Smart Battery System Specifications

Smart Battery Data Specification

Revision 1.0
February 15, 1995

Questions and comments regarding this specification may be forwarded to:

Intel Corporation
PHONE: (8am-5pm PST) (800) 628-8686
FAX: (916) 356-6100
INTERNATIONAL (916) 356-3551
Email: IAL_Support@ccm.hf.intel.com

Table of Contents

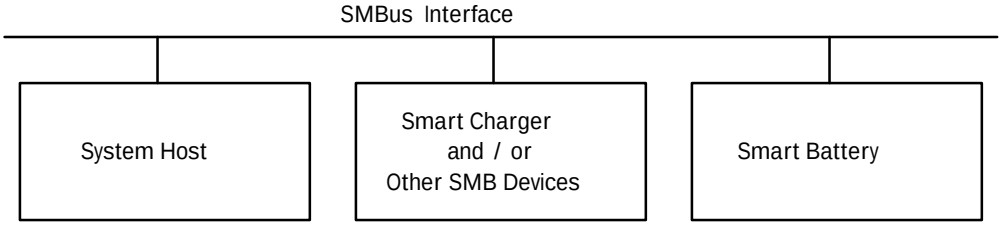
Revision History

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

1. Introduction

The Smart Battery Specification presents an ideal solution for many of the issues related to batteries used in portable electronic equipment such as laptop computer systems, cellular telephones or video cameras. Batteries presently have a number of limitations from both the user's and the equipment's perspective. First and foremost, they represent an unpredictable source of power. Typically a user has little advance knowledge that their battery is about to run out or how much operating time is left. Second, equipment powered by the battery cannot determine if the battery, in its present state, is capable of supplying adequate power for an additional load (such as spinning up a hard disk). Third, battery chargers must be individually tailored for use with a specific battery chemistry and may cause damage if used on another battery with a different chemistry.

This specification, as depicted below, defines the data that flows across the SMBus between the Smart Battery, SMBus Host, Smart Battery Charger and other devices. A more detailed description of the electrical interface and data protocols can be found in the supplementary documentation (refer to the *References* section).



The Major Components of the SMBus Interface :
Electrical: Refer to the System Management Bus Specification for more information
Protocol: Refer to the System Management Bus Specification for more information
Data: Described in this specification

This specification defines the information that the Smart Battery supplies to its user. It is not designed to limit innovation amongst battery manufacturers, but rather, provide the user and the SMBus Host with a consistent set of information about any particular Smart Battery.

1.1. Scope

This document specifies the data set that is communicated by a Smart Battery. The electrical and mechanical specifications are covered by other specifications (refer to the *References* section). This specification is generic with regard to the type of battery chemistry, the battery pack voltage, the battery pack capacity as well as the battery pack's physical packaging.

1.2. Audience

The audience for this document includes:

- Smart Battery manufacturers
- Readers of the System Management Bus Specification
- Designers of Smart Battery device drivers
- Designers of power management systems for Smart Battery powered portable electronic equipment

2. References

- *The I²C-bus and how to use it* (includes the specification), Phillips Semiconductors, January 1992
- *System Management Bus Specification, Revision 1.0*, Intel Inc., February, 1995
- *Smart Battery Charger Specification Revision 0.95a*, Duracell/Intel Inc., February, 1995
- *System Management Bus BIOS Interface Specification, Revision 1.0, February*, 1995
- *IEC SC21A - "Alkaline Secondary Cells and Batteries"*, IEC committee 21, Sub-committee A responsible for development of standard battery pack sizes and electrical specifications
- *IEC SC48B - "Connectors"*, IEC committee 48, Sub-committee B responsible for development of connector standards for the Smart Battery

3. Definitions

- **APM:** Advanced Power Management. A BIOS interface defined to enable system-wide power management control via software.
- **Battery:** One or more cells that are designed to provide electrical power.
- **Cell:** The cell is the smallest unit in a battery. Most batteries consist of several cells connected in series.
- **I²C-bus**: A two-wire bus developed by Phillips, used to transport data between low-speed devices.
- **Smart Battery:** A battery equipped with specialized hardware that provides present state, calculated and predicted information to its SMBus Host under software control. The content and method are described in this specification.
- **Smart Battery Charger:** A battery charger that periodically communicates with a Smart Battery and alters its charging characteristics in response to information provided by the Smart Battery.
- **Smart Device:** An electronic device or module that communicates over the SMBus with the SMBus Host and/or other Smart Devices. For example the back-light controller in a Notebook computer can be implemented as a Smart Device.
- **SMBus**: The System Management Bus is a specific implementation of an I²C-bus that describes data protocols, device addresses and additional electrical requirements that is designed to physically transport commands and information between the Smart Battery, SMBus Host, Smart Battery Charger and other Smart Devices.
- **SMBus Host:** A piece of portable electronic equipment powered by a Smart Battery. It is able to communicate with the Smart Battery and use information provided by the battery.

4. Smart Battery

In most systems today, the user never knows how much charge is left in the battery. While the user may translate this to the simple question "How long will this device continue to operate?"; the answer is complex. Many products that attempt to answer the question use the system's hardware to account for the battery's charge state. This approach is destined to fail when different batteries are used because the battery's characteristics and history are associated with the system, not the battery. The Smart Battery fixes this problem by maintaining its own information, thus allowing for a mixture of batteries (different chemistries and/or charge states) to be used in a device. The user will now have access to accurate information because each Smart Battery will accurately report its own characteristics.

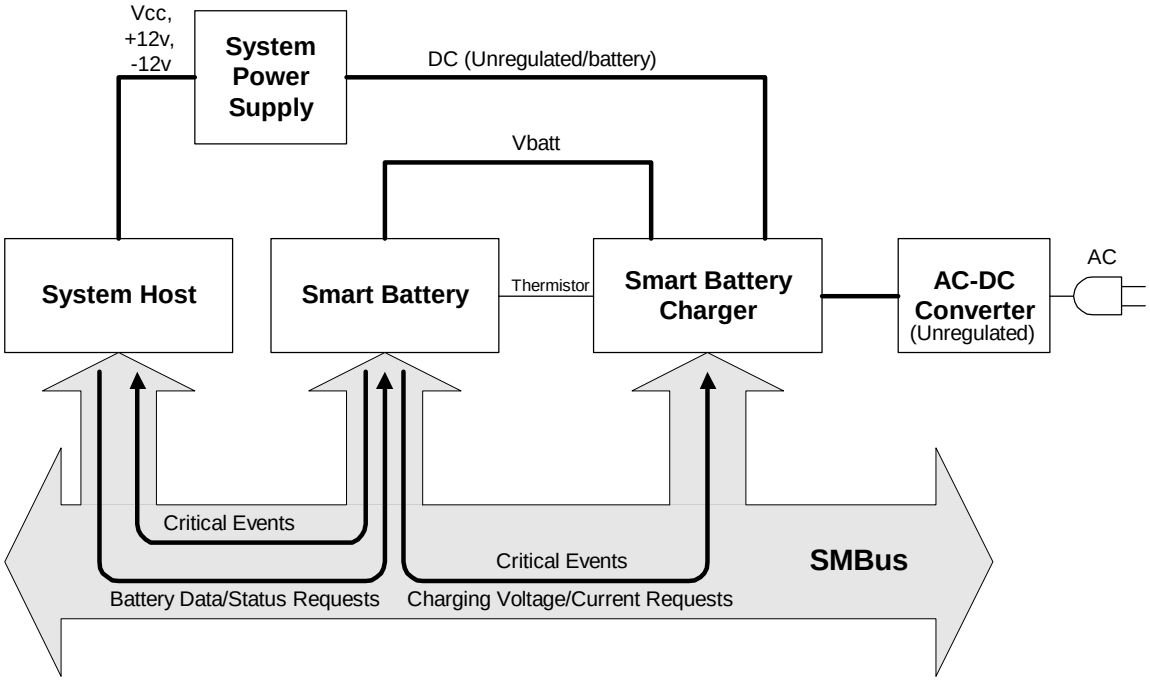
A good example is a video camcorder where a user may have multiple batteries each with different capacities as well as different charge states. Even with an accurate state-of-charge indication, a full one AH (ampere hour) battery is not equivalent to a full 1.5 AH battery. Though they both can power the same camcorder, what the user wants to know is whether or not either of these batteries has adequate capacity to record a one hour event. The Smart Battery

provides the user with accurate state of charge information along with an accurate prediction of the remaining operating time.

The goal of the Smart Battery interface is to provide adequate information for power management and charge control regardless of the particular battery's chemistry. Even though the major consumer of the battery information is the user, the system can also take advantage by using power consumption information to better manage its own power use. A charging system will be able to tell the user how long it will take to fully charge the battery.

4.1. Smart Battery Model

One possible Smart Battery model is a system consisting of a battery, battery charger and a host (notebook computer, video camera, cellular phone, or other portable electronic equipment). Since it is a system, it is important to examine the components and their interactions.



The Smart Battery Charger is a charging circuit that periodically polls the Smart Battery for its charging characteristics then adjusts its output to match the Smart Battery's requirements. This allows the battery to control its own charge cycle. Optionally, it may not allow the Smart Battery to supply power to the rest of the system when the Smart Battery is fully charged and the system is connected to AC power thus prolonging the life of the battery. The Smart Battery Charger will also receive critical events from the Smart Battery when it detects a problem. These include alarms for: over charge, over voltage, over temperature and temperature increasing too rapidly.

The SMBus Host represents a piece of electronic equipment that is powered by a Smart Battery and that can communicate with the Smart Battery. The SMBus Host requests information from the battery and then uses it in the system's power management scheme and/or uses it to provide the user information about the battery's present state and capabilities. The SMBus Host will also receive critical events from the Smart Battery when it detects a problem. In addition to the alarms sent to the Smart Battery Charger, it receives alarms for end of discharge, remaining

capacity below the user set threshold value and remaining run time below the user set threshold value.

4.2. Software Definition

The software interface is separated into three parts: SMBus Host-to-battery, charger-to-battery and battery-to-charger or SMBus Host. Additionally, a discussion about error signaling and handling is included.

4.2.1. SMBus Host-to-Smart Battery

The SMBus Host-to-battery communication is used to get data that is either presented to a user or to the SMBus Host's power management system. The user can get two types of data from the battery: factual data and predictive data. Factual data can be measured, such as temperature, pack voltage or charge/discharge current, or it can be a battery characteristic, such as the battery's chemistry. Predictive data is calculated, based on the battery's present state and the battery's characteristics, such as the battery's remaining life at the present rate of drain. Additionally, since the battery has a clock, information can be presented as a rolling average over a fixed interval.

The power management system may query a device driver to determine if an action will cause harm to the system's integrity. For example, spinning up a disk drive at the end of the battery's charge might cause its output voltage to drop below acceptable limits thus causing a system failure. In order to prevent this, the device driver needs information from the battery that will cause it to do the right thing. If the driver queries the battery and discovers that not enough power is available, it can request that the power management system turn off a non-critical power use such as the LCD screen back-light and then try again.

SMBus Host-to-Smart Battery communications are performed:

- To allow the user to know the Smart Battery's remaining life
- To tell the user how long it will take to charge the Smart Battery
- To allow Smart Batteries to provide accurate information to their user
- To determine the SMBus Host's real-time power requirements
- To enable power management based on "real" information supplied by the battery
- To enable battery manufacturers to collect information about a Smart Battery's usage
- To allow battery manufacturers to electronically "stamp" batteries at time of manufacture

.

4.2.2. Smart Battery Charger-to-Smart Battery or Smart Battery-to-Smart Battery Charger

An internal or external battery charger must understand the characteristics of the battery it is charging. Today's laptops, using NiMH and NiCd batteries, apply a constant current to the battery. End-of-charge is determined by charger noting a sharp rise in the battery's internal temperature. There is a potential problem with this scheme; when a battery with a different chemistry is placed in the same size package, even though the voltage may be the same, the charging characteristics may not.

A better method is to have the battery tell the charger when charging is complete and how to adjust the charging voltage and current so they best match the battery's present state. Chargers that cooperate with the battery have two distinct advantages over the simple model that watches for a thermal rise: first, they provide the battery with all the power it can handle (that is, maximum safe charge) without overcharging, and second, they will recognize and correctly charge batteries with different chemistries and voltages.

Smart Battery Charger to Smart Battery communications are performed:

- To allow Smart Batteries to be charged as rapidly and as safely as possible

- To allow new and different battery technologies to be used in existing equipment
- To allow access to the "correct" charger algorithm for the battery

4.2.3. Smart Battery-to-SMBus Host or Smart Battery Charger

A Smart Battery must have the ability to inform the SMBus Host of potentially critical conditions. These notifications represent a final effort on the part of the battery to inform both the Smart Battery Charger and the SMBus Host that power is about to fail or that the battery is being overcharged. The Smart Battery expects that the user, Smart Battery Charger or SMBus Host will take the appropriate corrective action.

Smart Battery-to-SMBus Host or Smart Battery Charger communications are performed:

- To allow the Smart Battery to warn other system components of potential problems.
- To allow the Smart Battery to warn the user about potentially dangerous situations that they can rectify.
- To allow the Smart Battery to instruct the Smart Battery Charger what Charge Current and Charge Voltage it would like to be charged with.

4.3. Software Error Detection and Signaling

The Smart Battery provides a simple system for error signaling. The error system is designed to minimize the amount of traffic on the I²C bus and the amount of code required to communicate with the battery.

4.3.1. Error Detection

When a Smart Battery detects an error condition (such as an unsupported command, data unavailable, busy or bad data) it signals the SMBus Host that an error has been detected. All functions processed by the Smart Battery are assumed to be error-free unless the Smart Battery signals the SMBus Host that an error has occurred. After processing each function, the Smart Battery must place the appropriate error code in its ERROR register (note: this includes “OK” or “no error detected”).

4.3.2. Error Signaling

A Smart Battery signals the SMBus Host that it has detected an unrecoverable error by taking advantage of the I²C bus requirement that an acknowledge bit must be sent by the slave after every byte is transferred. When the Smart Battery fails to provide the acknowledge bit, the SMBus Host is obliged to generate a STOP condition, thus causing a premature termination of the transfer. This signals the SMBus Host that an error has occurred. For some functions, invalid data is used as a signal that valid data is NOT available. In these cases, the function will place OK in the error register.

The Smart Battery must ALWAYS acknowledge its own address. Failure to do so might cause the SMBus Host or Smart Battery Charger to incorrectly assume the Smart Battery is NOT present in the system. The Smart Battery may choose not to acknowledge any byte following its address if it is busy or otherwise unable to respond.

4.3.3. Error Handling

When the SMBus Host detects that an error has occurred, it uses the BatteryStatus() function to get the error code from the Smart Battery. In the case where the error code is OK, there was an unrecognized bus error rather than a Smart Battery error and the SMBus Host should repeat the original function.

4.4. Smart Battery Characteristics

The Smart Battery may or may not be present in a system. Additionally, it may dynamically be added or removed while the system is powered. Therefore, it must exhibit predictable behaviors

when inserted in a system and/or when the system is turned on. The following is a description of the battery's states and a description of the actions that take place as a result state changes.

4.4.1. Initial Conditions

When a Smart Battery is first delivered, several values are already set:

- RemainingCapacityAlarm() which is set to 10% of the DesignCapacity()
- RemainingTimeAlarm() which is set to 10 minutes
- The BatteryMode()'s CHARGER_MODE bit and CAPACITY_MODE bit which are cleared
- The BatteryStatus() INITIALIZED bit which is set
- The CycleCount() which is cleared

4.4.2. On State

The Smart Battery enters the "on state" whenever the SMBus clock goes high. The battery should be active and able to communicate via the SMBus within 1 ms of detecting the SMBus clock going high. The battery may not disrupt traffic on the SMBus.

4.4.3. Off State

The Smart Battery must enter the "off state" whenever the SMBus clock and data lines both remain low for greater than two seconds. A Smart Battery may enter the "off state" in less time, however, in no case can it enter the off state in less than 10 times the SMB device TTIMEOUT value. The SMB lines may go low because the battery is removed from the system, the SMB host forces them low in order to reset the battery or power is removed from the SMBus (for example, when the system is turned off).

4.4.4. Off to On State Transition

Whenever the Smart Battery enters the "on state", the following values are cleared:

- The BatteryMode() CHARGER_MODE bit
- The BatteryMode() CAPACITY_MODE bit

The Smart Battery may not begin broadcasting ChargingVoltage(), ChargingCurrent() or AlarmWarning() messages to either the host or charger for at least 10 seconds after entering the "on state."

4.4.5. On to Off State Transition

Whenever the Smart Battery enters the "off state", the following values are cleared:

- If active, the BatteryMode() CHARGE_CONTROLLER_ENABLED bit
- If active, the BatteryMode() PRIMARY_BATTERY bit

The Smart Battery defaults to disable the internal charge controller in-order to prevent possible overloading of the power supply in systems when more than one battery is present. Without this default, it is possible for multiple batteries to concurrently demand more charging power than is available potentially starving the system of power.

The Smart Battery's defaults to act as a secondary battery in order to prevent large amounts of energy that could potentially flow between two primary batteries not at the same charge level.

5. Smart Battery Interface

The following functions are used by the Smart Battery to communicate with a SMBus Host, Smart Battery Charger and other devices connected via the SMBus.

The functions are described as follows:

FunctionName() 0xnn (command code)

Description:
A brief description of the function.
Purpose:
The purpose of the function, and an example where appropriate.
SMBus Protocol: Refer to Section 6 for details.
Input, Output or Input/Output: A description of the data supplied to or returned by the function.

The data is described as follows:

data type:	The type of data the function conveys (See Appendix B)
Units:	The units the data is presented in
Range:	The range of valid data
Granularity:	See paragraph below
Accuracy:	How "good" is the data.

Granularity is described in this specification as a percentage of an associated maximum value. The data's granularity is determined by several factors. For measured data, the number of bits supplied by the A-D converter used in the Smart Battery generally will determine the granularity. In the case of calculated values, the granularity is generally determined by the granularity of the least-accurate data.

For example, for a battery with a Design Voltage of 4.8 volts (4800 mv) the values would be:

	8-bit A/D	9-bit A/D	10-bit A/D	11-bit A/D
Granularity (%)	0.40	0.20	0.10	0.05
Actual value (mv)	19.2	9.6	4.8	2.4

However, for a 12 volt (12000 mv) battery they would be:

	8-bit A/D	9-bit A/D	10-bit A/D	11-bit A/D
Granularity (%)	0.40	0.20	0.10	0.05
Actual value (mv)	48.0	24.0	12.0	6

The fractional granularity values will always be rounded up to the next integer value. By specifying Voltage() in terms of DesignVoltage() rather than in absolute numerical values, the Smart Battery can supply useful data values while still retaining adequate dynamic range.

Accuracy is specified either relative to some battery characteristic (such as DesignVoltage()) or relative to a battery characteristic and the battery-supplied error value, MaxError(). Generally, absolute accuracy is possible only for values that are known at the time the battery is manufactured. For example, the temperature's accuracy is known at the time of manufacture.

This specification implies that an A-D with at least a 9-bit resolution be used to meet the minimum granularity requirements for "measured" values. Although the granularity and accuracy values specified represent a minimum standard of performance, better performance is encouraged.

For various classes of battery packs, the voltage, current and other parameters may have their limits or ranges clarified in ancillary battery pack specifications. These specifications will serve to better define the range over which high accuracy is required. Many of the default values contained in this specification may be superseded for a class of battery packs by values defined in an industry-wide ancillary pack specification. For example, although the battery temperature data can theoretically be reported ranging from absolute zero to the surface temperature of the sun, a class of battery packs destined for the consumer market may only require a temperature range of -10 to 45°C.

A Smart Battery that complies with this specification must support all the command codes contained in this specification. It must support the defaults as specified. Additionally, it must support all modes and functions specified except those which it can explicitly signal the presence or absence thereof (e.g. the presence of an internal charge controller and the ability to enable or disable that controller).

5.1. SMBus Host-to-Smart Battery Messages

5.1.1. ManufacturerAccess() (0x00)

Description:
This function is optional and its meaning is implementation specific.
Purpose:
The ManufacturerAccess() function's purpose is manufacturer specific.
SMBus Protocol: Read or Write Word
Input/Output: word -- Content determined by the Smart Battery's manufacturer

5.1.2. RemainingCapacityAlarm() (0x01)

Description:
Sets or gets the Low Capacity threshold value. Whenever the RemainingCapacity() falls below the Low Capacity value, the Smart Battery sends AlarmWarning() messages to the SMBus Host with the REMAINING_CAPACITY_ALARM bit set. A Low Capacity value of 0 disables this alarm. The Low Capacity value is set to 10% of design capacity at time of manufacture. The Low Capacity value will remain unchanged until altered by the RemainingCapacityAlarm() function. The Low Capacity value may be expressed in either current (ma) or power (10mwH) depending on the setting of the BatteryMode()'s CAPACITY_MODE bit (see BatteryMode()).
Purpose:
The RemainingCapacityAlarm() function can be used by systems that know how much power they will require to save their operating state. It enables those systems to more finely control the point at which they transition into suspend or hibernate state. The Low Capacity value can be read to verify the value in use by the Smart Battery's Low Capacity alarm.
SMBus Protocol: Read or Write Word
Input/Output: unsigned int -- value below which Low Capacity messages will be sent

	Battery Mode	
	CAPACITY_MODE bit = 0	CAPACITY_MODE bit = 1
Units:	maH @ C/5	10mwH @ P/5
Range:	0 to 65,535 maH	0 to 65,535 10mwH
Granularity:	not applicable	
Accuracy	see RemainingCapacity()	

5.1.3. RemainingTimeAlarm() (0x02)

Description:

Sets or gets the Remaining Time alarm value. Whenever the AverageTimeToEmpty() falls below the Remaining Time value, the Smart Battery sends AlarmWarning() messages to the SMBus Host with the REMAINING_TIME_ALARM bit set. A Remaining Time value of 0 effectively disables this alarm. The Remaining Time value is set to 10 minutes at time of manufacture. The Remaining Time value will remain unchanged until altered by the RemainingTimeAlarm() function.

Purpose:

The RemainingTimeAlarm() function can be used by systems that want to adjust when the remaining time alarm warning is sent. The Remaining Time value can be read to verify the value in use by the Smart Battery's RemainingTimeAlarm().

SMBus Protocol: Read or Write Word

Input/Output: unsigned int -- the point below which Remaining Time messages will be sent

Units:	minutes
Range:	0 to 65,535 minutes
Granularity:	not applicable
Accuracy:	see AverageTimeToEmpty()

5.1.4. BatteryMode() (0x03)

Description:

This function selects the various battery operational modes and reports the battery's capabilities, modes, and condition.

Defined capabilities include:

- Optional internal charge controller supported (INTERNAL_CHARGE_CONTROLLER)
- Optional internal primary battery control (PRIMARY_BATTERY_SUPPORT)

Defined modes include:

- Battery's capacity information is specified in maH or 10mwH (CAPACITY_MODE bit)
- Whether the ChargingCurrent() and ChargingVoltage() values are broadcast to the Smart Battery Charger when the Smart Battery requires charging (CHARGER_MODE bit)
- Optional internal charge controller enable (CHARGE_CONTROLLER_ENABLED)
- Optional internal primary battery control enable (PRIMARY_BATTERY)

The defined condition is the battery requesting a conditioning cycle (CONDITION_FLAG).

Purpose:

The CAPACITY_MODE bit allows power management systems to best match their electrical characteristics with those reported by the battery. For example, a switching power supply represents a constant power load, whereas a linear supply is better represented by a constant current model. The CHARGER_MODE bit allows a SMBus Host or Smart Battery Charger to override the Smart Battery's desired charging parameters by disabling the Smart Battery's broadcast of the ChargingCurrent() and ChargingVoltage(). The CONDITION_FLAG bit allows the battery to request a conditioning cycle.

SMBus Protocol: Read or Write Word

Input/Output: unsigned int - bit mapped - see below

Units:	not applicable
Range:	0...1
Granularity:	not applicable
Accuracy:	not applicable

The BatteryMode() word is divided into two halves, the MSB which is read/write and the LSB which is read only. Attempts to set (write 1's) the reserved bits in the MSB are prohibited.

15						MSB	8	7				LSB			0
R/W	R/W	res	res	res	res	R/W	R/W	R	res	res	res	res	res	R	R

The following table summarizes the meanings of the individual bits in the BatteryMode() word and specifies the default values if any. Power-on default values, where applicable, are discussed in section 4.4.

Field	Bits Used	Format	Allowable Values
INTERNAL_CHARGE_ CONTROLLER	0	read only bit flag	0 - Optional Function Not Supported 1 - Internal Charge Controller
PRIMARY_BATTERY_ SUPPORT	1	read only bit flag	0 - Optional Function Not Supported 1 - Primary or Secondary Battery Support
reserved	2-6		
CONDITION_FLAG	7	read only bit flag	0 - Battery OK 1 - Conditioning Cycle Requested
CHARGE_CONTROLLER _ENABLED	8	r/w bit flag	0 - Internal Charge Control Disabled (default) 1 - Internal Charge Control Enabled
PRIMARY_BATTERY	9	r/w bit flag	0 - Battery operating in its secondary role (default) 1 - Battery operating in its primary role
reserved	10-13		
CHARGER_MODE	14	r/w bit flag	0 - Enable broadcast to charger (default) 1 - Disable broadcast to charger
CAPACITY_MODE	15	r/w bit flag	0 - Report in ma or maH (default) 1 - Report in 10mw or 10mwH

INTERNAL_CHARGE_CONTROLLER bit set indicates that the battery pack contains its own internal charge controller. When the bit is set, this optional function is supported and the CHARGE_CONTROLLER_ENABLED bit will be activated.

PRIMARY_BATTERY_SUPPORT bit set indicates that the battery pack has the ability to act as either the primary or secondary battery in a system. When the bit is set, this optional function is supported and the PRIMARY_BATTERY bit will be activated.

CONDITION_FLAG bit set indicates that the battery is requesting a conditioning cycle. A conditioning cycle may be requested because of the characteristics of the battery chemistry and/or the electronics in combination with the usage pattern. The conditioning cycle is pack specific, but typically will consist of a full-charge to full-discharge back to full-charge of the pack. The battery will clear this flag after it detects that a conditioning cycle has been completed.

CHARGE_CONTROLLER_ENABLED bit is set to enable the battery pack's internal charge controller. When this bit is cleared, the internal charge controller is disabled (default). This bit is active only when the INTERNAL_CHARGE_CONTROLLER bit is set. The status of a battery pack's internal charge controller can be determined by reading this bit.

PRIMARY_BATTERY bit is set to enable a battery to operate as the primary battery in a system. When this bit is cleared, the battery operates in a secondary role (default). This bit is active only when the PRIMARY_BATTERY_SUPPORT bit is set. The role that the battery is playing can be determined by reading this bit.

CHARGER_MODE bit enables or disables the Smart Battery's transmission of ChargingCurrent() and ChargingVoltage() messages to the Smart Battery Charger. When set, the Smart Battery will NOT transmit ChargingCurrent() and ChargingVoltage() values to the Smart Battery Charger. When cleared, the Smart Battery will transmit the ChargingCurrent() and ChargingVoltage() values to the Smart Battery Charger when charging is desired. (See Section 5.3 for a more detailed explanation.)

CAPACITY_MODE bit indicates if capacity information will be reported in ma/maH or 10mw/10mwH. When set, the capacity information will be reported in 10mw/10mwH as appropriate. When cleared, the capacity information will be reported in ma/maH as appropriate.

Note1: The following functions are changed to accept or return values in ma/maH or 10mw/10mwH depending on the CAPACITY_MODE bit:
RemainingCapacityAlarm()
AtRate()
RemainingCapacity()
FullChargeCapacity()
DesignCapacity()
Note2: The following functions are calculated on the basis of capacity and may be calculated differently depending on the CAPACITY_MODE bit:
AtRateOK()
AtRateTimeToEmpty()
RunTimeToEmpty()
AverageTimeToEmpty()
Remaining Time Alarm()
BatteryStatus()

5.1.5. AtRate() (0x04)

Description:
The AtRate() function is the first half of a two-function call-set used to set the AtRate value used in calculations made by the AtRateTimeToFull(), AtRateTimeToEmpty(), and AtRateOK() functions. The AtRate value may be expressed in either current (ma) or power (10mw) depending on the setting of the BatteryMode()'s CAPACITY_MODE bit.
Purpose:
Since the AtRate() function is the first half of a two-function call-set, it is followed by the second function of the call-set that calculates and returns a value based on the AtRate value and the battery's present state.

- When the AtRate value is positive, the AtRateTimeToFull() function returns the predicted time to full-charge at the AtRate value of charge.
- When the AtRate value is negative, the AtRateTimeToEmpty() function returns the predicted operating time at the AtRate value of discharge.
- When the AtRate value is negative, the AtRateOK() function returns a Boolean value that predicts the battery's ability to supply the AtRate value of *additional* discharge energy (current or power) for 10 seconds.

The AtRate value is set to zero at time of manufacture (default).
SMBus Protocol: Read or Write Word
Input/Output: signed int -- charge or discharge, the AtRate value is positive for charge, negative for discharge and zero for neither (default)

	Battery Mode	
	CAPACITY_MODE bit = 0	CAPACITY_MODE bit = 1
Units:	ma	10mw
Charge Range:	1 to 32,767 ma	1 to 32,767 10mw
Discharge Range:	-1 to -32,768 ma	-1 to -32,768 10mw
Granularity:	1 unit	
Accuracy:	not applicable	

5.1.6. AtRateTimeToFull() (0x05)

Description:
Returns the predicted remaining time to fully charge the battery at the AtRate value (ma).
Purpose:
The AtRateTimeToFull() function is part of a two-function call-set used to determine the predicted remaining charge time at the AtRate value in ma. It will be used immediately after the SMBus Host sets the AtRate value. Refer to AtRate().
SMBus Protocol: Read Word
Output: unsigned int -- predicted time in minutes to fully charge the battery

Units:	minutes
Range:	0 to 65,534 min
Granularity:	2 min or better
Accuracy:	$\pm \text{MaxError}() * \text{FullChargeCapacity}() \div \lvert\text{AtRate}()\rvert$
Invalid Data Indication:	65,535 indicates the battery is not being charged

5.1.7. AtRateTimeToEmpty() (0x06)

Description:
Returns the predicted remaining operating time if the battery is discharged at the AtRate value.
Purpose:
The AtRateTimeToEmpty() function is part of a two-function call-set used to determine the remaining operating time at the AtRate value. It will be used immediately after the SMBus Host sets the AtRate value. Refer to AtRate().
SMBus Protocol: Read Word
Output: unsigned int -- estimated operating time left

Units:	minutes
Range:	0 to 65,534 min
Granularity:	2 min or better
Accuracy:	$-0,+\text{MaxError}() * \text{FullChargeCapacity}() \div \lvert\text{AtRate}()\rvert$
Invalid Data Indication:	65,535 indicates the battery is not being discharged

5.1.8. AtRateOK() (0x07)

Description:
Returns a Boolean value that indicates whether or not the battery can deliver the AtRate value of additional energy for 10 seconds (Boolean). If the AtRate value is zero or positive, the AtRateOK() function will ALWAYS return true.
Purpose:
The AtRateOK() function is part of a two-function call-set used by power management systems to determine if the battery can safely supply enough energy for an additional load. It will be used immediately after the SMBus Host sets the AtRate value. Refer to AtRate().
SMBus Protocol: Read Word
Output: Boolean -- indicates if the battery can supply the additional energy requested

Units:	Boolean
Range:	TRUE, FALSE
Granularity:	not applicable
Accuracy:	not applicable

5.1.9. Temperature() (0x08)

Description:

Returns the cell-pack's internal temperature (°K). The actual operational temperature range will be defined on a pack by pack basis. Typically it will be in the range of -20°C to +75°C.

Purpose:

The Temperature() function provides accurate cell temperatures for use by battery chargers and thermal management systems. A battery charger will be able to use the temperature as a safety check. Thermal management systems may use the temperature because the battery is one of the largest thermal sources in a system.

SMBus Protocol: Read Word

Output: unsigned int -- cell temperature in tenth degree Kelvin increments

Units:	0.1°K
Range:	0 to +6553.5°K
Granularity:	0.5°K or better
Accuracy:	±3°K

5.1.10. Voltage() (0x09)

Description:

Returns the cell-pack voltage (mv).

Purpose:

The Voltage() function provides power management systems with an accurate battery terminal voltage. Power management systems can use this voltage, along with battery current information, to characterize devices they control. This ability will help enable intelligent, adaptive power management systems.

SMBus Protocol: Read Word

Output: unsigned int -- battery terminal voltage in milli-volts

Units:	mv
Range:	0 to 65,535 mv
Granularity:	0.2% of DesignVoltage()
Accuracy:	±1.0% of DesignVoltage()

5.1.11. Current() (0x0a)

Description:

Returns the current being supplied (or accepted) through the battery's terminals (ma).

Purpose:

The Current() function provides a snapshot for the power management system of the current flowing into or out of the battery. This information will be of particular use in power management systems because they can characterize individual devices and "tune" their operation to actual system power behavior.

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()

5.1.12. AverageCurrent() (0x0b)

Description:

Returns a one-minute rolling average based on at least 60 samples of 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()

5.1.13. MaxError() (0x0c)

Description:

Returns the expected margin of error (%) in the state of charge calculation. For example, when MaxError() returns 10% and RelativeStateOfCharge() returns 50%, the Relative StateOfCharge() is actually between 50 and 60%. The MaxError() of a battery is expected to increase until the Smart Battery identifies a condition that will give it higher confidence in its own accuracy. For example, when a Smart Battery senses that it has been fully charged from a fully discharged state, it may use that information to reset or partially reset MaxError(). The Smart Battery can signal when MaxError() has become too high by setting the CONDITION_FLAG bit in BatteryMode().

Purpose:

The MaxError() function does not exist on most systems today. It has real value to the user in two ways: first, to give the user a confidence level about the state of charge and second, to give the Power Management system information about how aggressive it should be, particularly as the battery nears the end of its life.

SMBus Protocol: Read Word

Output: unsigned int -- percent uncertainty for selected information

Units:	%
Range:	0 to 100%
Granularity:	1%
Accuracy:	not applicable

5.1.14. RelativeStateOfCharge() (0x0d)

Description:
Returns the predicted remaining battery capacity expressed as a percentage of FullChargeCapacity() (%).

Purpose:
The RelativeStateOfCharge() function exists on most systems today (a.k.a. Fuel Gauge). It is used to estimate the amount of charge remaining in the battery. The problem with this paradigm is that the tank size is variable. As standardized battery packs come into service, physical size will have less to do with the actual capacity. Although the RelativeStateOfCharge() will continue to be used, new paradigms will be developed to communicate battery capacity, thus diminishing it's importance.

SMBus Protocol: Read Word

Output: unsigned int -- percent of remaining capacity

Units:	%
Range:	0 to 100%
Granularity:	1%
Accuracy:	-0, +MaxError()

5.1.15. AbsoluteStateOfCharge() (0x0e)

Description:
Returns the predicted remaining battery capacity expressed as a percentage of DesignCapacity() (%). Note that AbsoluteStateOfCharge() can return values greater than 100%.

Purpose:
See RelativeStateOfCharge() function description.

SMBus Protocol: Read Word

Output: unsigned int -- percent of remaining capacity

Units:	%
Range:	0 to 100+%
Granularity:	1%
Accuracy:	-0, +MaxError()

5.1.16. RemainingCapacity() (0x0f)

Description:
Returns the predicted remaining battery capacity. The RemainingCapacity() value is expressed in either current (maH at a C/5 discharge rate) or power (10mwH at a P/5 discharge rate) depending on the setting of the BatteryMode()'s CAPACITY_MODE bit.

Purpose:
The RemainingCapacity() function returns the battery's remaining capacity. This information is a numeric indication of remaining charge given by the Absolute or Relative StateOfCharge() functions and may be in a better form for use by power management systems.

SMBus Protocol: Read Word

Output: unsigned int -- remaining charge in maH or 10mwH

	Battery Mode	
	CAPACITY_MODE bit = 0	CAPACITY_MODE bit = 1
Units:	maH	10mwH
Range:	0 to 65,535 maH	0 to 65,535 10mwH
Granularity:	0.2% of DesignCapacity() or better	
Accuracy:	-0, +MaxError() * FullChargeCapacity()	

5.1.17. FullChargeCapacity() (0x10)

Description:

Returns the predicted pack capacity when it is fully charged. The FullChargeCapacity() value is expressed in either current (maH at a C/5 discharge rate) or power (10mwH at a P/5 discharge rate) depending on the setting of the BatteryMode()'s CAPACITY_MODE bit.

Purpose:

The FullChargeCapacity() function provides the user with a means of understanding the "tank size" of their battery. This information, along with information about the original capacity of the battery, can be presented to the user as an indication of battery wear.

SMBus Protocol: Read Word

Output: unsigned int -- estimated full charge capacity in maH or 10mwH

	Battery Mode	
	CAPACITY_MODE bit = 0	CAPACITY_MODE bit = 1
Units:	maH	10mwH
Range:	0 to 65,535 maH	0 to 65,535 10mwH
Granularity:	0.2% of Design Capacity or better	
Accuracy:	-0, +MaxError() * FullChargeCapacity()	

5.1.18. RunTimeToEmpty() (0x11)

Description:

Returns the predicted remaining battery life at the present rate of discharge (minutes). The RunTimeToEmpty() value is calculated based on either current or power depending on the setting of the BatteryMode()'s CAPACITY_MODE bit. This is an important distinction because use of the wrong calculation mode may result in inaccurate return values.

Purpose:

The RunTimeToEmpty() can be used by the power management system to get information about the relative gain or loss in remaining battery life in response to a change in power policy. This information is NOT the same as the AverageTimeToEmpty(), which is not suitable to determine the effects that result from a change in power policy.

SMBus Protocol: Read Word

Output: unsigned int -- minutes of operation left

Units: minutes
Range: 0 to 65,534 min
Granularity: 2 min or better
Accuracy: -0, +MaxError() * FullChargeCapacity() / Current()
Invalid Data Indication: 65,535 indicates battery is not being discharged

5.1.19. AverageTimeToEmpty() (0x12)

Description:

Returns a one-minute rolling average of the predicted remaining battery life (minutes). The AverageTimeToEmpty() value is calculated based on either current or power depending on the setting of the BatteryMode()'s CAPACITY_MODE bit. This is an important distinction because use of the wrong calculation mode may result in inaccurate return values.

Purpose:

The AverageTimeToEmpty() displays state-of-charge information in a more useful way. By averaging the instantaneous estimations, the remaining time will not appear to "jump" around as it does on many of today's systems.

SMBus Protocol: Read Word

Output: unsigned int -- minutes of operation left

Units:	minutes
Range:	0 to 65,534 min
Granularity:	2 min or better
Accuracy:	-0, +MaxError() * FullChargeCapacity() / AverageCurrent()
Invalid Data Indication:	65,535 indicates battery is not being discharged

5.1.20. AverageTimeToFull() (0x13)

Description:

Returns a one minute rolling average of the predicted remaining time until the Smart Battery reaches full charge (minutes).

Purpose:

The AverageTimeToFull() function can be used by the SMBus Host's power management system to aid in its policy. It may also be used to find out how long the system must be left on to achieve full charge.

SMBus Protocol: Read Word

Output: unsigned int -- remaining time in minutes

Units:	minutes
Range:	0 to 65,534 minutes
Granularity:	2 minutes or better
Accuracy:	±MaxError() * FullChargeCapacity() / AverageCurrent()
Invalid Data Indication:	65,535 indicates the battery is not being charged

5.1.21. BatteryStatus() (0x16)

Description:

Returns the Smart Battery's status word (flags). Some of the BatteryStatus() flags (REMAINING_CAPACITY_ALARM and REMAINING_TIME_ALARM) are calculated based on either current or power depending on the setting of the BatteryMode()'s CAPACITY_MODE bit. This is important because use of the wrong calculation mode may result in an inaccurate alarm.

Purpose:

The BatteryStatus() function is used by the power management system to get alarm and status bits, as well as error codes from the Smart Battery. This is basically the same information broadcast to both the SMBus Host and the Smart Battery Charger by the AlarmWarning() function except that the AlarmWarning() function sets the Error Code bits all high before sending the data..

SMBus Protocol: Read Word

Output: unsigned int - Status Register with alarm conditions bit mapped as follows:

* * * * * * Alarm Bits * * * * *

0x8000	OVER_CHARGED_ALARM
0x4000	TERMINATE_CHARGE_ALARM
0x2000	reserved
0x1000	OVER_TEMP_ALARM
0x0800	TERMINATE_DISCHARGE_ALARM
0x0400	reserved
0x0200	REMAINING_CAPACITY_ALARM
0x0100	REMAINING_TIME_ALARM

* * * * * * Status Bits * * * * *

0x0080	INITIALIZED
0x0040	DISCHARGING
0x0020	FULLY_CHARGED
0x0010	FULLY_DISCHARGED

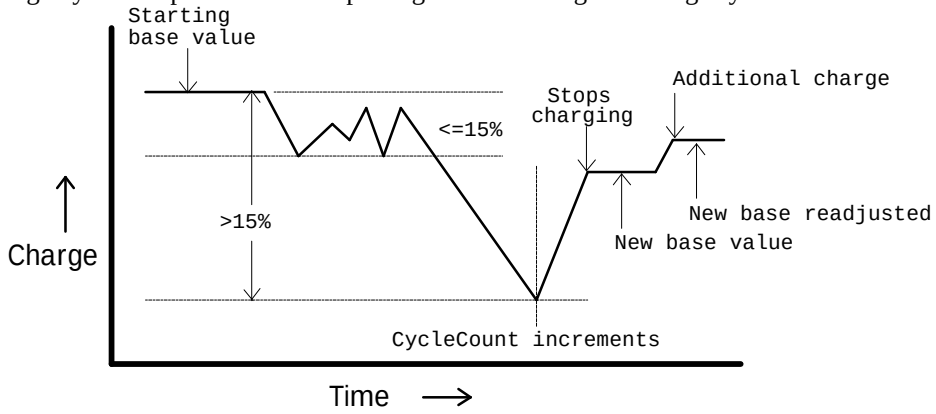
* * * * * * Error Code * * * * *

0x0000-0x000f	reserved for error codes - see Appendix C

5.1.22. CycleCount() (0x17)

Description:

Returns the number of charge/discharge cycles the battery has experienced. A charge/discharge cycle is defined as: starting from a base value equivalent to the battery's highest AbsoluteStateOfCharge() reached after the battery is no longer accepting current before the present charge/discharge cycle is completed and ending when the battery starts accepting current and its AbsoluteStateOfCharge() has decreased by 15% or more from that base value. The CycleCount() will be incremented after each charge/discharge cycle is completed. The relatively large hysteresis prevents false reporting of small charge/discharge cycles.



Purpose:

The CycleCount() function provides a means to determine their battery's wear. It may be used to give advanced warning that the battery is nearing its end of life.

SMBus Protocol: Read Word

Output: unsigned int -- count of charge/discharge cycles the battery has experienced

Units:	cycle
Range:	0 to 65,534 cycles 65,535 indicates battery has experienced 65,535 or more cycles.
Granularity:	1 cycle
Accuracy:	absolute count

5.1.23. DesignCapacity() (0x18)

Description:

Returns the theoretical capacity of a new pack. The DesignCapacity() value is expressed in either current (maH at a C/5 discharge rate) or power (10mwH at a P/5 discharge rate) depending on the setting of the BatteryMode()'s CAPACITY_MODE bit.

Purpose:

The DesignCapacity() function is used by the SMBus Host's power management in conjunction with FullChargeCapacity() to determine battery wear. The power management system may present this information to the user and also adjust its power policy as a result.

SMBus Protocol: Read Word

Output: unsigned int -- battery capacity in maH or 10mwH

	Battery Mode	
	CAPACITY_MODE bit = 0	CAPACITY_MODE bit = 1
Units:	maH	10mwH
Range:	0 to 65,535 maH	0 to 65,535 10mwH
Granularity:	not applicable	
Accuracy:	not applicable	

5.1.24. DesignVoltage() (0x19)

Description:
Returns the theoretical voltage of a new pack (mv).
Purpose:
The DesignVoltage() function can be used to give additional information about a particular Smart Battery's expected terminal voltage.
SMBus Protocol: Read Word
Output: unsigned int -- the battery's designed terminal voltage in mv

Units:	mv
Range:	0 to 65,535 mv
Granularity:	not applicable
Accuracy:	not applicable

5.1.25. SpecificationInfo() (0x1a)

Description:
Returns the version number of the Smart Battery specification the battery pack supports, as well as voltage and current scaling information in a packed unsigned integer. Power scaling is the product of the voltage scaling times the current scaling. The SpecificationInfo is packed in the following fashion: (major version number * 0x10 + minor revision number) + (voltage scaling + current scaling * 0x10) * 0x100.
Purpose:
The SpecificationInfo() function is used by the SMBus Host's power management system to determine what information the Smart Battery can provide.
SMBus Protocol: Read Word
Output: unsigned int -- packed specification number and scaling information

Field	Bits Used	Format	Allowable Values
Revision	0...3	4 bit binary value	0 - 15
Version	4...7	4 bit binary value	1 - 15
VScale	8...11	4 bit binary value	0 - 3 (multiplies voltage by 10 ^ VScale)
IPScale	12...15	4 bit binary value	0 - 3 (multiplies current by 10 ^ IPScale)

Example: The specification version supported by a particular battery is 1.0 and all current readings are to be scaled by a factor of 10. Power readings will be scaled by the voltage factor times the current factor (10^0 * 10 ^ 1) or 10 in this case. SpecificationInfo() will return 4112 (0x1010):

5.1.26. ManufactureDate() (0x1b)

Description:
This function returns the date the cell pack was manufactured in a packed integer. The date is packed in the following fashion: (year-1980) * 512 + month * 32 + day.
Purpose:
The ManufactureDate() provides the system with information that can be used to uniquely identify a particular battery.
SMBus Protocol: Read Word
Output: unsigned int -- packed date of manufacture

Field	Bits Used	Format	Allowable Values
Day	0...4	5 bit binary value	1 - 31 (corresponds to date)
Month	5...8	4 bit binary value	1 - 12 (corresponds to month number)
Year	9...15	7 bit binary value	0 - 127 (corresponds to year biased by 1980)

5.1.27. SerialNumber() (0x1c)

Description:
This function is used to return a serial number. This number when combined with the ManufacturerName(), the DeviceName(), and the ManufactureDate() will uniquely identify the battery (unsigned int).
Purpose:
The SerialNumber() function is used to identify a particular battery. This may be important in systems that are powered by multiple batteries where the system can log information about each battery that it encounters.
SMBus Protocol: Read Word
Output: unsigned int

5.1.28. ManufacturerName() (0x20)

Description:
This function returns a character array containing the battery's manufacturer's name. For example, "MyBattCo" would identify the Smart Battery's manufacturer as MyBattCo.
Purpose:
The ManufacturerName() function returns the name of the Smart Battery's manufacturer. The manufacturer's name can be displayed by the SMBus Host's power management system display as both an identifier and as an advertisement for the manufacturer. The name is also useful as part of the information required to uniquely identify a battery.
SMBus Protocol: Read Block
Output: string -- character string

5.1.29. DeviceName() (0x21)

Description:
This function returns a character string that contains the battery's name. For example, a DeviceName() of "MBC101" would indicate that the battery is a model MBC101.
Purpose:
The DeviceName() function returns the battery's name for display by the SMBus Host's power management system as well as for identification purposes.
SMBus Protocol: Read Block
Output: string -- character string

5.1.30. DeviceChemistry() (0x22)

Description:
This function returns a character string that contains the battery's chemistry. For example, if the DeviceChemistry() function returns "NiMH," the battery pack would contain nickel metal hydride cells.
Purpose:
The DeviceChemistry() function gives cell chemistry information for use by charging systems.
SMBus Protocol: Read Block
Output: string -- character string

Note: The following is a partial list of chemistries and their expected abbreviations. These abbreviations are NOT case sensitive.

Lead Acid	PbAc
Lithium Ion	LION
Nickel Cadmium	NiCd
Nickel Metal Hydride	NiMH
Nickel Zinc	NiZn
Rechargeable Alkaline-Manganese	RAM
Zinc Air	ZnAr

5.1.31. ManufacturerData() (0x23)

Description:

This function allows access to the manufacturer data contained in the battery (data).

Purpose:

The ManufacturerData() function may be used to access the manufacturer's data area. The information and its format are proprietary, but might include items such as: lot codes, number of deep cycles, discharge patterns, deepest discharge, etc. The Smart Battery manufacturer is free to use this data as they see fit.

SMBus Protocol: Read Block

Output: block data - data whose meaning is assigned by the Smart Battery's manufacturer

5.2. Smart Battery or SMB Host-to-Smart Battery Charger Messages

Whenever the BatteryMode() CHARGER_MODE bit is set to zero (default) and the Smart Battery detects the presence of a Smart Battery Charger (level 2 charger - refer to the Smart Battery Charger Specification), the Smart Battery will send the ChargingCurrent() and ChargingVoltage() values to the Smart Battery Charger. For example, the Smart Battery may detect the presence of a Smart Battery Charger by recognizing a charge current or voltage at the its terminals. The Smart Battery will continue broadcasting these values at whatever interval it deems appropriate, not less than 5 seconds nor greater than 1 minute, in order to maintain correct charging. The Smart Battery may not begin broadcasting ChargingVoltage() and ChargingCurrent() values to the charger for at least 10 seconds after it enters the "on state." See BatteryMode().

If a Smart Charger cannot provide the requested charging voltage and/or current, it can:

- terminate charge
- request a different charging voltage and/or current
- accept what is being supplied

For example, a Smart Battery based on NiMH cells may request a constant current of 2.5 amps. The system's power supply may be limited thus allowing the Smart Charger to provide only 1 amp to the Smart Battery. In this case, the Smart Battery could decide that a lower charging current was OK and allow charging to continue at the lower rate.

5.2.1. ChargingCurrent() (0x14)

Description:
Sends the desired charging rate to the Smart Battery Charger (ma).
Purpose:
The ChargingCurrent() function sets the maximum current that a Smart Battery Charger may deliver to the Smart Battery. In combination with the ChargingVoltage() function and the battery's internal impedance, this function determines the Smart Battery Charger's desired operating point. Together, these functions permit a Smart Battery Charger to dynamically adjust its charging profile (current/voltage) for optimal charge. The Smart Battery can effectively turn off the Smart Battery Charger by returning a value of 0 for this function. Smart Battery Chargers may be operated as a constant voltage source above their maximum regulated current range by returning a ChargingCurrent() value of 65535.

Note1: This is the same value as that listed in 5.3.1 but it is written by the Smart Battery to the Smart Battery Charger.
Note2: The Smart Battery Charger responds to current requests in one of three ways:

- supply the current requested
- supply its programmatic maximum current if the request is greater that its programmatic maximum and less than 65535
- supply its maximum safe current if the request is 65535.

Note3: The battery returns a value based on its desired charge rate plus the system's measured power requirements if any.
SMBus Protocol: Write Word
Output: unsigned int -- maximum charger output current in ma

Units:	ma
Range:	0 to 65,534 ma
Granularity:	0.2% of the DesignCapacity() or better
Accuracy:	not applicable

Invalid Data Indication: 65,535 indicates the Smart Battery Charger should operate as a voltage source outside its maximum regulated current range.

5.2.2. ChargingVoltage() (0x15)

Description:
Sends the desired charging voltage to the Smart Battery Charger (mv).
Purpose:
The ChargingVoltage() function sets the maximum voltage that a Smart Battery Charger may deliver to the Smart Battery. In combination with the ChargingCurrent() function and the battery's internal impedance, this function determines the Smart Battery Charger's desired operating point. Together, these functions permit a Smart Battery Charger to dynamically adjust its charging profile (current/voltage) for optimal charge. The Smart Battery can effectively turn off the Smart Battery Charger by returning a value of 0 for this function. Smart Battery Chargers may be operated as a constant current source above their maximum regulated voltage range by returning a ChargingVoltage() value of 65535.

Note1: This is the same value as that listed in 5.3.2 but it is written by the Smart Battery to the Smart Battery Charger.
Note2: The Smart Battery Charger to responds to the voltage requests in one of three ways:

- supply the voltage requested
- supply its programmatic maximum voltage if the request is greater that its programmatic maximum and less than 65535
- supply its maximum voltage if the request is 65535.

SMBus Protocol: Write Word
Output: unsigned int -- charger output voltage in mv

Units:	mv
Range:	0 to 65,534 mv
Granularity:	0.2% of DesignVoltage() or better
Accuracy:	not applicable

Invalid Data Indication: 65,535 indicates the Smart Battery Charger should operate as a current source outside its maximum regulated voltage range.

5.3. Smart Battery Charger or SMB Host-to-Smart Battery Messages

Whenever the BatteryMode() CHARGER_MODE bit is set to one, the Smart Battery Charger (level 3 charger - refer to the Smart Battery Charger Specification) may poll the battery using these functions to determine the Smart Battery's charging requirements. The Smart Battery Charger may continue requesting these values at whatever interval it deems appropriate, not less than 5 seconds, in order to maintain correct charging. See BatteryMode() CHARGER_MODE bit for more information.

5.3.1. ChargingCurrent() (0x14)

Description:

Returns the Smart Battery's desired charging rate (ma).

Purpose:

The ChargingCurrent() function returns the maximum current that a Smart Battery Charger may deliver to the Smart Battery. In combination with the ChargingVoltage() function and the battery's internal impedance, this function determines the Smart Battery Charger's desired operating point. Together, these functions permit a Smart Battery Charger to dynamically adjust its charging profile (current/voltage) for optimal charge. The Smart Battery can effectively turn off the Smart Battery Charger by returning a value of 0 for this function. Smart Battery Chargers may be operated as a constant voltage source above their maximum regulated current range by returning a ChargingCurrent() value of 65535.

Note1: The Smart Battery Charger is expected to respond to the results of current requests in one of three ways:

- supply the current requested
- supply its programmatic maximum current if the request is greater that its programmatic maximum and less than 65535
- supply its maximum safe current if the request is 65535.

Note2: The Smart Battery returns a value based on its desired charge rate plus the system's measured power requirements if any.

SMBus Protocol: Read Word

Output: unsigned int -- maximum charger output current in ma

Units:	ma
Range:	0 to 65,534 ma
Granularity:	0.2% of the DesignCapacity() or better
Accuracy:	not applicable

Invalid Data Indication: 65,535 indicates the Smart Battery Charger should operate as a voltage source outside its maximum regulated current range.

5.3.2. ChargingVoltage() (0x15)

Description:

Returns the Smart Battery's desired charging voltage (mv).

Purpose:

The ChargingVoltage() function sets the maximum voltage that a Smart Battery Charger may deliver to the Smart Battery. In combination with the ChargingCurrent() function and the battery's internal impedance, this function determines the Smart Battery Charger's desired operating point. Together, these functions permit a Smart Battery Charger to dynamically adjust its charging profile (current/voltage) for optimal charge. The Smart Battery can effectively turn off the Smart Battery Charger by returning a value of 0 for this function. Smart Battery Chargers may be operated as a constant current source above their maximum regulated voltage range by returning a ChargingVoltage() value of 65535.

Note: The Smart Battery Charger is expected to respond to the results of voltage requests in one of three ways:

- supply the voltage requested
- supply its programmatic maximum voltage if the request is greater that its programmatic maximum and less than 65535
- supply its maximum voltage if the request is 65535.

SMBus Protocol: Read Word

Output: unsigned int -- charger output voltage in mv

Units:	mv
Range:	0 to 65,534 mv
Granularity:	0.2% of the DesignVoltage() or better
Accuracy:	not applicable

Invalid Data Indication: 65,535 indicates the Smart Battery Charger should operate as a current source outside its maximum regulated voltage range.

5.4. Smart Battery Critical Messages

Whenever the Smart Battery detects a critical condition, it becomes a bus master and sends AlarmWarning() messages to both the Smart Battery Charger and the SMBus Host, as appropriate, notifying them of the critical condition(s). The message sent by the AlarmWarning() function is similar to the message returned by the BatteryStatus() function. The Smart Battery will continue broadcasting the AlarmWarning() messages at 10 second intervals until the critical condition(s) has been corrected. The Smart Battery may not begin broadcasting AlarmWarning() messages to either the host or charger for at least 10 seconds after it enters the "on state." See Appendix C for the meaning of the individual bits.

5.4.1. AlarmWarning() (0x16)

Description:

This message is sent by the Smart Battery acting as a bus master device to the SMBus Host and/or the Smart Battery Charger to notify them that one or more alarm conditions exist. Alarm indications are encoded as bit fields in the Battery's Status, which is then sent to the SMBus Host and/or Smart Battery Charger by this function. The AlarmWarning() is repeated at 10 second intervals until the condition(s) causing the alarm has been corrected.

Note: The SMBus specification requires that the command code for this function be the same as the Smart Battery's address. All alarm conditions are sent to the SMBus Host but only those alarms relating to charging are sent to the Smart Battery Charger.

Purpose:

The AlarmWarning() will be used by the SMBus Host to notify the user about Alarms generated by the Smart Battery. The SMBus Host's power management system and the Smart Battery Charger are responsible for processing the alarm and taking appropriate action. The Smart Battery Charger will use the information to properly charge the system. For example, if the OVER_TEMP_ALARM bit is set, it is expected that the Smart Battery Charger will cease charging the battery to prevent damage.

SMBus Protocol:

Output: unsigned int - Status Register with alarm conditions bit mapped:

* * * * * * Alarm Bits * * * * *	
0x8000	OVER_CHARGED_ALARM
0x4000	TERMINATE_CHARGE_ALARM
0x2000	reserved
0x1000	OVER_TEMP_ALARM
0x0800	TERMINATE_DISCHARGE_ALARM
0x0400	reserved
0x0200	REMAINING_CAPACITY_ALARM
0x0100	REMAINING_TIME_ALARM
* * * * * * Status Bits * * * * *	
0x0080	INITIALIZED
0x0040	DISCHARGING
0x0020	FULLY_CHARGED
0x0010	FULLY_DISCHARGED
* * * * * * Error Code * * * * *	
0x0000-0x000f	All bits set high prior to AlarmWarning() transmission.

Note: Alarm Bits 0x0200 and 0x0100 cause the AlarmWarning() to be sent **only** to the SMBus Host. All other Alarm Bits cause the AlarmWarning() to be sent to both the SMBus Host and the Smart Battery Charger.

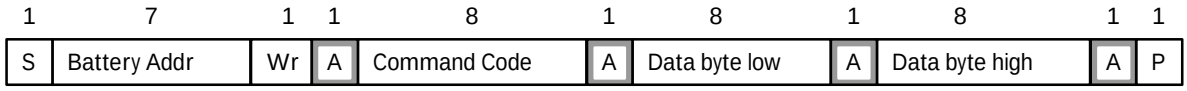
6. Smart Battery Data Protocols

The SMBus Host, acting in the role of an SMBus **master**, uses the Read Word and Write Word protocols to communicate numeric data with the Smart Battery. Non-numeric data, such as the ManufacturerName(), is read using the Read Block protocol. In the case where the Smart Battery needs to inform the SMBus Host about an Alarm condition or to inform the Smart Battery Charger about its desired charging voltage or current, the Smart Battery, acting as an SMBus **master,** uses the Write Word protocol to communicate with the SMBus Host or Smart Battery Charger acting as an SMBus **slave**.

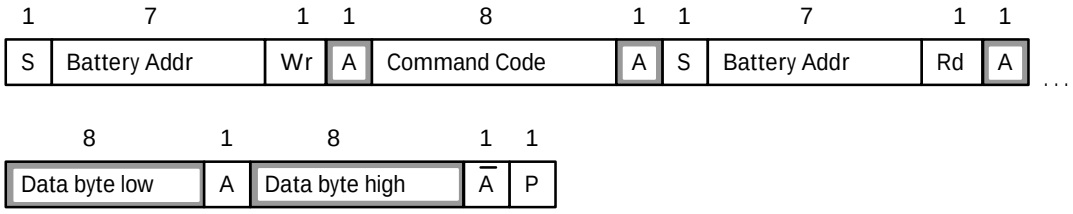
6.1.SMBus Host-to-Smart Battery Message Protocol

The SMBus Host communicates with a Smart Battery using one of four protocols: Read Word, Write Word, Read Block or Write Block. The particular protocol used is determined by the command.

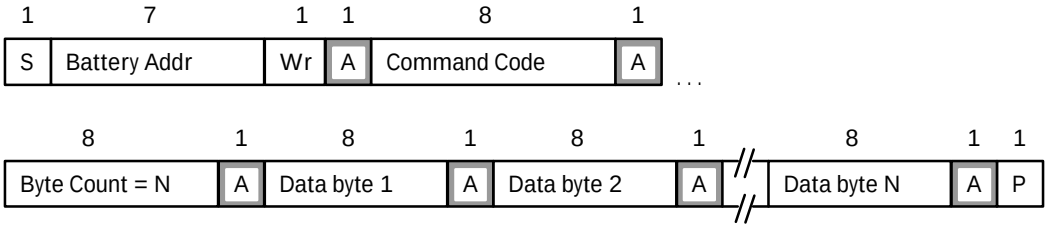
Each of the protocols used is shown below.



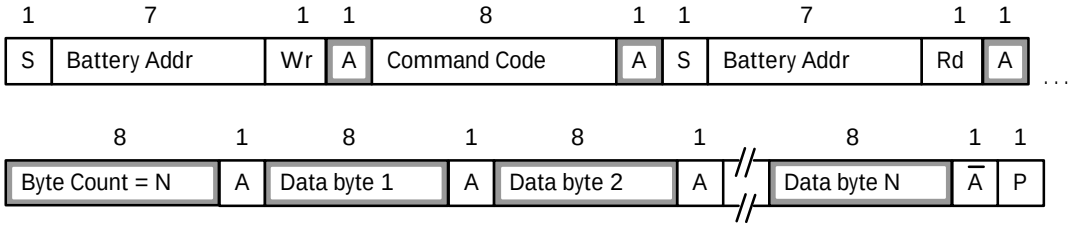
Write Word



Read Word



Block Write



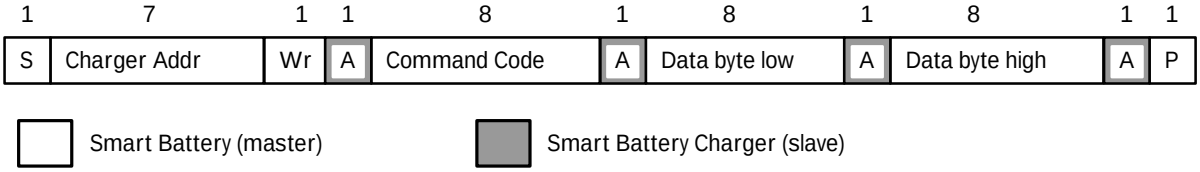
Block Read

SMBus Host (master)

Smart Battery (slave)

6.2.Smart Battery-to-Smart Battery Charger Message Protocol

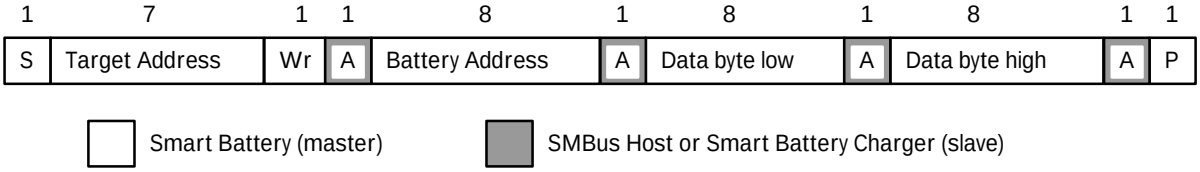
In some cases, the Smart Battery, acting as an SMBus **master** will try to alter the charging characteristics of the Smart Battery Charger, behaving as an SMBus **slave** using the SMBus Write Word protocol. Communication begins with the Smart Battery Charger's address, followed by a Command Code and a two byte value. The Smart Battery Charger attempts to adjust its output to correspond with the request.



Battery Originated Messages for the Charger

6.3.Smart Battery Critical Message Protocol

A Smart Battery to SMBus Host or Smart Battery Charger message is sent using the SMBus Write Word protocol. Communication begins with the SMBus Host's or Smart Battery Charger's address, followed by the Smart Battery's address which replaces the Command Code. The SMBus Host or Smart Battery Charger can now determine that the Smart Battery was the originator of the message and that the following 16 bits are its status.



Alarm Message

Appendix A The command set in tabular form

In the following table, the function name, its access (r,w), data type and command. For a battery to be recognized as a Smart Battery, it must support all the functions described by this specification.

Smart Battery Slave Functions

Function	Code	Access	Data
ManufacturerAccess	0x00	r/w	word
RemainingCapacityAlarm*	0x01	r/w	maH or 10mwH
RemainingTimeAlarm*	0x02	r/w	minutes
BatteryMode	0x03	r/w	bit flags
AtRate	0x04	r/w	ma or 10mw
AtRateTimeToFull	0x05	r	minutes
AtRateTimeToEmpty*	0x06	r	minutes
AtRateOK*	0x07	r	Boolean
Temperature	0x08	r	0.1°K
Voltage	0x09	r	mv
Current	0x0a	r	ma
AverageCurrent	0x0b	r	ma
MaxError	0x0c	r	percent
RelativeStateOfCharge	0x0d	r	percent
AbsoluteStateOfCharge	0x0e	r	percent
RemainingCapacity	0x0f	r	maH or 10mwH
FullChargeCapacity	0x10	r	maH or 10mwH
RunTimeToEmpty*	0x11	r	minutes
AverageTimeToEmpty*	0x12	r	minutes
AverageTimeToFull	0x13	r	minutes
ChargingCurrent	0x14	r	ma
ChargingVoltage	0x15	r	mv
BatteryStatus*	0x16	r	bit flags
CycleCount	0x17	r	count
DesignCapacity	0x18	r	maH or 10mwH
DesignVoltage	0x19	r	mv
SpecificationInfo	0x1a	r	unsigned int
ManufactureDate	0x1b	r	unsigned int
SerialNumber	0x1c	r	number
reserved	0x1d - 0x1f		
ManufacturerName	0x20	r	string
DeviceName	0x21	r	string
DeviceChemistry	0x22	r	string
ManufacturerData	0x23	r	data

* Value affected by the BatteryMode(), CAPACITY_MODE bit setting.

Notes:

- All unused function codes are reserved (0x1d - 0x1f, 0x24 ... 0xff)
- The upper two bits of all command codes are specifically reserved for future use to optionally address multiple batteries.

Smart Battery Master Functions

Function	Code	Access	Data
ChargingCurrent (to Smart Battery Charger)	0x14	w	ma
ChargingVoltage (to Smart Battery Charger)	0x15	w	mv
AlarmWarning (to SMBus Host)	0x16	w	word
AlarmWarning (to Smart Battery Charger)	0x16	w	word

Appendix B Units of Measure

Units describing physical properties

ma	milliamps
maH	milliamp hours @ C/5 rate
AH	amp hours @ C/5 rate
10mw	ten milliwatts
10mwH	ten milliwatt hours @ P/5 rate
mv	millivolts
C	total capacity of the battery in maH, measured at a drain rate of C/5 ma
P	total capacity of the battery in 10mwH, measured at a drain rate of P/5 mw
%	percent
K	degrees kelvin
K/Min	temperature rate of change
mv/Min	voltage rate of change

Units describing atomic data types

char	8 bit value that represents an ASCII character
byte	8 bit value
int	16 bit signed value
unsigned int	16 bit unsigned value
word	unsigned int
Boolean	word, FALSE = 0 and TRUE != FALSE

Units describing aggregate/packed data types

data a block of unsigned bytes (32 byte maximum - see SMBus Specification) where the first byte indicates the number of bytes in the block and is exclusive (e.g. {02,01,02} is a block containing three bytes, the first (02) is the length and the second (01) and third (02) are the data)

string a block of chars (32 byte maximum) where the first byte indicates the number of chars in the block and is exclusive (e.g. {08,'M','y','B','a','t','t','C','o'} is a block containing nine chars, the first (08) is the length of the string and the second through the ninth chars form the string, "MyBattCo")

Miscellaneous

charge	the battery's present charge state as a percentage of full charge

capacity the amount of charge remaining in the battery in maH @ C/5 rate of discharge or in 10mwH @ P/5 rate of discharge

Appendix C Status Bits and Error Codes

Alarm Bits

OVER_CHARGED_ALARM bit is set whenever the Smart Battery detects that it is being charged beyond an end-of-charge indication. This bit will be cleared when the Smart Battery detects that it is no longer being over-charged.

TERMINATE_CHARGE_ALARM bit is set when the Smart Battery detects that one or more of its charging parameters are out of range (e.g. its voltage or current are too high). This bit will be cleared when the parameter falls back in into the allowable range. Failure to correct the problem may result in permanent damage to the battery.

OVER_TEMP_ALARM bit will be set when the Smart Battery detects that its internal temperature is greater than allowed. This bit will be cleared when the internal temperature falls back into the acceptable range.

TERMINATE_DISCHARGE_ALARM bit is set when the Smart Battery determines that it has supplied all the charge it can without being damaged (i.e., continued use will result in permanent capacity loss to the battery). This bit will be cleared when the battery reaches a state-of-charge sufficient for it to once again safely supply power.

REMAINING_CAPACITY_ALARM bit is set when the Smart Battery detects that its RemainingCapacity() is less than that set by the RemainingCapacityAlarm() function. This bit will be cleared when either the value set by the RemainingCapacityAlarm() function is lower than the RemainingCapacity() or when the RemainingCapacity() in increased by charging the Smart Battery.

REMAINING_TIME_ALARM bit is set when the Smart Battery detects that the estimated remaining time at the present discharge rate is less than that set by the RemainingTimeAlarm() function. This bit will be cleared when either the value set by the RemainingTimeAlarm() function is lower than the AverageTimeToEmpty() or when the AverageTimeToEmpty() in increased by charging the Smart Battery.

Status Bits

INITIALIZED bit is set when the Smart Battery is calibrated at time of manufacture. It will be cleared when the battery detects that its calibration data has been lost or altered due to unknown causes.

DISCHARGING bit is set when the Smart Battery determines that it is not being charged. This bit will be cleared when the battery detects that it is being charged.

FULLY_CHARGED bit is set when the Smart Battery determines that has reached a charge termination point. This bit will be cleared when the battery may be charged again.

FULLY_DISCHARGED bit is set when the Smart Battery determines that it has supplied all the charge it can without being damaged (that is, continued use will result in permanent capacity loss to the battery). This bit will be cleared when the RelativeStateOfCharge() is greater than or equal to 20%.

Error Codes

The following table describes the error codes that must be supported by the Smart Battery and the conditions that cause them. For an error code other than OK, an error condition must have been signaled by a not acknowledging the data transfer. See section 4.2.

Error	Code	Access	Description
OK	0x0000	r / w	The Smart Battery processed the function code without detecting any errors.
Busy	0x0001	r / w	The Smart Battery is unable to process the function code at this time.
ReservedCommand	0x0002	r / w	The Smart Battery detected an attempt to read or write to a function code reserved by this version of the specification.
UnsupportedCommand	0x0003	r / w	The Smart Battery does not support this function code which is defined in this version of the specification.
AccessDenied	0x0004	w	The Smart Battery detected an attempt to write to a read only function code.
Overflow/Underflow	0x0005	r/w	The Smart Battery detected a data overflow or under flow.
BadSize	0x0006	w	The Smart Battery detected an attempt to write to a function code with an incorrect size data block.
UnknownError	0x0007	r / w	The Smart Battery detected an unidentifiable error.

###