Smart Battery System Specifications

Smart Battery Data Specification

Revision 1.1
December 11, 1998

Questions and comments regarding this specification may be forwarded to:
Email: battery@sbs-forum.org
Or: questions@sbs-forum.org

For additional information on Smart Battery System Specifications, visit the SBS Implementers Forum (SBS-IF) at:
www.sbs-forum.org

Table of Contents

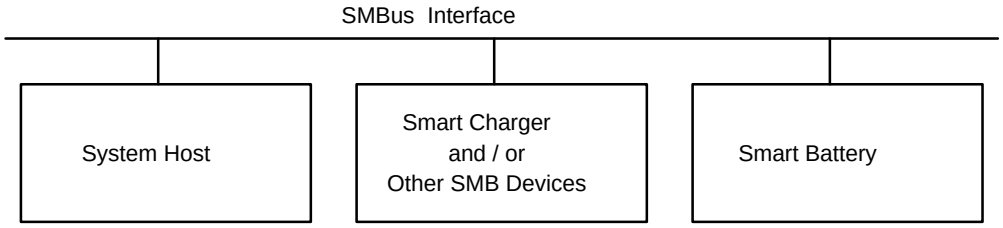
Revision History

Revision Number	Date	Author	Notes
1.0	2/15/95	R Dunstan	Version 1.0 Release
1.1	12/11/98	D Friel	Version 1.1 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

- *Smart Battery Charger Specification*, Revision 1.1, SBS-Implementers Forum, December, 1998
- *Smart Battery Selector Specification*, Revision 1.1, SBS-Implementers Forum, December, 1998
- *Smart Battery System Manager Specification*, Revision 1.1, SBS-Implementers Forum, December, 1998
- *System Management Bus Specification*, Revision 1.1, SBS-Implementers Forum, December, 1998
- *System Management Bus BIOS Interface Specification*, Revision 1.0, February 15, 1995
- *ACPI Specifications*, Version 1.0a, Intel Corporation, Microsoft Corporation, Toshiba Corp., July 1998 (http://www.teleport.com/~acpi)
- *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
- 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 batteries.)

3. Definitions

- **ACPI:** Advanced Configuration and Power Interface. A definition for operating system interface to power management and control functions. This allows more OS control over system hardware and permits more advanced power management control.
- **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, parallel, or series-parallel combination.
- **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.
- **Packet Error Check (PEC):** An additional byte in the SMBus protocols used to check for errors in an SMBus transmission. Refer to the System Management Bus Specification Revision 1.1. A Smart Battery indicates its ability to support PEC with the *version* value in SpecificationInfo() function.

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

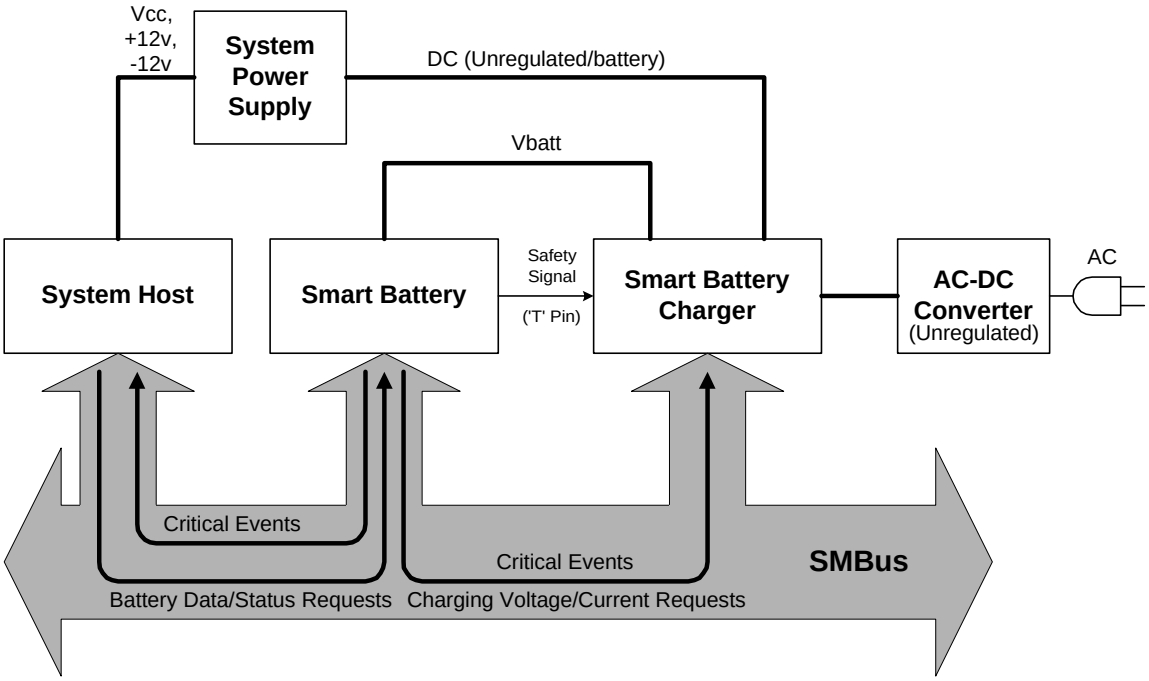
One possible Smart Battery model is a system consisting of a battery, battery charger and a host (notebook computer, video camera, cellular phone, or other portable electronic equipment). Since it is a system, it is important to examine the components and their interactions. (Systems may include additional components such as additional batteries, battery selectors, temperature sensors, and/or other SMBus devices.)

The Smart Battery consists of a collection of cells capable of providing power. Electronics included with the Smart Battery may monitor particular environmental parameters in order to calculate the appropriate data values required by this specification. The electronics need not be inside the Smart Battery if the battery is not removable from the device.

The Smart Battery communicates with other devices (such as the SMBus Host and the Smart Battery Charger) via two separate communication interfaces:

- The first uses the SMBus CLOCK and DATA lines and is the primary communication channel between the Smart Battery and other SMBus devices. The Smart Battery will provide data when requested, send charging information to the Smart Battery Charger, and broadcast critical alarm information when parameters (measured or calculated) exceed predetermined limits within the particular Smart Battery.

- The other required communication interface is the secondary signaling mechanism or ‘Safety Signal’ which was previously described as the ‘T-pin’ on a Smart Battery pack connector. This is a variable resistance output from the Smart Battery which indicates when charging is permitted. It is meant as an alternate signaling method should the SMBus become inoperable. It is primarily used by the Smart Battery Charger to confirm correct charging. (Refer to Section 4.4.4 ‘Safety Signal Hardware Requirements’ and the Smart Battery Charger Specification for additional information.)



The Smart Battery Charger is a charging circuit that provides the Smart Battery with charging current and charging voltage to match the Smart Battery's requested requirements. This allows the battery to control its own charge cycle. Optionally, the Smart Battery Charger 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 charging conditions or temperature conditions which exceed the limits set within the particular Smart Battery.

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.

Although shown as one complete bus, the SMBus may be segmented to only include devices which need to communicate with one another, such as the SMBus Host, the Smart Battery, and the Smart Battery Charger. Separate segments may be used for other SMBus devices. Additional Smart Batteries can be added using the Smart Battery Selector and/or the Smart Battery System Manager to automatically configure the SMBus segments.

4.2. Smart Battery 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 Smart 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. Predictive data can also be calculated based on specified parameters to predict battery performance at charge or discharge currents presently not measured by the battery. (See the AtRate functions.) 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 either Ni- or Li-based batteries, must apply an appropriate current and voltage to the battery for charging. End-of-charge is determined by various methods depending on the specific chemistry and environmental conditions. The difficulty is determining the charge voltage and current for a particular battery and chemistry. Even though the battery pack voltage and perhaps chemistry may be the same, the charging characteristics may not. (For example, not all LiION chemistries require the same charging voltages, even though their pack voltages are the same.)

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: 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, or same chemistry with different voltages.

To improve reliability and safety between the Smart Battery and the Smart Battery Charger, the ‘Safety Signal’ (a secondary signaling mechanism) **must** also be supported by both devices. This mechanism allows an independent communication path to be used to enhance the safety of the charging circuit. (See section 4.4.4. ‘Safety Signal Hardware Requirements’ and the Smart Battery Charger Specification for additional information.)

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.

Such critical notifications may occur from the Smart Battery in two methods: First, using the SMBus AlarmWarning() broadcasts to the SMBus Host or Smart Battery Charger; and second, by using the 'Safety Signal' secondary signaling mechanism.

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.

It is possible to disable some of the broadcasts to the Smart Battery Charger, but this feature must be used with utmost care.

4.3. Error Detection and Signaling

The SMBus Specification provides a simple system for error signaling. The error system is designed to minimize the amount of traffic on the SMBus and system complexity required to communicate with the Smart Battery. Both the Smart Battery and the SMBus host are responsible for detecting and signaling errors. Please also refer to the System Management Bus (SMBus) Specification, Version 1.1, 1998, for additional optional SMBus error detection mechanisms (PEC, packet error checking.)

4.3.1. Error Detection

The Smart Battery is responsible for detecting errors based on the SMBus Specification. When a Smart Battery detects an error condition (such as an overflow, underflow, timeout, unsupported/reserved command, data unavailable, busy or bad size) 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 the BatteryStatus() error bits. (Note: this includes "OK" or "no error detected").

The SMBus Host is also responsible for monitoring SMBus errors that may occur when a Smart Battery is attached to a 'live' SMBus. This could occur during battery insertion or removal from a system. Although the Smart Battery may not interfere with SMBus transactions during this time, the act of inserting or removing a device from the SMBus may cause inadvertent errors. Following the detection of a device insertion or removal, the SMBus Host should generate a START-STOP condition to reset all communication on the bus segment and then re-try any transmissions that may have been in progress. Insertion or removal of a Smart Battery may be detected when the 'Safety Signal' transitions from or to an open-circuit value (>100k.)

When a SMBus device acting as the bus master detects an error, it must attempt to return the bus to the idle state by generating a STOP condition.

4.3.2. Error Signaling

A Smart Battery signals the SMBus Host that it has detected an unrecoverable error by taking advantage of the SMBus 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. Some functions will return invalid data when the calculated value is out of the specified range and then report an OK or Overflow error in the lower nibble of the BatteryStatus register. For example, when reading AverageTimeToFull while the battery is being discharged, the function may return 65535 (indicating a meaningless result) and still report an OK error code.

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, although the 'Safety Signal' can be used to detect the presence of a Smart Battery in a system. Note however that the Smart Battery may choose not to acknowledge any byte following its address if it is **busy** or otherwise unable to respond. If this occurs, the requestor should re-try the data request.

An alternative method of signaling an error is useful when the Smart Battery is returning data to the master in the Read Word protocol. In this case, the battery does not have the opportunity to signal an error by withholding the acknowledge bit, since the master is required by the Read Word protocol to acknowledge the bytes transferred from the battery. If the battery detects an error, it may signal the master by pulling the data or clock line low for longer than the Ttimeout period specified in the SMBus Specification 1.10. The master is then obliged to abort the transaction and return the bus to the idle state.

After each SMBus transaction directed to the Smart Battery device address, the Smart Battery must place the appropriate error code in the lower nibble of the BatteryStatus() register. If the transaction completed successfully, the error codes should be cleared to signify that no error was detected. Timeout and other errors not described by one of the error code types may be signaled with an Unknown Error.

The Smart Battery may signal an error condition by modulating the Safety Signal. This technique uses the Safety Signal as a secondary channel to augment the SMBus communication in the case of critical conditions. The Safety Signal may be forced to a value that prohibits charging when continued charging is unsafe. The battery may also simulate a removal and reinsertion by momentarily forcing the Safety Signal above 100kΩ.

4.3.3. Error Handling

When the Smart Battery signals an error, the Host may use the BatteryStatus() function to get the error code from the Smart Battery to learn of the nature of the error. In the case where the error code is OK, the SMBus Host may assume that an unrecognized bus error occurred which the Smart Battery did not detect. 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, or the SMBus Host detects an error. When either the host or battery signals an error, the host may elect to reattempt the transaction immediately or at a later time.

4.3.4. Error Timing

There is no insurance that another SMBus master device may interrogate the Smart Battery immediately following a communications transfer between the Host and the Smart Battery. This additional data read would cause the BatteryStatus() error codes to be reset based on this most recent communication transfer.

The safest method to insure that the read of BatteryStatus() corresponds to the most recently read data value is to perform this read of BatteryStatus() immediately after the read of the initial data value. This may be accomplished by issuing a SMBus START condition after the SMBus STOP condition from the previous transmission. (Allowing for the minimum time between STOP and START conditions as required by the SMBus Specification.) Since a bus master is required to check for bus idle time of 50 us, this technique may prevent another SMBus compliant master device from gaining control of the bus.

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 of state changes.

4.4.1. Initial Conditions

When a Smart Battery is first delivered, several values must be preset:

Function (Data Value)	Initial Value	Units
RemainingCapacityAlarm()	10% of DesignCapacity() value	mAh
RemainingTimeAlarm()	10	minutes
BatteryMode()	Bit 15: CAPACITY_MODE=0 Bit 14: CHARGER_MODE=0 Bit 13: ALARM_MODE=0 Bit 9: PRIMARY_BATTERY=0 Bit 8: CHARGE_CONTROLLER_ENABLED=0	bit value
BatteryStatus()	Bit 7: INITIALIZED=1	bit value
CycleCount()	Don't care*	decimal

*Note: CycleCount may be at a small value (typically less than 5) if the battery pack manufacturer or assembler has done some initial testing or conditioning of the pack.

4.4.2. Smart Battery 'On State'

The Smart Battery enters the "On State whenever it detects that the SMBus Clock and Data lines go high. The battery should be active and able to communicate via the SMBus within 1 ms of detecting these SMBus lines going high. The battery may not disrupt traffic on the SMBus, however the physical act of inserting a new device (the battery) onto a live bus may cause an inadvertent communication interruption
The Smart Battery may not begin broadcasting ChargingVoltage(), ChargingCurrent() or AlarmWarning() messages to either the SMBus Host or Smart Battery Charger for at least 10 seconds after entering the "On State."

When the Smart Battery enters the "On State" the following values must be reinitialized:

Function (Data Value)	Initial Value	Units
BatteryMode()	Bit 15: CAPACITY_MODE=0 Bit 14: CHARGER_MODE=0 Bit 13: ALARM MODE=0 Bit 9: PRIMARY_BATTERY=0 Bit 8: CHARGE_CONTROLLER_ENABLED=0	bit value

The Smart Battery defaults to disabling the internal charge controller (if it exists) 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.

4.4.3. Smart Battery 'Off State'

The Smart Battery may enter the "Off State" whenever the SMBus Clock and Data lines both remain low for greater than 2.5 seconds. A Smart Battery may enter this "Off state" in less time, however, in no case can it enter the off state in less than 250 ms. The SMBus lines may go low because the battery is removed from the system; the SMBus 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. Safety Signal Hardware Requirements (Smart Battery Charger Interface)

The Smart Battery **must** provide an additional signal to allow for safe charging. This 'Safety Signal' is also commonly referenced as the 'T-pin' or 'Thermistor' on some Smart Battery hardware connectors. The 'Safety Signal' is an output from the Smart Battery and may be used by a Smart Battery Charger (or other device) to determine if charging of the Smart Battery is permitted. This signal is a variable resistance output as measured between the 'Safety Signal' pin and the negative terminal of the battery. The circuit creating this signal may be very simple for some battery chemistries (such as an actual thermistor, for example, for NiCd and NiMH chemistries) or more flexible for other chemistries, (LiION, for example.)

The Smart Battery Charger's capabilities are altered by the value of the Safety Signal. As a required safety feature, the charger must NOT charge a battery when it senses the resistance between the Safety Signal pin and ground to be in the range between 425 and 3150 ohms. A NiMH battery which may use a 103AT thermistor as the source of the Safety Signal would enter this range if it got too hot; or the Safety Signal of a Li-ion battery may which use discrete resistors could be set to this range in an emergency condition. The valid ranges of the Safety Signal are summarized below along with the charger's capabilities for the range. (Please also refer to the Smart Battery Charger Specification.)

Safety Signal Resistance, Rss Ohms (Ω)	**Charger Status Bits**	**Description**	**Wake-up Charge*[1]**	**Controlled Charge*[2]**	**Notes**
0 < Rss < 575	RES_UR, RES_HOT	Under-range	allowed for initial time-out period	allowed	Charger can "Wake-up Charge" for time-out period; "Controlled Charge" allowed.
425 < Rss < 3150	RES_HOT	Hot	not allowed	not allowed	Fail-safe charge termination – charger must not supply current
2850 < Rss < 31.5k	(none)	Normal range	allowed indefinitely	allowed	Charger can "Wake-up Charge" indefinitely; "Controlled Charge" allowed.
28.5k < Rss < 105k	RES_COLD	Cold	allowed for initial time-out period	allowed	Charger can "Wake-up Charge" for time-out period only.
Rss > 95k	RES_OR, RES_COLD	Over-range	not allowed	not allowed	Can be used as battery detect; charger does not supply current.

*NOTES: 1) In the table above, "Wake-up Charge" refers to a maximum amount of charge that the Smart Battery may accept prior to re-enabling itself and communicating on the SMBus. This amount of charge (maximum current and maximum time) is defined by the Smart Battery Charger Specification to be 100 mA (maximum) for 140 to 210 seconds. Removal of the Smart Battery from the Smart Battery Charger and re-insertion will reset the 'Wake-Up' charge which may then be repeated.
2) The reference to "Controlled Charge" as used in the table indicates that the Smart Battery Charger is using the Smart Battery's values of ChargingVoltage() and ChargingCurrent() to control the charging conditions. (These may be read from the Smart Battery or received from the Smart Battery via broadcasts.)

4.4.5. Data Polling and Update Requirements

There is no limit to the speed (other than SMBus limits) or rate at which data may be requested from the Smart Battery. Continuous data polling at high rates is permitted and allowed, though not encouraged due to limitations on SMBus bandwidth and availability. The Smart Battery may delay any data request by holding the CLOCK line low for up to 25 ms. This may be done in order to re-calculate the requested data value or to retrieve data from a storage device.

The Smart Battery also has the option to hold the CLOCK line low for up to 35 ms to cause a timeout to abort the transmission request completely. Additionally, the Smart Battery may NACK (Not Acknowledge) any byte following it's own address to abort any transmission.

Devices which continuously poll the Smart Battery at high rate risk missing AlarmWarning and ChargingVoltage/ChargingCurrent broadcasts from the Smart Battery and so should therefore read these values at least once per ten (10) seconds to insure proper notification of these conditions and values.

Data from the Smart Battery should be updated at reasonable rates so as to insure useful information to a power management system. Long delays between actual changes in battery parameters and the updated values should be avoided. Generally, data values should be updated whenever battery parameters change, depending on the operational mode of the battery (active or at rest) and these updates should occur within five (5) seconds of the associated parameter changes.

Refer to the Smart Battery Data functions for Current, AverageCurrent, Voltage, and Temperature (Sections 5.1.9 to 5.1.12) for specific measurement integration times and update requirements.

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. The same is true of some capacities represented in terms of DesignCapacity() rather than absolute values.

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). Portions of this specification designated 'optional' are not required for compliance.

5.1. SMBus Host to Smart Battery Messages

5.1.1. ManufacturerAccess() (0x00)

Description:
This function is optional and its meaning is implementation specific. It may be used by a battery manufacturer or silicon supplier to return specific version information, internal calibration information, or some other manufacturer specific function. There is no implied or required use for this function and therefore it may be used for multiple purposes. The only requirement is the data protocol listed below: read word or write word.
Purpose:
The ManufacturerAccess() function's purpose is manufacturer specific. No functional requirement is implied although example uses are mentioned in this text.
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 alarm 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.
(If the ALARM_MODE bit is set in BatteryMode() then the AlarmWarning() message is disabled for a set period of time. See the BatteryMode() function for further information.)
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 capacity (mAh) 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 to indicate a first level near end of discharge state. Since the alarm and the RemainingCapacity() value itself are expressed at C/5 or P/5 discharge rates, the value may not directly correspond to the actual present discharge rate. Although this provides a finely controlled alarm set-point, the RemainingTimeAlarm() and related time functions are better suited to for indicating at which point a system should transition into a 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.
(If the ALARM_MODE bit is set in BatteryMode() then the AlarmWarning() message is disabled for a set period of time. See the BatteryMode() function for further information.)
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. Since the time functions incorporate all aspects of the discharge (rate, temperature, state-of-charge) they are better suited for predicting transition points to suspend or hibernate states. 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 flags minor conditions requiring attention.

Defined capabilities include:

- Internal charge controller supported (INTERNAL_CHARGE_CONTROLLER bit)
- Internal primary battery control supported (PRIMARY_BATTERY_SUPPORT bit)
  Note: These capabilities listed are optional but their indicating flag bits must be supported.

Defined modes include:

- Battery's capacity information is specified to be reported in either mAh or 10 mWh (CAPACITY_MODE bit)
- Whether the ChargingCurrent() and ChargingVoltage() values are to be broadcast to the Smart Battery Charger when the Smart Battery requires charging (CHARGER_MODE bit)
- Internal charge controller enable (CHARGE_CONTROLLER_ENABLED bit) [Optional]
- Internal primary battery control enable (PRIMARY_BATTERY bit) [Optional]

Defined conditions requiring attention include:

- Battery requesting a conditioning cycle (CONDITION_FLAG bit)

Purpose:
To allow configuration of the Smart Battery for particular application requirements. (See individual bit definitions which follow.)
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 Most Significant Byte (MSB) which is read/write and the Least Significant Byte (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	R/W	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. More detailed explanations can be found in the listing following the table below.

Field	Bits Used	Format	Allowable Values
INTERNAL_CHARGE_ CONTROLLER	0	read only bit flag	0 - Function Not Supported 1 - Internal Charge Controller Supported
PRIMARY_BATTERY_ SUPPORT	1	read only bit flag	0 - Function Not Supported 1 - Primary or Secondary Battery Support
Reserved	2-6		Undefined
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	10-12		Undefined
ALARM_MODE	13	r/w bit flag	0 - Enable AlarmWarning broadcasts to Host and Smart Battery Charger (default) 1 - Disable AlarmWarning broadcast to Host and Smart Battery Charger
CHARGER_MODE	14	r/w bit flag	0 - Enable ChargingVoltage and ChargingCurrent broadcasts to Smart Battery Charger (default) 1 - Disable broadcasts of ChargingVoltage and ChargingCurrent to Smart Battery Charger
CAPACITY_MODE	15	r/w bit flag	0 - Report in mA or mAh (default) 1 - Report in 10mW or 10mWh

(Note: 'Reserved' bits are not defined and are intended for use in future revisions of the specification, therefore, their use for other purposes is not allowed.)

Specific Definitions for each bit flag condition are listed below:

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 available for activation and control of the actual internal charger.

The definition of an Internal Charge Controller is a device which accepts power from the battery terminals but may regulate or otherwise control the current and voltage that actually reaches the battery's cells. The INTERNAL_CHARGE_CONTROLLER bit simply indicates the presence of the internal charger while the CHARGE_CONTROLLER_ENABLED bit actually controls the on/off state of this internal charger. (See 'Examples' following this section.)

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 function is supported and the PRIMARY_BATTERY bit will be available for activation and control of this function.

The Primary/Secondary battery feature is used with batteries containing internal discharge control mechanisms to allow multiple batteries to be connected in parallel. The PRIMARY_BATTERY_SUPPORT bit simply indicates the presence of this internal control while the PRIMARY_BATTERY bit actually controls the on/off state of this internal control. (See 'Examples' following this section.)

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 CONDITION_FLAG is the first signal from the Smart Battery that it has limited ability to determine the present state-of-charge. As a result other data values may be less accurate than required by this specification.

(A more serious flag is the INITIALIZED status bit flag found in the BatteryStatus() register. Refer to Section 5.1.21 for the BatteryStatus() register.)

Status Flag	Location	Smart Battery Performance	Action Required
CONDITION_FLAG=1	BatteryMode() Bit 7	Useable, Safe, Reliable, but less accurate	Perform Condition Cycle (see text)
INITIALIZED=0	BatteryStatus() Bit 7	Useable, Safe, but use data with caution (less reliable)	See User Manual

When the CONDITION_FLAG is set, the Smart Battery is still fully functional, reliable, and safe. However, the System Host may represent to the user that a condition cycle should be performed as soon as possible to return the Smart Battery to full accuracy. While the CONDITION_FLAG is set, the Smart Battery Data values should be used with more tolerance.

The condition cycle is pack specific, but typically will consist of a full-charge, full-discharge, and repeated to full-charge of the battery pack. The Smart Battery electronics will clear this flag after it detects that a condition cycle has been completed. Refer to the Smart Battery electronics' supplier documentation for specific conditioning cycle procedures required.

NOTE: Please refer to the INITIALIZED status bit flag in the BatteryStatus() register in Section 5.1.21 for a more detailed definition.

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, indicating that this function is supported. The status of a battery pack's internal charge controller can be determined by reading this bit.

The definition of an Internal Charge Controller is a device which accepts power from the battery terminals but may regulate or otherwise control the current and voltage that actually reaches the battery's cells. The INTERNAL_CHARGE_CONTROLLER bit simply indicates the presence of the internal charger while the CHARGE_CONTROLLER_ENABLED bit actually controls the on/off state of this internal charger. (See 'Examples' following this section.)

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.

The optional Primary/Secondary battery feature is used with batteries containing internal discharge control mechanisms to allow multiple batteries to be connected in parallel. The PRIMARY_BATTERY_SUPPORT bit simply indicates the presence of this internal control while the PRIMARY_BATTERY bit actually controls the on/off state of this internal control. (See 'Examples' following this section.)

ALARM_MODE bit is set to disable the Smart Battery's ability to master the SMBus and send AlarmWarning() messages to the SMBus Host and the Smart Battery Charger. When set, the Smart Battery will NOT master the SMBus and AlarmWarning() messages will NOT be sent to the SMBus Host and the Smart Battery Charger **for a period of no more than 65 seconds and no less than 45 seconds.** When cleared (default), the Smart Battery WILL send the AlarmWarning() messages to the SMBus Host and the Smart Battery Charger any time an alarm condition is detected. (See also Section 5.4 for a more detailed explanation of alarm conditions and operations.)

When the ALARM_MODE bit is set, the system assumes responsibility for **detecting and responding** to Smart Battery alarms by reading the BatteryStatus() to determine if any of the alarm bit flags are set. At a minimum, this requires the system to poll the Smart Battery BatteryStatus() every 10 seconds at all times the SMBus is active. The system is expected to take appropriate action.

The ALARM_MODE bit is automatically cleared by the Smart Battery electronics every 60 seconds so that any accidental activation of this mode will not be persistent. A SMBus Host which does not want the Smart Battery to be a master on the SMBus must therefore continually set this bit at least once per 45 seconds to keep the ALARM_MODE bit set.

The ALARM_MODE bit defaults to a cleared state when the Smart Battery first enters the "On-State" as defined in Section 4.4.2.

The condition of the ALARM_MODE bit does NOT affect the operation or state of the CHARGER_MODE bit which is used to prevent broadcasts of ChargingCurrent() and ChargingVoltage() to the Smart Battery Charger. (See CHARGER_MODE bit flag definition.)

The system must correctly interpret **all** alarms and immediately inhibit charging whenever a charger related alarm is detected. The system may only resume charging only when ALL the charging alarm bits are cleared. A summary of expected actions for each alarm bit (including discharge related alarms) is described in the table below:

AlarmWarning bit actions:

Name	Bit	Action When Set:	Action When Cleared:
OVER CHARGED ALARM	15	System MUST stop charging	Charging may resume if TERMINATE_CHARGE_ALARM and OVER_TEMP_ALARM are cleared and ChargingVoltage and ChargingCurrent values are non-zero
TERMINATE CHARGE ALARM	14	System MUST stop charging	Charging may resume if OVER_CHARGED_ALARM and OVER_TEMP_ALARM are cleared and ChargingVoltage and ChargingCurrent values are non-zero
Reserved	13		Undefined
OVER TEMP ALARM	12	System MUST stop charging	Charging may resume if OVER_CHARGED_ALARM and TERMINATE_CHARGE_ALARM are cleared and ChargingVoltage and ChargingCurrent are non-zero
TERMINATE DISCHARGE ALARM	11	System should stop discharge as soon as possible	System takes no action
Reserved	10		Undefined
REMAINING CAPACITY ALARM	9	System MUST notify the host that an alarm event has occurred	System takes no action
REMAINING TIME ALARM	8	System MUST notify the host that an alarm event has occurred	System takes no action

Note: 'Reserved' bits are not defined and are intended for use in future revisions of the specification, therefore, their use for other purposes is not allowed.

Note that the system MUST poll the Smart Battery every 10 seconds even when the system is suspended or while it is powered with AC connected if the ALARM_MODE bit is set.

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.)

When the CHARGER_MODE bit is set, the system assumes responsibility for **safely charging** the Smart Battery. At a minimum, this requires the system to poll the Smart Battery for ChargingVoltage() and ChargingCurrent() at the same rate the Smart Battery would normally send these charging messages to the Smart Battery Charger (e.g. every 5 seconds to 60 seconds.)

The CHARGER_MODE bit allows a SMBus Host or Smart Battery Charger to disable the Smart Battery's broadcast of the ChargingCurrent() and ChargingVoltage().

The use of CHARGER_MODE does NOT affect the use of ALARM_MODE. If only CHARGER_MODE bit is set, AlarmWarning messages relating to charging will still occur and be broadcast the Smart Battery Charger and SMBus Host. (See ALARM_MODE bit flag definition.)

The CHARGER_MODE bit defaults to a cleared state when the Smart Battery first enters the "On-State" as defined in Section 4.4.2.

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.
After changing the CAPACITY_MODE bit, all related values (such as AtRate()) should be re-written while the new mode is active. This is because changes made to the CAPACITY_MODE bit do not retroactively affect values which may have been previously written in another mode. For example, a value written to AtRate() while the CAPACITY_MODE bit was 0 will cause AtRate calculations to be made using the mAH value. Changing the CAPACITY_MODE bit to 1 will not automatically cause all the AtRate calculations to be re-calculated using the 10mWH equivalent, although this is permitted, it is not required.

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 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()

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()
RemainingTimeAlarm()
BatteryStatus()
Optional: AtRateTimeToFull()

Not all of these two lists of values are expected to be automatically converted when the state of the CAPACITY_MODE bit is changed. For example, if a value of '-100' is written to AtRate() while CAPACITY_MODE=0 this would represent "-100 mA." If the CAPACITY_MODE bit is then set, then reading back the value from AtRate() may return "-100" which would now be "-100 10mW." However, the Smart Battery electronics may internally re-calculate the value and report "-X 10mW" using the appropriate conversion or calculation algorithm to determine "X."

BatteryMode() Examples

Internal Charge Controller

One example of an Internal Charge Controller is a circuit located within the Smart Battery which is capable of regulating and controlling the charge current and voltage delivered to the battery cells. This circuit may detect when the battery cells are 'full' and then shut-off the charge current to the cells. This does not therefore require the use of an external Smart Battery Charger circuit to regulate and control charge although a Smart Battery Charger can still be used to supply the minimum current and voltage needed.

The INTERNAL_CHARGE_CONTROLLER bit in BatteryMode() is set to indicate that this optional feature is supported by the particular Smart Battery. The CHARGE_CONTROLLER_ENABLED bit is used to actually activate the charging control. This is done so the system has control over when the battery is allowed to charge. A Smart Battery may NOT automatically decide to charge itself by self-activating the CHARGE_CONTROLLER_ENABLED bit. The default condition is that the Internal Charge Controller is off (not charging, CHARGE_CONTROLLER_ENABLED=0) when the Smart Battery enters the "On-State."

Primary/Secondary Battery

The optional Primary/Secondary Battery feature may be used by system implementations which allow their outputs to be tied together. For example, a battery pack may have an internal circuit consisting of a diode and a pair of MOSFETs in parallel. When these packs are connected in parallel, they are able to supply power but with a loss across the diode until the MOSFETs are enabled.

The PRIMARY_BATTERY bit can be used to control these MOSFETs so that only one battery pack is the Primary battery at a time. When the Primary is removed, the other batteries in the system are still in parallel through their diode connections so that system power is maintained. The power management system can the select another battery to become the Primary battery and enable it's internal MOSFETs through the PRIMARY_BATTERY bit in BatteryMode().

Other implementations using the Internal Charge Controller and a Primary/Secondary Battery features are possible and acceptable as long as they conform to the minimum requirements of this specification:

1) Support for the feature must be indicated by the appropriate flag bit in BatteryMode() being set. (Either INTERNAL_CHARGE_CONTROLLER bit and/or PRIMARY_BATTERY_SUPPORT bit.)

2) Activation of the features must default to disabled or 'off' and can only be activated using the appropriate control bit in BatteryMode() being set. (CHARGE_CONTROLLER_ENABLED bit and/or PRIMARY_BATTERY bit.)

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. (Configuration of the CAPACITY_MODE bit will alter the calculation of AtRate functions. Changing the state of CAPACITY_MODE may require a re-write to the AtRate() function using the appropriate units.)

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. (This does NOT include the present charge or discharge rate and so is calculated independently from the present charge or discharge rate of the battery.)
- When the AtRate value is negative, the AtRateTimeToEmpty() function returns the predicted operating time at the AtRate value of discharge. (This does NOT include the present charge or discharge rate and so is calculated independently from the present charge or discharge rate of the battery.)
- 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 a minimum of 10 seconds. (This DOES include the present discharge rate of the battery and so is calculated differently from the previous 'Time' values listed.)

Timing Note: The Smart Battery may NACK the second function and return a 'Busy' error code in BatteryStatus() while calculating the result based on a new AtRate() value. Conversely, the Smart Battery may use clock stretching during the second function to allow time to calculate the result.

AtRate()	AtRateOK()	AtRateTimeToEmpty()	AtRateTimeToFull()
+X, Charge, Positive (> 0)	TRUE, non-zero	65,535	Time remaining to reach 'full' condition at the '+X' charge rate specified in AtRate().
Zero (0)	TRUE, non-zero	65,535	65,535
-Y, Discharge, Negative (< 0)	See AtRateOK() definition table, below.	Time remaining to reach an 'empty' condition at the '-Y' discharge rate specified in AtRate().	65,535

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	

The AtRate() function requires a two-step operation:
1) A value is written to AtRate() in the appropriate units as determined by the BatteryMode() CAPACITY_MODE bit (current or power).
2) One of the AtRate result functions is read: either AtRateTimeToFull(), AtRateTimeToEmpty(), or AtRateOK().

5.1.6. AtRateTimeToFull() (0x05)

Description:
Returns the predicted remaining time to fully charge the battery at the previously written AtRate value in mA.
Note: This function is only required to return a value when the CAPACITY_MODE bit is cleared and the AtRate() value is written in mA units. If the CAPACITY_MODE bit is set, then AtRateTimeToFull() may return 65535 to indicate over-range and return an error code indicating overflow. Alternately, this function may return a remaining time to full based on a 10 mW value in AtRate().
All other AtRate functions are required to return both values corresponding to the CAPACITY_MODE setting except AtRateTimeToFull(). Support for power capacity (10 mW) reporting in AtRateTimeToFull() is optional.
Purpose:
The AtRateTimeToFull() function is part of a two-function call-set used to determine the predicted remaining charge time at the AtRate value (mA.) It will be used immediately after the SMBus Host sets the AtRate() value. The calculated AtRateTimeToFull() value is independent of the present charge or discharge rate of the battery. Refer to AtRate() for additional usage information.
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 previously written AtRate value. (Result will depend on the setting of CAPACITY_MODE bit.)
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. The calculated AtRateTimeToEmpty() value is independent of the present charge or discharge rate of the battery. Refer to AtRate() for additional usage information.
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 previously written AtRate value of **additional energy** for 10 seconds (Boolean). If the AtRate value is zero or positive, the AtRateOK() function will ALWAYS return true. Result may depend on the setting of CAPACITY_MODE bit.

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() for additional usage information.

SMBus Protocol: Read Word

Output: Boolean -- indicates if the battery can supply the **additional** energy requested

Units:	Boolean
Range:	TRUE (non-zero), FALSE (zero)
Granularity:	not applicable
Accuracy:	not applicable

AtRate()	Current()	AtRateOK()
-Y, Discharge, Negative (< 0)	+X, Positive, Charge (> 0)	TRUE if (-Y + X) > 0 TRUE if (-Y + X) < 0 AND (-Y + X) discharge rate can be supported for 10 seconds or more
-Y, Discharge, Negative (< 0)	-Z, Negative, Discharge (< 0)	TRUE only if (-Y + -Z) discharge rate can be supported for 10 seconds or more

5.1.9. Temperature() (0x08)

Description:
Returns the cell-pack's internal temperature (°K). The actual operational temperature range will be defined at a pack level by a particular manufacturer. 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. (Kelvin units are used to facilitate simple unsigned handling of temperature information and to permit easy conversion to other units.)

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 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() capacity 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 in absolute terms but relative to a specific discharge rate. This information is a numeric indication of remaining charge which can also be represented by the Absolute or Relative StateOfCharge() functions and may be in a better form for use by power management systems. (StateOfCharge() functions return values in percentage format which is a relative representation while RemainingCapacity() function returns a more absolute value defined at a specific discharge value.)

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 which contains Alarm and Status bit 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 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

Note: ‘Reserved’ bits are not defined and are intended for use in future revisions of the specification, therefore, their use for other purposes is not allowed.

The following table summarizes the meanings of the individual bits in the BatteryStatus() word and when each may be expected to be set and cleared. Additional explanations follow the table. Appendix C explains the Error Code definitions.

Name	Bit	Set When:	Action When Set:	Cleared When:
OVER CHARGED ALARM	15	Battery is fully charged and charging is complete	Stop Charging	Charging is no longer detected and condition causing alarm is removed
TERMINATE CHARGE ALARM	14	Charging should be suspended temporarily	Stop Charging (Charging may be re-started when conditions permit.)	Charging is no longer detected and condition causing alarm is removed
Reserved	13	Undefined	Undefined	Undefined
OVER TEMP ALARM	12	Temperature is above pre-set limit	Stop Charging (Charging may be re-started when conditions permit.)	Temperature drops into acceptable range. (Charging may not necessarily be re-started at this point.)
TERMINATE DISCHARGE ALARM	11	Battery capacity is depleted	Stop Discharge As Soon As Possible	Discharge is no longer detected.
Reserved	10	Undefined	Undefined	Undefined
REMAINING CAPACITY ALARM	9	Value of RemainingCapacity() is less than the value of RemainingCapacity Alarm().	(Undefined)	Value of RemainingCapacity Alarm() is zero or is less than the value of RemainingCapacity()
REMAINING TIME ALARM	8	Value of AverageTimeToEmpty() is less than the value of RemainingTimeAlarm().	(Undefined)	Value of RemainingTime Alarm() is zero or is less than the value of AverageTimeToEmpty()
INITIALIZED	7	Battery electronics are first calibrated or configured at time of manufacture.	None required.	Battery electronics have determined that calibration or configuration information has been lost and accuracy is significantly impaired. (User should be notified to be skeptical of battery data.)
DISCHARGING	6	Battery is discharging. This may include self-discharge so it does not always indicate that a discharge current is present.	None required.	Battery is accepting a charge current.
FULLY CHARGED	5	Battery is full and further charge is not required.	Stop Charging.	Battery is no longer considered in a full state. (May or may not request to be charged.)
FULLY DISCHARGED	4	Battery capacity is depleted.	Stop Discharging.	RelativeStateOfCharge() value is greater than 20%
Error Codes	3-0	See definition in Appendix C for listing of codes.		

OVER_CHARGED_ALARM bit is set whenever the Smart Battery detects that it is being charged beyond a Fully Charged state. When this bit is set, charging should be completely stopped as soon as possible. Charging further can result in permanent damage to the battery.
This bit will be cleared when the Smart Battery detects that it is no longer being charged. Charging should not automatically restart.

TERMINATE_CHARGE_ALARM bit is set when charging should be stopped but the Smart Battery may not yet be in a Fully Charged state. Charging is effectively 'suspended,' usually temporarily. Charging may resume when the Smart Battery detects that its charging parameters are back in allowable ranges and ChargingVoltage() and ChargingCurrent() values are both returned to non-zero values. This bit is cleared when the Smart Battery detects that it is no longer being charged.

OVER_TEMP_ALARM bit will be set when the Smart Battery detects that its internal temperature is greater than a preset allowable limit. When this bit is set, charging should be stopped as soon as possible. The Smart Battery may not yet be in a Fully Charged state.
Charging is effectively 'suspended,' usually temporarily. Charging may resume when the Smart Battery detects that its internal temperature is below a preset limit to allow charging again. (This limit may be a different value than what caused the original alarm.)
This bit is cleared when the internal temperature has dropped below an acceptable limit, which may or may not be the original alarm threshold (although charging may not always resume at this point.)

> In all cases where charging is stopped due to an Alarm bit being set, re-start of charging should only occur when ChargingVoltage() and ChargingCurrent() values are returned to non-zero values. Charging should not be automatically re-started solely based on the Alarm bit being cleared.

TERMINATE_DISCHARGE_ALARM bit is set when the Smart Battery determines that it has supplied all the charge it can at the present discharge rate. Discharge should be stopped as soon as possible. This bit will be cleared when the Smart Battery detects that the discharge has stopped or that the rate has lessened. (Note that since this is rate dependent, it may occur at a high discharge rate and disappear when the discharge rate has slowed such that the Smart Battery can continue to be discharged at the lower rate.)

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() is increased by charging the Smart Battery. (NOTE: This Alarm bit can be disabled by writing zero to the RemainingCapacityAlarm() value.)

REMAINING_TIME_ALARM bit is set when the Smart Battery detects that the estimated remaining time at the present discharge rate represented by the value in AverageTimeToEmpty() 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() is increased by charging the Smart Battery or decreasing the discharge rate. (NOTE: This Alarm bit can be disabled by writing zero to the RemainingTimeAlarm() value.)

INITIALIZED bit is SET when the Smart Battery electronics are calibrated or configured for the first time, typically at the time of battery pack assembly or manufacture. It will be cleared when the battery detects that this calibration or configuration data has been lost or altered and a significant degradation in accuracy is possible.

The INITIALIZED status bit is the **second** and more serious signal from the Smart Battery that it has perhaps lost the ability to determine the present state-of-charge. As a result other data values required by this specification may be inaccurate.

(The first signal from the Smart Battery is typically the CONDITION_FLAG found in the BatteryMode() register.)

When the INITIALIZED status bit is NOT set, the Smart Battery Data values should be used in a limited and cautious manner. Data values are still reported and the Smart Battery is still functional and safe although perhaps significantly less accurate.

In contrast, when the CONDITION_FLAG is set, the Smart Battery is still fully functional, reliable, and safe. However, the System Host may represent to the user that a condition cycle should be performed as soon as possible to return the Smart Battery to full accuracy. While the CONDITION_FLAG is set, the Smart Battery Data values should be used with more tolerance.

Status Flag	Location	Smart Battery Performance	Action Required
CONDITION_FLAG=1	BatteryMode() Bit 7	Useable, Safe, Reliable, but less accurate	Perform Condition Cycle
INITIALIZED=0	BatteryStatus() Bit 7	Useable, Safe, but use data with caution (less reliable)	See User Manual

NOTE: Please refer to the CONDITION_FLAG status bit flag in the BatteryMode() register in Section 5.1.4 for more definition.

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 full charge point. This bit will be cleared when the battery may want to be charged again, which is chemistry and manufacturer specific.

FULLY_DISCHARGED bit is set when the Smart Battery determines that it has supplied all the charge it can. Discharge should be stopped soon. This bit will be cleared when the RelativeStateOfCharge() is greater than or equal to 20%. This status bit may be set prior to the 'TERMINATE_DISCHARGE_ALARM' as an early or first level warning of end of battery charge.

5.1.22. CycleCount() (0x17)

Description:
Returns the number of cycles the battery has experienced. A cycle is defined as:

An amount of discharge approximately equal to the value of DesignCapacity.

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. The CycleCount returned value multiplied by the DesignCapacity value can give an approximate “odometer” reading for the total capacity delivered by the Smart Battery.
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 **and capacity** scaling information in a packed unsigned integer. Power scaling is the product of the voltage scaling times the current scaling.

These scaling functions do NOT affect ChargingCurrent() and ChargingVoltage() values.

A Smart Battery Charger cannot be assumed to know this scaling information. (However, a 'Level 3' or 'Host Controlled' Smart Battery Charger may read this value if required for specific applications.)

This value may also indicate a version of SMBus error checking implementation. Refer to the SMBus Specification for actual implementation information.

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. It can be used by Smart Battery Systems where the defined 16-bit data values do not provide enough range for higher power applications.

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	0001 - Version 1.0 and 1.1 all other values reserved
Version	4...7	4 bit binary value	0001 – Version 1.0 0010 – Version 1.1 0011 - Version 1.1 with optional PEC support all other values reserved
VScale	8...11	4 bit binary value	0 - 3 (multiplies voltages* by 10 ^ VScale)
IPScale	12...15	4 bit binary value	0 - 3 (multiplies currents* and capacities by 10 ^ IPScale)

*Note: Except ChargingVoltage() and ChargingCurrent() values.

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.

Due to the 1980 offset, there is no 'Year 2000' issue with the encoding of this function.

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
Lithium Polymer	LiP

Other names may only be assigned by the Smart Battery Data Working Group of the SBS-IF. Please contact the SBS-IF and the SBData WG as listed at the beginning of this document.

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 SMBus Host to Smart Battery Charger Messages

Whenever the Smart Battery wants to be charged and the BatteryMode() CHARGER_MODE bit is zero (default) the Smart Battery will send the ChargingCurrent() and ChargingVoltage() values to the Smart Battery Charger address. 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" as described in Section 4.4 'Smart Battery Characteristics.' See also the definition of BatteryMode().

If a Smart Battery Charger cannot provide the requested charging voltage and/or current, then the Smart Battery 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 Battery 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.

Tolerances of Values: The requested ChargingCurrent() and ChargingVoltage() values must have sufficient tolerances to still allow full charge to occur without impacting the Smart Battery Charger's ability to provide acceptable values.

5.2.1. ChargingCurrent() (0x14)

Description:
Sends the desired charging rate to the Smart Battery Charger (mA).
This represents the maximum current which may be provided by the Smart Battery Charger to permit the Smart Battery to reach a Fully Charged state.
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 this is written (broadcast) 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 than its programmatic maximum and less than 65535
- supply its maximum safe current if the request is 65535.

Note3: The battery returns a value based solely on its desired charge rate.
Note4: ChargingCurrent() is NOT altered by the current scaling factor in SpecificationInfo().
Note5: It is incumbent upon the Smart Battery to be able to withstand considerable variations in the actual charging current supplied if the load varies rapidly during charging.

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).
This represents the maximum voltage which may be provided by the Smart Battery Charger to permit the Smart Battery to reach a Fully Charged state.

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 this is written (broadcast) 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 than its programmatic maximum and less than 65535
- supply its maximum voltage if the request is 65535.

Note3: ChargingVoltage() is NOT altered by the voltage scaling factor in SpecificationInfo().

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 SMBus Host to Smart Battery Messages

If it so desires, the Smart Battery Charger 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, but not less than 5 seconds nor more than 60 seconds, in order to maintain correct charging. (Both ChargingVoltage() and ChargingCurrent() must be read at least once each every 60 seconds in order to continue charging properly.)

When operating in this mode, the Smart Battery Charger may wish to disable the automatic broadcast of the ChargingVoltage() and ChargingCurrent() values by setting the BatteryMode() CHARGER_MODE bit to one. See the definition of the BatteryMode() CHARGER_MODE bit for more information.

In addition to reading ChargingVoltage() and ChargingCurrent() values, if the ALARM_MODE bit is set in the BatteryMode(), the Smart Battery Charger must also continuously read the BatteryStatus() function for any AlarmWarning bits which may occur during charging. See the definition of the BatteryMode() ALARM_MODE bit for more information.

5.3.1. ChargingCurrent() (0x14)

Description:
Returns the Smart Battery's desired charging rate (mA).
This represents the maximum current which may be provided by the Smart Battery Charger and which is required by the Smart Battery to reach a Fully Charged state.
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: This is the same value as that listed in 5.2.1 but it is requested (read) by the Smart Battery Charger from the Smart Battery.
Note2: 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 than its programmatic maximum and less than 65535
- supply its maximum safe current if the request is 65535.

Note3: The Smart Battery returns a value based on its desired charge rate plus the system's measured power requirements if any.
Note4: ChargingCurrent() is NOT altered by the current scaling factor in SpecificationInfo().
Note5: It is incumbent upon the Smart Battery to be able to withstand considerable variations in the actual charging current supplied if the load varies rapidly during charging.
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).
This represents the maximum voltage which may be provided by the Smart Battery Charger to permit the Smart Battery to reach a Fully Charged state.

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 5.2.2 but it is requested (read) by the Smart Battery Charger from the Smart Battery.
Note2: 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 than its programmatic maximum and less than 65535
- supply its maximum voltage if the request is 65535.

Note3: ChargingVoltage() is NOT altered by the voltage scaling factor in SpecificationInfo().

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 the same as the message returned by the BatteryStatus() function, except for the lowest nibble (4 bits). 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 SMBus Host or Smart Battery Charger for at least 10 seconds after it enters the "On State" as described in Section 4.4 'Smart Battery Characteristics.'

If the Smart Battery's default of broadcasting AlarmWarning messages is turned off by setting ALARM_MODE bit to 1, the system assumes responsibility for **detecting and responding** to Smart Battery alarms. At a minimum, this requires the SMBus Host to poll the Smart Battery BatteryStatus() every 10 seconds at all times the SMBus is active. The SMBus Host is expected to take appropriate action(s). For example, the SMBus Host must correctly interpret the charger related alarms and immediately inhibit charging whenever one is detected. The SMBus Host may only allow charging to resume when ALL the charging alarm bits are cleared. A summary of expected actions for each alarm bit can be found in section 5.1.4.

Warning: It is imperative, when the ALARM_MODE bit is set that the SMBus Host polls the Smart Battery's BatteryStatus() and take appropriate actions based on the bits set. The system's overall safety and the integrity of the data will be adversely affected if this is not done properly.

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.

When any of the 'Alarm Bits' listed below is set, the AlarmWarning() value is broadcast to the SMBus Host address, and if appropriate, to the Smart Battery Charger address. The AlarmWarning() is repeated at 10 second intervals to each device until the condition(s) causing the alarm has been corrected.

The state of the 'Status Bits' do not cause the AlarmWarning() to be broadcast.

The SMBus specification requires that the command code for this function be the same as the Smart Battery's address.

All alarm conditions must be sent to the SMBus Host (including alarms concerning charging) 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.

The following table summarizes the meanings of the individual bits in the BatteryStatus() word and when each may be expected to be set and cleared. Additional explanations follow the table. Appendix C explains the Error Code definitions.

Name	Bit	Set When:	Action When Set:	Cleared When:
OVER CHARGED ALARM	15	Battery is fully charged and charging is complete	Stop Charging	Charging is no longer detected
TERMINATE CHARGE ALARM	14	Charging should be suspended temporarily	Stop Charging (Charging may be re-started when conditions permit.)	Charging is no longer detected
Reserved	13	Undefined	Undefined	Undefined
OVER TEMP ALARM	12	Temperature is above pre-set limit	Stop Charging (Charging may be re-started when conditions permit.)	Temperature drops into acceptable range. (Charging may not necessarily be re-started at this point.)
TERMINATE DISCHARGE ALARM	11	Battery capacity is depleted	Stop Discharge As Soon As Possible	Discharge is no longer detected.
Reserved	10	Undefined	Undefined	Undefined
REMAINING CAPACITY ALARM	9	Value of RemainingCapacity() is less than the value of RemainingCapacity Alarm().	(Undefined)	Value of RemainingCapacity Alarm() is zero or is less than the value of RemainingCapacity()
REMAINING TIME ALARM	8	Value of AverageTimeToEmpty() is less than the value of RemainingTimeAlarm().	(Undefined)	Value of RemainingTime Alarm() is zero or is less than the value of AverageTimeToEmpty()
INITIALIZED	7	Battery electronics are first calibrated or configured at time of manufacture.	None required.	Battery electronics have determined that calibration or configuration information has been lost and accuracy is significantly impaired. (User should be notified to be skeptical of battery data.)
DISCHARGING	6	Battery is discharging. This may include self-discharge so it does not always indicated that a discharge current is present.	None required.	Battery is accepting a charge current.
FULLY CHARGED	5	Battery is full and further charge is not required.	Stop Charging.	Battery is no longer considered in a full state. (May or may not request to be charged.)
FULLY DISCHARGED	4	Battery capacity is depleted.	Stop Discharging.	RelativeStateOfCharge() value is greater than 20%
Fixed Nibble	3-0	Always	None	Never

OVER_CHARGED_ALARM bit is set whenever the Smart Battery detects that it is being charged beyond a Fully Charged state. When this bit is set, charging should be completely stopped as soon as possible. Charging further can result in permanent damage to the battery.
This bit will be cleared when the Smart Battery detects that it is no longer being charged. Charging should not automatically restart.

TERMINATE_CHARGE_ALARM bit is set when charging should be stopped but the Smart Battery may not yet be in a Fully Charged state. Charging is effectively 'suspended,' usually temporarily. Charging may resume when the Smart Battery detects that its charging parameters are back in allowable ranges and ChargingVoltage() and ChargingCurrent() values are both returned to non-zero values. This bit is cleared when the Smart Battery detects that it is no longer being charged.

OVER_TEMP_ALARM bit will be set when the Smart Battery detects that its internal temperature is greater than a preset allowable limit. When this bit is set, charging should be stopped as soon as possible. The Smart Battery may not yet be in a Fully Charged state.
Charging is effectively 'suspended,' usually temporarily. Charging may resume when the Smart Battery detects that its internal temperature is below a preset limit to allow charging again. (This limit may be a different value than what caused the original alarm.)
This bit is cleared when the internal temperature has dropped below an acceptable limit, which may or may not be the original alarm threshold (although charging may not always resume at this point.)

> In all cases where charging is stopped due to an Alarm bit being set, re-start of charging should only occur when ChargingVoltage() and ChargingCurrent() values are returned to non-zero values. Charging should not be automatically re-started solely based on the Alarm bit being cleared.

TERMINATE_DISCHARGE_ALARM bit is set when the Smart Battery determines that it has supplied all the charge it can. Discharge should be stopped as soon as possible. This bit will be cleared when the Smart Battery detects that the discharge has stopped.

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() is increased by charging the Smart Battery. (NOTE: This Alarm bit can be disabled by writing zero to the RemainingCapacityAlarm() value.)

REMAINING_TIME_ALARM bit is set when the Smart Battery detects that the estimated remaining time at the present discharge rate represented by the value in AverageTimeToEmpty() 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() is increased by charging the Smart Battery or decreasing the discharge rate. (NOTE: This Alarm bit can be disabled by writing zero to the RemainingTimeAlarm() value.)

INITIALIZED bit is set when the Smart Battery electronics are calibrated or configured for the first time, typically at the time of battery pack assembly or manufacture. It will be cleared when the battery detects that this calibration or configuration data has been lost or altered and a significant degradation in accuracy is possible.

The INITIALIZED status bit is the **second** and more serious signal from the Smart Battery that it has perhaps lost the ability to determine the present state-of-charge. As a result other data values required by this specification may be inaccurate.

(The first signal from the Smart Battery is typically the CONDITION_FLAG found in the BatteryMode() register.)

When the INITIALIZED status bit is NOT SET, the Smart Battery Data values should be used in a limited and cautious manner. Data values are still reported and the Smart Battery is still functional and safe although perhaps significantly less accurate.

In contrast, when the CONDITION_FLAG is set, the Smart Battery is still fully functional, reliable, and safe. However, the System Host may represent to the user that a condition cycle should be performed as soon as possible to return the Smart Battery to full accuracy. While the CONDITION_FLAG is set, the Smart Battery Data values should be used with more tolerance.

Status Flag	Location	Smart Battery Performance	Action Required
CONDITION_FLAG=1	BatteryMode() Bit 7	Useable, Safe, Reliable, but less accurate	Perform Condition Cycle
INITIALIZED=0	BatteryStatus() Bit 7	Useable, Safe, but use data with caution (less reliable)	See User Manual

NOTE: Please refer to the CONDITION_FLAG status bit flag in the BatteryMode() register in Section 5.1.4 for a more detailed definition.

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 full charge point. This bit will be cleared when the battery may want to be charged again, which is chemistry and manufacturer specific.

FULLY_DISCHARGED bit is set when the Smart Battery determines that it has supplied all the charge it can. Discharge should be stopped soon. This bit will be cleared when the RelativeStateOfCharge() is greater than or equal to 20%. This status bit may be set prior to the 'TERMINATE_DISCHARGE_ALARM' as an early or first level warning of end of battery charge.

6. Smart Battery Data Protocols

The SMBus Host, acting in the role of a 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 a SMBus **master,** uses the Write Word protocol to communicate with the SMBus Host or Smart Battery Charger acting as an SMBus **slave**.

Packet Error Checking (PEC) mechanisms are available using the SMBus V1.1 Specification. These optional error checking features require additional bytes in the base protocols but do not interfere with the basic protocols. Please refer to the System Management Bus Specification, Revision 1.1, for more details.

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.

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.

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.

Appendix A. The command set in tabular form

The following table summarizes the Smart Battery command set. It includes the function name, code, access (r,w), and data type. For a battery to be recognized as a Smart Battery, it must support all the functions described by this specification. Included in this table are five *optional* command codes reserved for additional manufacturer-defined functions. In order to preserve compatibility, these *optional* functions may in no way effect the battery's conformance to this specification.

Slave Functions	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
reserved	0x25-0x2e		
OptionalMfgFunction5	0x2f	r/w	data
reserved	0x30-0x3b		
OptionalMfgFunction4 OptionalMfgFunction3-1	0x3c 0x3d-0x3f	r/w r/w	word word

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

Notes: All unused function codes are reserved (0x1d - 0x1f, 0x25 - 0x2e, 0x30 - 0x3b, 0x0 - 0xff)

The upper two bits of all function codes are specifically reserved for future use to optionally address multiple batteries.

Smart Battery Master Functions

Master Functions	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

Note: The operation of these functions may be affected by the CHARGER_MODE and ALARM_MODE control bits located in the BatteryMode function.

Appendix B. Units of Measure

Units describing physical properties

ms	milliseconds
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. Error Codes

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. The Smart Battery detected an attempt to access an unsupported optional manufacturer function code.
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.

###