POWERSMART

Smart Solutions for Portable Power

Corporate Headquarters:
One Research Drive
Shelton, CT 06484
Phone: 1-203-925-1340

Portable and Wireless 1998

SBS Smart Battery Interface Guidelines

By Dan Friel and Mike Mattera

Abstract

The Smart Battery System (SBS), developed by a 10-member industry group, contains multiple specifications detailing a complete power system for portable devices. A SBS Smart Battery is detailed in the System Management Bus (SMBus) and Smart Battery Data (SBData) Specifications. Information regarding the general operational and interface guidelines of SBS Smart Batteries in a system is found throughout these specifications. This paper addresses the four key areas facing system designers wishing to incorporate SBS Smart Batteries into their devices, whether they are notebook computers or other portable products.

The key areas are related to commonly asked questions regarding SBS Smart Batteries and SMBus communications. In particular, these areas are only briefly discussed or explained in the appropriate SMBus and SBData Specifications but are critical to insuring a fault-tolerant, robust Smart Battery System. These topics address communication faults and error recovery, timing and calculations issues, and data accuracy definitions. These items are brought together here to assist the system designer and to provide additional detail and explanation.

Contents

Author Biographies

Dan Friel is the co-author of many of the SBS Specifications and the co-inventor of Duracell's and PowerSmart's Smart Battery electronics. He is chairman of the SBS Implementers' Forum Charger-Selector-Safety Working Group. Mr. Friel is a founding employee of PowerSmart Inc. and was previously with Duracell's Smart Battery Development Group for four years. He holds a BSEE from Purdue University.

Mike Mattera is a founding member of the SBS Implementers' Forum and has been active in the promotion and advancement of the SBS specifications. After working 18 years for Duracell's New Products and Technology Division, Mr. Mattera is a founding member of PowerSmart Inc. He holds a BS in Marketing from Manhattan College.

SBS Smart Battery Interface Guidelines

System Management Bus Collision Avoidance and Recovery

The SMBus is a derivative of the I^2C or Inter-IC bus developed by Philips Semiconductor. The differences between the SMBus and I^2C are primarily logic level thresholds and similar electrical parameters. The collision avoidance and recovery techniques, however, are identical. The bus is a true multi-master wired-OR system if all the associated components correctly adhere to the specifications. However, in some systems, a multi-master arrangement is not implemented, or not implemented correctly, and there are issues with collision and recovery that must be addressed.

The I^2C Specification details the approach used to prevent collision (arbitration) on a wired-OR system: that is that the bus lines must be monitored by the device attempting to drive the bus. Any output that does not coincide with the monitor input is therefore a potential collision and must follow the rules of arbitration. For example, if a device is attempting to drive the bus with a logic HI yet this device's bus monitor input reads a logic LOW, then some other device is driving the bus and the first device must completely release the bus (both lines.) More detail on device arbitration rules can be found in the Philips Semiconductor publications 'The I^2C-bus and How to Use It.'

Another key point in preventing bus collisions is the ability to determine if the SMBus is idle and available. This can be accomplished in two basic ways: by measuring the time the bus lines are both at logic HI, or by watching for START and STOP conditions (primarily STOP conditions) on the bus itself.

Both methods have issues, which must be considered during system design. First, the HI time for the CLOCK and DATA bus lines is specified at 50 uSec but is not always observed by all system components, particularly microcontrollers which may have interrupt routines. Secondly, if a collision or communications fault occurs, a STOP condition may not be issued.

The advised method is therefore to generate a 'bus reset' whenever a collision is suspected or some other communication fault occurs. The 'bus reset' will not only help avoid additional collisions but will also allow the SMBus and all attached devices to correctly recover. This is also suggested when conditions on the bus are unknown or could be corrupted, such as when the bus is first powered-up, when a new device is attached or removed, or when a bus switch (as through a Smart Battery Selector) has occurred.

A 'bus reset' is simply a START condition followed by a minimum 5 uSec delay followed immediately by a STOP condition and then an idle bus (both lines HI) for 50 uSec minimum. Any device that is already in a communication will accept the START condition as a possible Repeated-START while the associated STOP will then end the communication and return the bus to an idle condition. It is suggested, however, that this only be attempted when it is believed that the bus is already idle but that all devices may not be aware that the bus is idle. This 'bus reset' procedure will insure that all devices see a valid START and STOP condition and that the bus is again idle.

PowerSmart ICs and modules use the presence of a STOP condition to determine if the SMBus is idle. It is suggested that a 'bus reset' procedure be used whenever a Smart Battery is attached or removed from a system.

SMBus and Smart Battery Timeout and Clock Stretching Operation

The Smart Battery Data Specification, originally co-written by Duracell and Intel, details the operation of a Smart Battery which reports data over the SMBus to a Host device requesting the data. The System Management Bus Specification indicates how the data transfer is to occur, which protocols to use, and how errors should be handled.

One particular aspect of the SMBus that is different from the base I^2C specification is the area of Timeouts and Clock Delays. There are three such values listed in the specification, a general Timeout value for devices ($T_{TIMEOUT}$); one specifically for slave devices ($T_{LOW:SEXT}$); and one specifically for master devices ($T_{LOW:MEXT}$). All deal with delays in the timing of the CLOCK signal.

Each of these values has a specific use and meaning for SMBus devices although originally they were developed for the Smart Battery in particular. Due to the nature of the Smart Battery itself and the need to conserve power and perform continuous calculations, these timeout values were created to assist the Smart Battery SMBus device. Other devices with similar constraints may also employ these values without penalty.

The $T_{TIMEOUT}$ value is really a signaling mechanism which is used to cause the requesting device (Host Master) to end the communication transfer and generate a STOP condition. It is commonly used by a Slave device when the Slave has decided it cannot complete the communication request and must end the transfer prematurely. Normally, as per the SMBus and I^2C specifications, the Slave device would simply generate a NACK (Not Acknowledge) at the next opportunity. However, if the Slave was in the process of sending data to the Master, then the Slave does not control the ACK since the ACK is being used by the Master to confirm receipt of the data from the Slave. (This is from the SMBus Read Word and Read Block protocols.) In this case (when the ACK is not available to the Slave) then the Slave may cause a $T_{TIMEOUT}$ condition by holding the CLOCK line LOW for a minimum of 25 mSec and a maximum of 35 mSec. The Master must recognize this $T_{TIMEOUT}$ condition and after the CLOCK line is released by the Slave, the Master must generate a STOP condition to formally end the communication transfer. $T_{TIMEOUT}$ must occur over one complete CLOCK interval, not successive CLOCK intervals.

The $T_{LOW:SEXT}$ time is the maximum time a Slave device may extend or hold the CLOCK line LOW before a $T_{TIMEOUT}$ would occur. This delay is to allow the Slave device time to perform a calculation to obtain the requested data. Typically $T_{LOW:SEXT}$ occurs all within one CLOCK interval and after the last Slave address in a Read transaction, but it is possible to occur over multiple CLOCK intervals and therefore is cumulative.

The $T_{TIMEOUT}$ value and $T_{LOW:SEXT}$ values are intrinsically tied and basically indicate the same timing value. The maximum for $T_{LOW:SEXT}$ (25 mSec) is the minimum for $T_{TIMEOUT}$. The $T_{LOW:SEXT}$ and $T_{TIMEOUT}$ values share a common point but care must be taken not to design device timing too close to these endpoints. A typical minimum for $T_{TIMEOUT}$ is 27 mSec and a typical maximum for $T_{LOW:SEXT}$ is 23 mSec. This provides a few mSec buffer to make recognition of the timing values easier.

The $T_{LOW:MEXT}$ time is identical in purpose to the $T_{LOW:SEXT}$ time but shorter and specifically aimed at Master devices in an effort to minimize bus delays. As with $T_{LOW:SEXT}$, it is the minimum time that a Master may hold the CLOCK line before a Slave may consider the Master to have failed during the communication transfer. In essence, a $T_{LOW:MEXT}$ beyond 10 mSec is similar to a $T_{TIMEOUT}$ for a Slave device. It indicates that the communication transfer is invalid or a fault has occurred and that the Slave should release any control over the SMBus lines and wait for a new transfer.

Smart Battery Data Calculation Latency and Bus Busy Response

The Smart Battery Data (SBData) Specification details the various data values which a Smart Battery is expected to return to a Host device which may request such data points. These include the battery voltage, current, and temperature values but also much more useful information which can be used to provide better system power management, not only for the battery itself but for other components in the system.

In the process of calculating these SBData values, the Smart Battery requires some processing time to perform the requested calculations. Additionally, the Smart Battery needs time to update internal registers with values related to calculations for state-of-charge and similar items. These two requirements are the cause of 'latency' and 'bus busy' factors common with Smart Batteries.

Regarding 'latency,' there are specific limits to the time a Smart Battery may take to report back the data requested. The SMBus Read Word and Read Block protocols are complete communication transfer protocols where the requested and the returned data are contained in the same communication. In order to provide calculation time for the Smart Battery, the delay must be integrated into the actual communications protocols. This is the purpose of CLOCK stretching and the use of the $T_{LOW:SEXT}$ timing parameter. This parameter allows up to 25 mSec for a Slave device to calculate a data value during a Read Word or Read Block transfer. This is commonly referred to as the 'latency' of response of the Smart Battery.

Typically this 'latency' is minimal, requiring only a few mSec to as much as 10 mSec. However some SBData values which operate on Host specified data, such as AtRate functions, may take up to 20 mSec. The AtRateTimeToEmpty is a good example: The Host Master first uses a Write Word protocol to send the Smart Battery an AtRate discharge value in mA or mW. Next, the Host Master requests via a Read Word protocol the Smart Battery's calculated AtRateTimeToEmpty value. The Smart Battery must first use the previously written AtRate value, make any needed conversions for mA or mW, check the present state-of-charge information, calculate a new run time based on the AtRate value, and then report the requested value. This whole process may require quite a few instruction cycles of the processor inside the Smart Battery.

Another fact of Smart Battery operation is the need to update internal registers to compensate for changes in state-of-charge. As the Smart Battery is operating it continually monitors the voltage, current, and temperature of the cells in the particular battery. These measurements are then used with other pre-defined and learned values to calculate an internal state-of-charge value. (Other values are also calculated but this is the simplified example.)

All these internal calculations again require a particular amount of processor time to be performed. The time required typically prevents the internal processor from handling other functions, such as SBData requests over the SMBus, while updating the state-of-charge. This is the cause of the 'bus busy' condition common in Smart Batteries. For some interval of time there is a portion of that time which is used for these internal calculations and therefore SBData requests are not processed.

During these 'bus busy' intervals, the Smart Battery must still Acknowledge (ACK) its own address but it may choose to generate a Not Acknowledge (NACK) at the Command Code or any byte following the first address ACK. Similarly, the Smart Battery may instead decide to generate a Timeout condition by holding the CLOCK line LOW for greater than the $T_{TIMEOUT}$ period of 25 mSec. Either method is acceptable and in both cases the Host Master device is required to generate a valid STOP condition to terminate the request. Since the 'bus busy' interval is periodic, it is best for a Host Master to not make SBData requests at a simple periodic rate.

Smart Battery Data Accuracy and Granularity

Within the Smart Battery Data Specification there are a number of data values which have 'Accuracy' and 'Granularity' specifications associated with the particular data value. These indicate the relative 'goodness' of the data and the minimum which can be expected.

The granularity value given for each of the SBData functions is simply the minimum amount that value should change from one reading to the next. For example, the Current data value returns the amount of current measured by the Smart Battery. If a very slowly increasing current were applied to the Smart Battery, the Current function may return these values: 495 mA, 500 mA, and then 505 mA. The granularity is therefore 5 mA since that is the minimum amount of change between each reading.

The granularity is specified as a function of some fixed value represented elsewhere, such as the original design voltage or design capacity of the Smart Battery. This allows it to be correctly sized for a particular Smart Battery.

Granularity is specified to provide some consistency in the SBData values returned by batteries from different manufacturers. As an example, if granularity were not specified some Smart Battery manufacturers could make very inaccurate current measurements and hide the inaccuracies by only reporting current in 500 mA increments. Without a granularity specification this would still be legal, although obviously not very useful, for a power management system. Similarly, if the granularity value were not specified, the RunTimeToEmpty value could be reported in 30-minute increments and could simply drop from 30 minutes to zero without warning, again not very useful.

As an example of calculating a granularity, the SBData Voltage function specifies 0.2% of the Smart Battery's original pack voltage, which is listed in the SBData DesignVoltage function. In a Smart Battery with a DesignVoltage of 10.8V, the granularity for the Voltage data would be at least:

10800 mV x 0.002 = 21.6 mV (where 0.002 is 0.2%)

Note that all the granularity specifications are minimum values and higher granularity, thus a lower value between increments, is encouraged.

Although granularity specifies the increments in which the data will change, accuracy specifies how close to an actual value the data will really be. For example, in the SBData Specification, Voltage accuracy is specified as a percentage of the fixed design voltage, or +/- 1.0%. Again, for a 10.8V Smart Battery, this means that the value reported will be within +/- 108 mV. Therefore, accuracy defines a boundary and granularity defines the increment of movement which can be differentiated within the boundary.

In most cases where accuracy is critical, the range is -0/+X, where X is the specified accuracy. This is true for many SBData functions such as RunTimeToEmpty and similar functions whereby the value reported already incorporates the accuracy information so that the data is never misrepresented.

Accuracy is specified to prevent inaccurate information transmission from low quality and unreliable Smart Battery packs to end-user devices. The accuracy values specified in the SBData Specification represent a minimum performance standard. All PowerSmart Inc. products meet or exceed SBData granularity requirements and exceed all accuracy values specified.