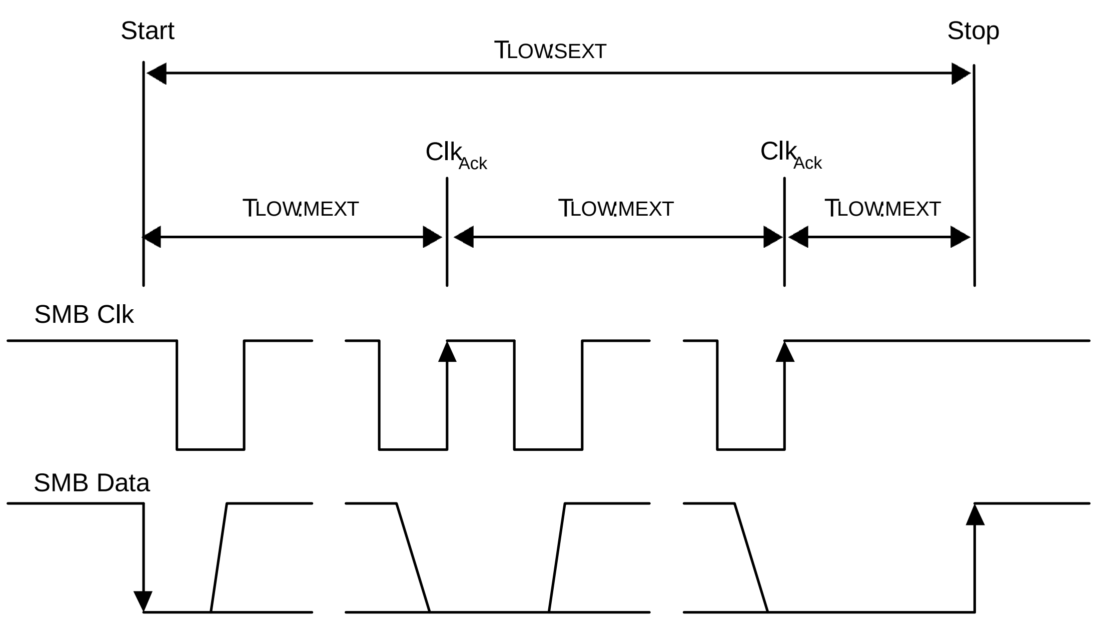
The SMBus is designed to provide a predictable communications link between a system and its devices. However some devices, such as a Smart Battery using a microcontroller to support both bus and maintain battery data, may require more time than might normally be expected. These specifications take such devices into account while maintaining a relatively predictable communications. The following are general comments on the SMBus' timing:

- The bus may be at 0 KHz when idle.
- The FSMB Min is intended to dissuade components from taking too long to complete a transaction.
- An idle bus can be detected by observing that both the clock and data remain high for longer than THIGH Max.
- Every device must be able to recognize and react to a start condition at FSMB Max.

2.1.2. Timeouts

The following diagram illustrates the definition of the timeout intervals, TLOW:SEXT and TLOW:MEXT.

TTIMEOUT Measurement Intervals



2.1.3. Slave device timeout definitions and conditions

A slave device must always timeout when any clock is held low longer than $T_{TIMEOUT}$ maximum.

2.1.4. Master device timeout definitions and conditions

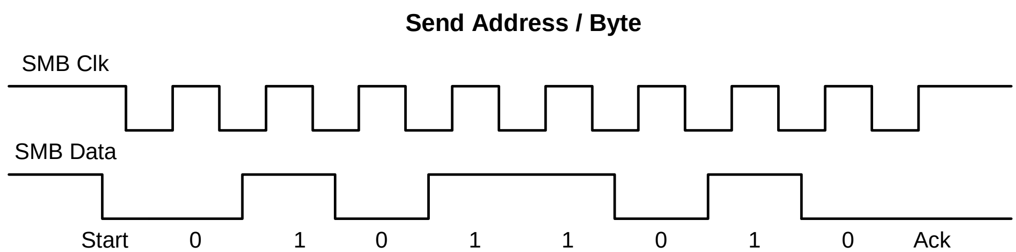
$T_{LOW:MEXT}$ is defined as the cumulative time a master device is allowed to extend its clock cycles within one byte in a message as measured from:

- start to ack
- ack to ack
- ack to stop.

A system host may not violate $T_{LOW:MEXT}$ except while forcing a slave device timeout.

2.1.5. Sample transaction diagram

This drawing illustrates a data transactions on the SMBus.



2.2. DC Specifications

2.2.1. Parameters

The System Management Bus is designed to operate over a wide range of voltages. The following electrical specifications ensure that chips designed today for 5.0 V systems will still work on future bus implementations that operate at 2.0 V or even lower.

Symbol	Parameter	Limits		Units	Comments
		Min	Max		
V_{IL}	Data, Clock Input Low Voltage	-0.5	0.6	V	
V_{IH}	Data, Clock Input High Voltage	1.4	5.5	V	
V_{OL}	Data, Clock Output Low Voltage		0.4	V	@ $I_{PULLUP\ MIN}$
I_{LEAK}	Input Leakage		±1	µA	
I_{PULLUP}	Current through pullup resistor or current source	100	350	µA	

In cases where a microcontroller is used as the SMBus host, the parameter I_{LEAK} may be exceeded. However, because of the relatively low pullup current, the system designer must ensure that the loading on the bus remains within acceptable limits. Additionally, to prevent bus loading, any components that remain connected to the active bus while unpowered (that is, their V_{CC} lowered to zero), MUST also meet the leakage current specification while unpowered.

Systems can be designed today using CMOS components, such as microcontrollers. It is the responsibility of the system designer to ensure that all SMB components comply with the SMB timing requirements, and are able to operate within the voltage requirements of the specific system.

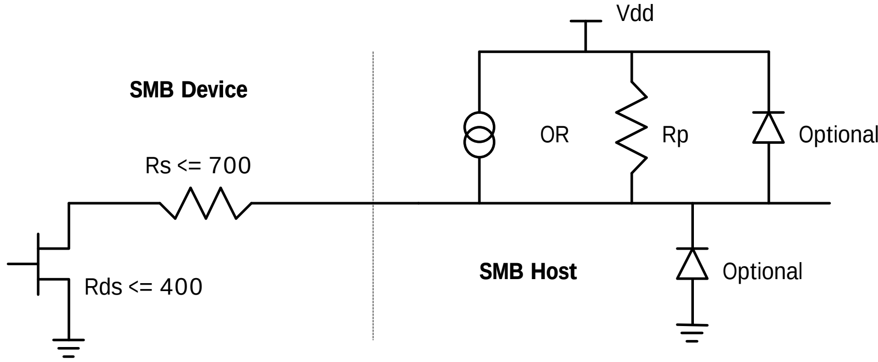
The I²C bus references its electrical characteristics to V_{DD}. Components attached to SMB may operate at different voltages. Therefore the SMB cannot assume that all devices will share a common V_{DD}, hence fixed voltage logic levels.

Although the SMB uses fixed voltages for its logic levels, Fall Time is a function of the actual V_{DD} used by the system. Rise and fall times are calculated as follows:

- RiseTime = (V_{ILMAX} - 0.15) to (V_{IHMIN} + 0.15)
- FallTime = 0.9V_{DD} to (V_{ILMAX} - 0.15)

2.2.2. Circuit model

The following diagram shows the electrical model of the SMB.



SMB Circuit Model

The value of the pullup resistors (Rp) will vary depending on the system's VDD and the bus's actual capacitance. Current sources offer best performance but with increased cost.

The optional diodes, shown in the diagram above, are for ESD protection. They may be necessary in systems where removable SMB devices such as the Smart Battery are used.

3. Protocol

3.1. Usage Model

The System Management Bus Specification refers to three types of devices. A *slave* is a device that is receiving or responding to a command. A *master* is a device that issues commands, generates the clocks, and terminates the transfer. A *host* is a specialized master that provides the main interface to the system's CPU. There may be at most one host in a system. One example of a hostless system is a simple battery charging station. The station might sit plugged into a wall waiting to charge a smart battery.

A device may designed so that it is never a master, only a slave. A device may act as a slave most of the time, but in special instances it may become a master. It can also work the other way around as in the case of the host, where a device is mostly a master, but in special cases it might become a slave.

3.2. Device Identification -- Slave Address

Each device that uses the System Management Bus has a unique address called the *Slave Address*. Masters and the host have a slave address for those instances when another master wants to talk with them. For reference, the following Slave Addresses are reserved by the I²C specification and thus cannot be used by any of the devices on this particular interface:

Slave Address Bits 7-1	R/W bit Bit 0	Description
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
1111 0XX	X	10-bit slave addressing
1111 1XX	X	Reserved for future use

In addition to the above reserved addresses, the following addresses are reserved for the System Management Bus.

Slave Address	Description
0001 000	SMB Host
0001 100	SMB Alert Response Address
1100 001	SMB Device Default Address
0101 000	reserved for ACCESS.bus host
0110 111	reserved for ACCESS.bus default address
1001 0XX	Unrestricted Addresses

All other addresses are reserved for formal assignment by the SMBus address coordinating committee.

The SMB Alert Response Address (0001100) can be a substitute for device master capability. See Appendix A for details.

The SMB Device Default Address is reserved for future use by SMB devices which may allow assignable addresses.

Unrestricted addresses (10010XX) are up for grabs. They are not intended for production parts and will never be assigned to any device. They are provided for prototyping and experimenting.

Addresses not specified here or within the appendices are reserved for future use. All 10-bit slave addresses are reserved for future use. The host should be able to support access to 10-bit devices.

The host has the lowest address so that emergency messages going to the host have the highest priority. Emergency messages may carry the I²C General Call address if they pertain to more than one device.

3.3. Using a Device

A smart SMB device will have a set of commands by which data can be read and written. All commands are 8 bits (1 byte) long. Command arguments and return values can vary in length. Accessing a command that does not exist or is not supported provokes an error condition. In accordance with the I²C specification, the Most Significant Bit is transferred first.

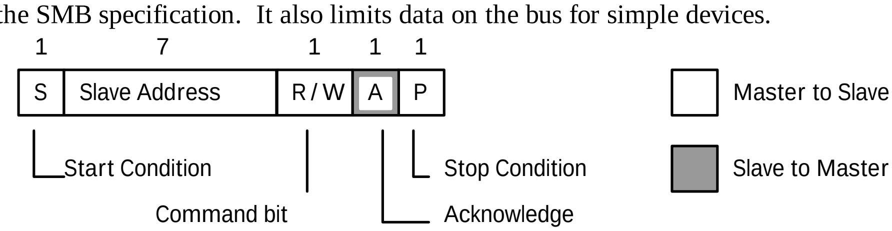
There are eight possible command protocols for any given device. A slave device may use any or all of the eight protocols to communicate. The host device should be able to support all command protocols. The modes are Quick Command, Send Byte, Receive Byte, Write Byte/Word, Read Byte/Word, Process, Block Read, and Block Write.

Commands may be thought of as register accesses.

3.3.1. Quick Command

Here, part of the slave address denotes the command -- the R/W bit. The R/W bit may be used to simply turn a device function on or off, or enable/disable a low-power Standby mode. There is no data sent or received.

The quick command implementation is good for very small devices that have limited support for the SMB specification. It also limits data on the bus for simple devices.

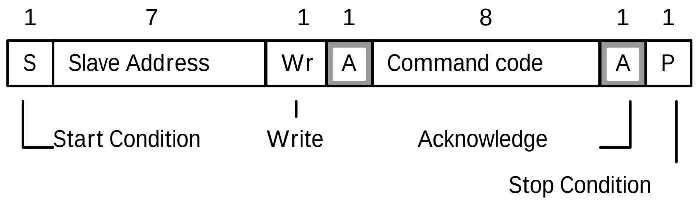


Quick Command Protocol

3.3.2. Send Byte

A simple device may recognize its own slave address and accept up to 256 possible encoded commands in the form of a byte that follows the slave address.

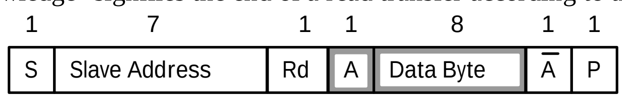
All or parts of the Send Byte may contribute to the command. For example, the highest 7 bits of the command code might specify an access to a feature, while the least significant bit would tell the device to turn the feature on or off. Or, a device may set the "volume" of its output based on the value it received from the Send Byte protocol.



Send Byte Protocol

3.3.3. Receive Byte

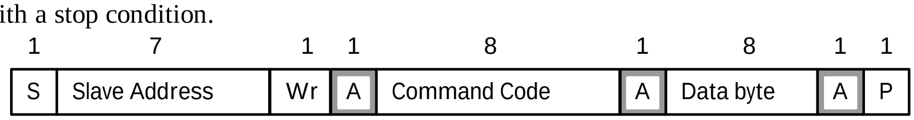
The Receive Byte is similar to a Send Byte, the only difference being the direction of data transfer. A simple device may have information that the host needs. It can do so with the Receive Byte protocol. The same device may accept both Send Byte and Receive Byte protocols. A "Not ACKnowledge" signifies the end of a read transfer according to the I²C specification.



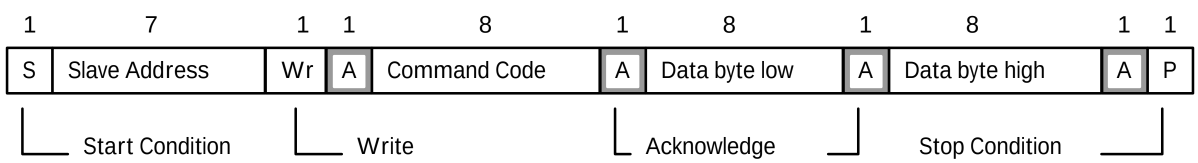
Receive Byte Protocol

3.3.4. Write Byte/Word

The first byte of a Write Byte/Word access is the command code. The next 1 or 2 bytes are the data to be written. In this example the master asserts the slave device address followed by the write bit. The device acknowledges and the master delivers the command code. The slave again acknowledges before the master sends the data byte or word (low byte first). The slave acknowledges each byte according to the I²C specification, and the entire transaction is finished with a stop condition.



Write Byte Protocol

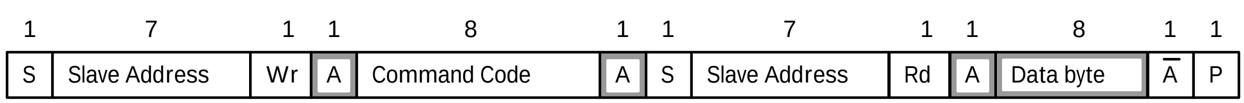


Write Word Protocol

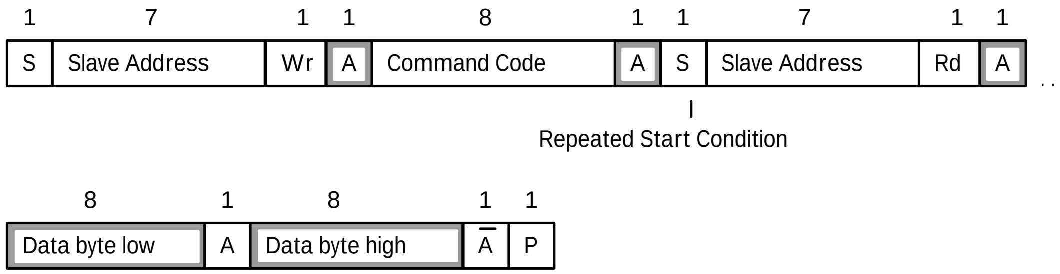
3.3.5. Read Byte/Word

Reading data is slightly more complicated than writing data. First the host must write a command to the slave device. Then it must follow that command with a repeated start condition to denote a read from that device's address. The slave then returns 1 or 2 bytes of data.

Note that there is no stop condition before the repeated start condition, and that a "Not ACKnowledge" signifies the end of the read transfer.



Read Byte Protocol

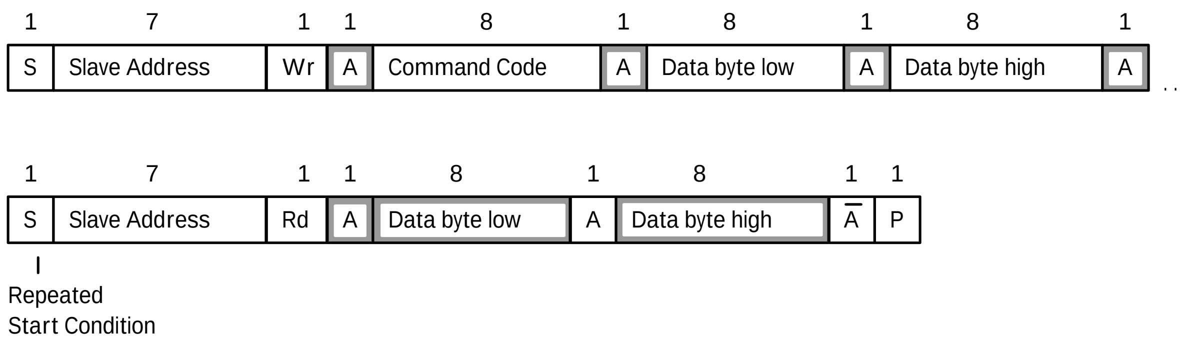


Read Word Protocol

3.3.6. Process Call

The process call is so named because a command sends data and waits for the slave to return a value dependent on that data. The protocol is simply a Write Word followed by a Read Word, but without a second command or stop condition.

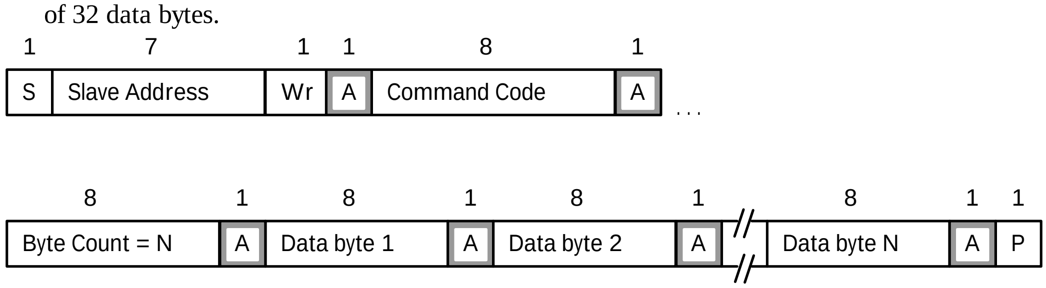
The slave can perform any calculations or lookups during the time it takes to transmit the repeated start condition and slave address.



Process Call

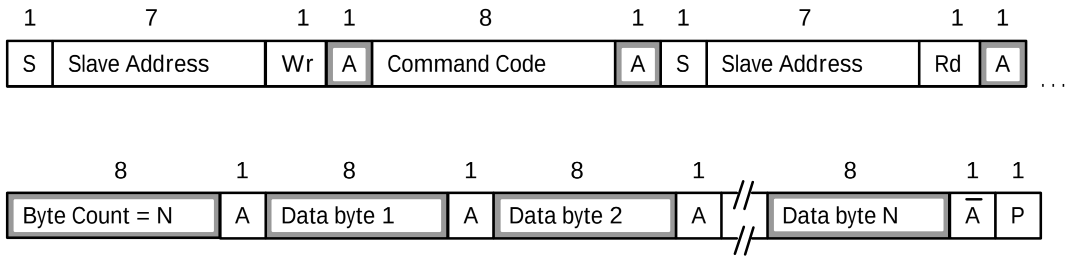
3.3.7. Block Read/Write

The Block Write begins with a slave address and a write condition. After the command code the host issues a byte count which describes how many more bytes will follow in the message. If a slave had 20 bytes to send, the first byte would be the number 20 (14h), followed by the 20 bytes of data. The byte count may not be 0. A Block Read or Write is allowed to transfer a maximum of 32 data bytes.



Block Write

A Block Read differs from a block write in that the repeated start condition exists to satisfy the I²C specification's requirement for a change in the transfer direction.

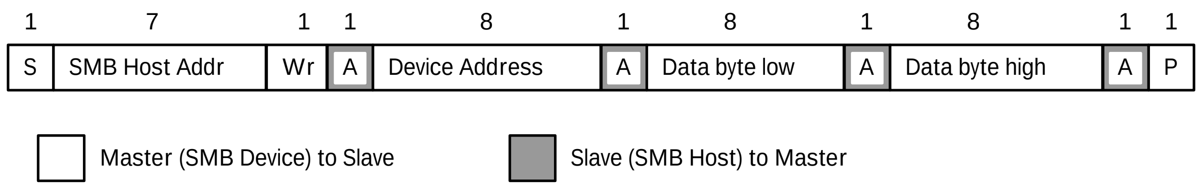


Block Read

3.4. Communicating with the Host

A message destined for the host could appear from an unknown device in an unknown format. To prevent possible confusion on the host's part, only one method of communication is allowed, a modified Write Word. The standard Write Word protocol is modified by replacing the command code with the calling device's address. This protocol is used when an SMB device becomes a **master** to communicate with the SMB host acting as a **slave**.

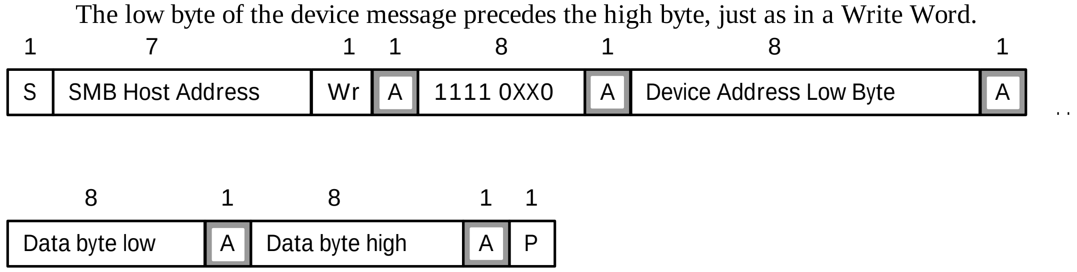
Device to Host communication will begin with the host address. The message's Command Code will actually be the initiating device's address. The host now knows the origin of the following 16 bits of device status.



7-bit Addressable Device to Host Communication

The Write Word protocol will be modified slightly for 10-bit addressing. If the device has a 10-bit address, it sends the I²C reserved address for 10-bit addressing (1111 0XX) followed by a 0 to make it 8 bits, the undefined bits being the 2 most significant bits of the 10-bit address. The next byte completes the address. 16 bits of device status follow.

The low byte of the device message precedes the high byte, just as in a Write Word.



10-bit Addressable Device to Host Communication

3.5. Reporting Errors

Any transfer may be aborted by either the slave or the master -- the master can issue a Stop Condition and the slave can withhold acknowledgment after any byte or cause a timeout to occur thus terminating the transfer.

If the device detects an error, it may signal it to the master.. The master can later visit the slave's Error Flag (if it is supported) to find out what went wrong. It is optional for the master to check and it is optional for the slave to provide the Error Flag.

Withholding acknowledgment is required for the last byte in a read operation under the I^{2}C specification. This acknowledgment, or lack thereof, is generated by the master and therefore will not be interpreted as an error.

A device may decide to generate an error indication for one or more of the following reasons:

- Device is not ready to process the request for data (either read or write)
- Device does not recognize the command code or function requested
- Device does not permit the command code or function requested
- Overflow or underflow condition
- Incorrect size of data in a block read/write transfer
- Unrecognized or unsupported data transfer protocol used in transaction
- Any other known or unknown error condition

The error may be generated in order to stop the transaction and indicate that any data already transferred is not reliable.

A device may signal an error by:

- Not ACKnowledge signal at the end of a byte transfer. This method is used when the SMBus device is acting as a slave-receiver and is receiving data from a master-transmitter. The master-transmitter, usually the SMBus Host device, will look for an ACKnowledge bit from the slave after every byte is transmitted

- Hold either the SMBCLK or SMBDATA lines low for longer than $T_{TIMEOUT}$ to cause a device timeout to occur. When a device is acting as a slave-transmitter, the ACKnowledge bit is generated from the master-receiver when the preceding byte has been received correctly. A slave-transmitter may signal an error condition by holding either the SMBCLK or SMBDATA lines low for longer than the $T_{TIMEOUT}$ period. Doing so will cause a timeout condition and the SMBus will then be restored to an idle state (both SMBCLK and SMBDATA returned high.)

In either case, the master device must attempt to generate a Stop Condition on the SMBus to end the transaction.

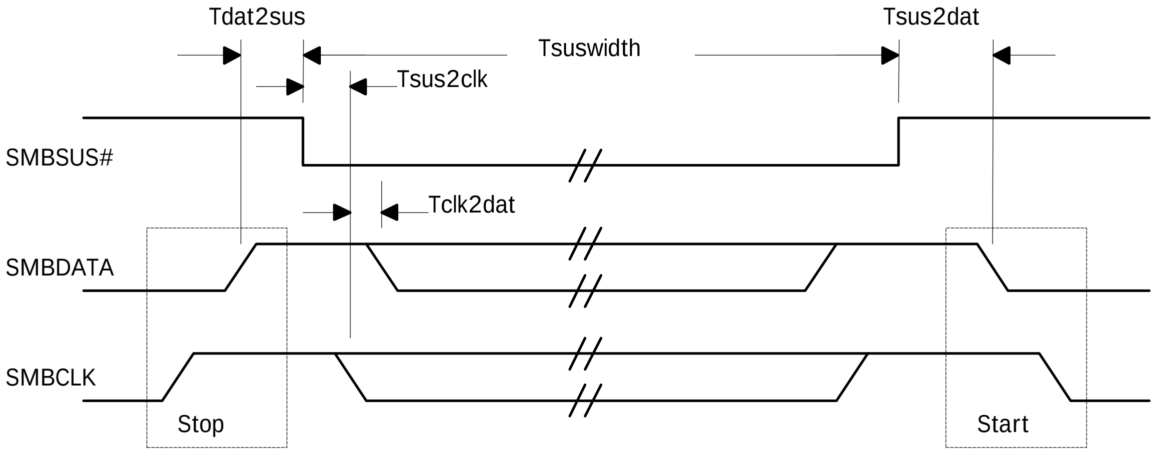
4. Appendix A: Optional SMB Signals

4.1. SMBSUS#

An **optional** third signal, SMBSUS#, goes low when the system enters the Suspend Mode. Suspend Mode refers to a low-power mode where most devices are stalled or powered down. Upon resume, the SMBSUS# returns high. The system then returns all devices to there operational state.

The SMBSUS# signal is included for clarity and completeness since many of the functions served by the System Management Bus relate to suspend and resume of the system.

The system can use the SMBCLK and SMBDATA lines to program device behavior. During normal operating mode the system may issue configuration commands to different devices. Those commands may tell the device how it is supposed to behave whenever the SMBSUS# line goes active. For example, the system might tell a power plane switcher to turn off power to the hard drive but keep the keyboard controller on when the system goes into suspend mode.



SMB During Suspend

Timing	Min	Typical
$T_{DAT2SUS}$	0ns	tens of ms
$T_{sus2clk}$	0ns	tens of ns
$T_{clk2dat}$	0ns	0ns
$T_{suswidth}$		minutes, hours, weeks
$T_{sus2dat}$	0ns	hundreds of ms

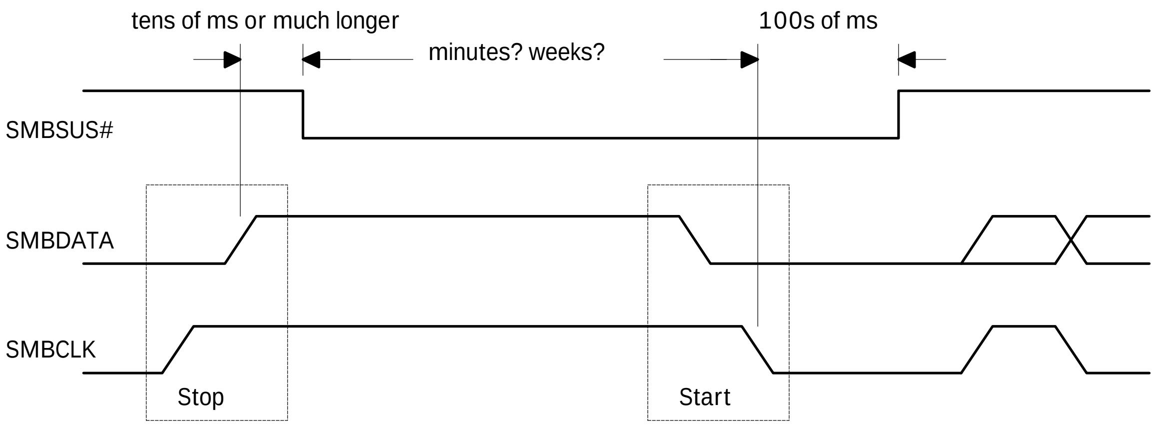
SMBSUS# is not a wired-OR signal. It is an output from the device that controls system management functions, and an input to everything else.

During suspend there is no activity on the System Management Bus unless the SMB is used to resume from suspend mode. Activity resumes after coming out of suspend.

Anytime after a stop condition the SMBSUS# signal may go active low signifying the system is going into Suspend Mode. This can happen immediately (min = 0ns), but will probably take much longer. In fact, the final SMB message might terminate minutes or hours before SMBSUS# goes low. Suspend Mode could last a couple of seconds, minutes, hours, or weeks. Before the System Management Bus can send another message the system must come out of

Suspend Mode, a process known as Resume. The resume process probably has to supply voltage to the System Management Bus anyway, although the SMB may be awake during suspend. The resume process can take a long time, perhaps hundreds of milliseconds. Careful power-down sequencing of the SMBCLK and SMBDATA pullups will prevent devices from falsely seeing a start condition on the bus.

If power is supplied to the System Management Bus during suspend, a device may use it to awaken the system. The host or another device will watch for a start condition on the bus. That device initiates the resume sequence. Communication on the bus resumes when the system is out of suspend.



Using SMB to Resume from Suspend

Since the SMBSUS# is an optional signal, some system devices may not know if the system is in suspend mode or not. Such a device may assume that if both SMBCLK and SMBDATA lines are high that the bus is alive and active. The possibility exists that this device may try to send a critical message to another device which accepts the SMBSUS# signal and is therefore asleep. Therefore it is important that a system is able to resume on a start condition if a non-suspendable master resides on the System Management Bus and that master can send critical messages to suspended devices.

4.2. SMBALERT#

Another **optional** signal is an interrupt line for devices that want to trade their ability to master for a pin. SMBALERT# is a wired-or signal just as the SMBCLK and SMBDATA signals are. SMBALERT# is used in conjunction with the SMB General Call Address. Messages invoked with the SMB are 2 bytes long.

A slave-only device can signal the host through SMBALERT# that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (ARA). Only the device(s) which pulled SMBALERT# low will acknowledge the Alert Response Address. The host performs a modified Receive Byte operation.

1	7	1	1	8	1	1
S	Alert Response Address	Rd	A	Device Address	$\overline{A}$	P

A 7-bit Addressable Device Responds to an ARA

1	7	1	1	8	1	8	1	1
S	Alert Response Address	Rd	A	1111 0XX0	A	Device Address Low Byte	$\overline{A}$	P

A 10-bit Addressable Device Responds to an ARA

If more than one device pulls SMBALERT# low, the highest priority (lowest address) device will win communication rights via standard I^2C arbitration during the slave address transfer.

After acknowledging the slave address the device should disengage its SMBALERT# pulldown. If the host still sees SMBALERT# low when the message transfer is complete, it knows to read the ARA again.

A host which does not implement the SMBALERT# signal may periodically access the ARA.

5. Appendix B: SMB Device Address Assignments

The following table represents the SMBus device assignments as of February 15, 1995.

Slave Address	Description	Specification
0001 000	SMB Host	System Management Bus Specification[1] v 1.0 February 1995
0001 001	Smart Battery Charger	Smart Battery Charger Specification[1] v 0.95a February 1995
0001 010	Smart Battery Selector	Smart Battery Selector Specification[1] v 0.9 March 1995
0001 011	Smart Battery	Smart Battery Data Specification[1] v 1.0 February 1995
0001 100	SMB Alert Response	System Management Bus Specification[1] v 1.0 February 1995
0101 000	ACCESS.bus host	
0101 100	LCD Contrast Controller	TBA
0101 101	CCFL Backlight Driver	TBA
0110 111	ACCESS.bus default address	
1000 0XX	PCMCIA Socket Controllers (4 addresses)	TBA
1000 100	(VGA) Graphics Controller	TBA
1001 0XX	Unrestricted addresses	System Management Bus Specification[1] v 1.0 February 1995
1100 001	SMB Device Default Address	System Management Bus Specification[1] v 1.0 February 1995

Notes
[1] - Available from Intel Corporation 1-800-628-8686 (International 1-916-356-3551)

###