Smart Battery System Specifications

Smart Battery Selector Specification

Revision 1.0
9/5/96

Please forward questions and comments regarding this specification to:

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

TABLE OF CONTENTS

Revision History

Revision Number	Date	Author	Notes
0.9	4/13/95	R. Dunstan	Initial public release
1.0	9/5/96	M. Fruin / R. Dunstan	Revision 1.0

Smart Battery Selector Specification

1. Introduction

The Smart Battery Selector Specification is an adjunct to the Smart Battery Data Specification, providing a solution for many of the complexities associated with the implementation of multiple-battery systems such as notebook computers. Systems that operate with more than one battery pose a number of challenges for both the system's designer and the system's power management software. Since batteries and AC power can come and go, literally, at the user's whim without regard for the system's power requirements, a Smart Battery Selector must be capable of responding to these events without compromising the integrity of the system's power supply. Additionally, the selector should notify the system's power management software when a change has taken place, such as when a battery is inserted or removed.

This specification describes battery selector options and the interface the selector presents to the host system. The actual selector implementation can range from a single integrated component, to emulation done using a notebook's keyboard controller, or to a microcontroller implementing selector functionality in addition to a Smart Battery Charger. The purpose of this specification is to describe the minimum expected functionality and interface.

1.1. Scope

This document specifies the data set used by a Smart Battery Selector and the minimal functionality that such a device must provide. The actual electrical and mechanical specifications will be described by a battery selector component's manufacturer. This specification is generic with regard to the actual implementation.

1.2. Audience

The audience for this document includes:

- Smart Battery System component manufacturers
- Smart Battery System designers
- Designers of power management systems for Smart Battery powered portable electronic equipment

2. References

- *The I²C-bus and how to use it* (includes the specification), Philips Semiconductors, January 1992
- *System Management Bus Specification, Revision 1.0*, Intel Inc., February, 1995
- *Smart Battery Charger Specification, Revision 1.0,* Duracell/Intel Inc., June, 1996
- *Smart Battery Data Specification, Revision 1.0,* Duracell/Intel Inc., February, 1995
- *System Management Bus Windows Programming Interface, Revision 1.0,* Intel Inc., February, 1995
- *Smart Battery System Windows Programming Interface, Revision 1.0,* Intel Inc., February, 1995
- *System Management Bus BIOS Interface Specification, Revision 1.0*, February, 1995

3. Definitions

- **APM:** Advanced Power Management. A BIOS interface defined to enable system-wide power management control via software.
- **Battery:** One or more cells that are designed to provide electrical power.
- **Cell:** The cell is the smallest unit in a battery. Most batteries consist of several cells connected in a series and/or parallel combination.
- **I²C-bus**: A two-wire bus developed by Philips, used to transport data between low-speed devices.
- **Smart Battery:** A battery equipped with specialized hardware that provides present state and calculated and predicted information to its SMBus Host under software control.
- **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 Battery Selector:** A device that arbitrates between two or more Smart Batteries. It controls the power and SMBus paths between the SMBus Host, a Smart Battery Charger and the Smart Batteries.
- **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 with some measure of intelligence powered by a Smart Battery. It is able to communicate with the Smart Battery and use information provided by the battery.

4. Smart Battery Selector

Why is a Smart Battery Selector necessary? It would seem that the obvious method for adding an additional Smart Battery to any system would be simply to allocate an additional SMBus address for the second battery. Unfortunately, this is not practical; the SMBus address is merely one part of the system, defining the data path for smart battery data transactions between the system host, the charger and the battery itself. Other significant connections are required.

In most multiple-battery systems today, some device or mechanism must be present to arbitrate between the batteries. Additionally, this device must be able to dynamically re-configure the system's power with sufficient speed to prevent any negative side-effects caused by removal of the system's primary battery. A user may neither know nor care which battery is powering the system, but the user does expect that the system will keep on working. For example, in a system where a slot is shared between a battery and a floppy disk drive, the user may simply remove the battery to install the floppy drive. If this example system was powered by the battery in the shared slot, the Smart Battery Selector must be able to switch to the other battery quickly enough that the system's power integrity is not compromised.

The system's power management must also be informed that there has been a change in the SelectorState(). The Smart Battery Selector must have the ability to notify the system (preferably via an interrupt) that a change has occurred so the system can get updated information as required.

The battery selector described in this specification can provide this level of functionality.

4.1. Smart Battery Selector Requirements and Considerations

There are several requirements for a system operating from battery power in a multiple-battery system. If these are not considered, there may be undesired consequences ranging from bad data to potentially dangerous operating conditions.

When operating from either battery or AC power:

- Any request that will result in an invalid selector topology will be ignored. For example, using SelectorState() to request two batteries to power the system at the same time will be rejected.
- SelectorState() will always return the actual state of the selector. Software may use this feature to verify that a previous request was valid.

When operating from battery power:

- The power output of the battery in use must be connected to the system power supply and the power output of the unselected battery must be isolated from the system power supply. This is necessary to prevent potentially high-current conditions that can occur if two or more batteries with different characteristics or states are simply connected together.
- The SMBus connection from the battery in use must be directed to the system host and the SMBus connection from the unselected battery must not interfere with the operation of the battery in use. This ensures the integrity of critical messages between the battery in use and the host. "Critical" messages from the unselected battery are by definition not critical in this context. If there are critical messages from the battery to the charger, they will be sent through the data path linking the charger and the battery.
- The host must be able to communicate with any battery NOT powering the host (unselected) to determine capacity, etc., without interfering with the normal operation of the selected battery. This feature enables displays that show the condition of every battery in the system, not just the active one.

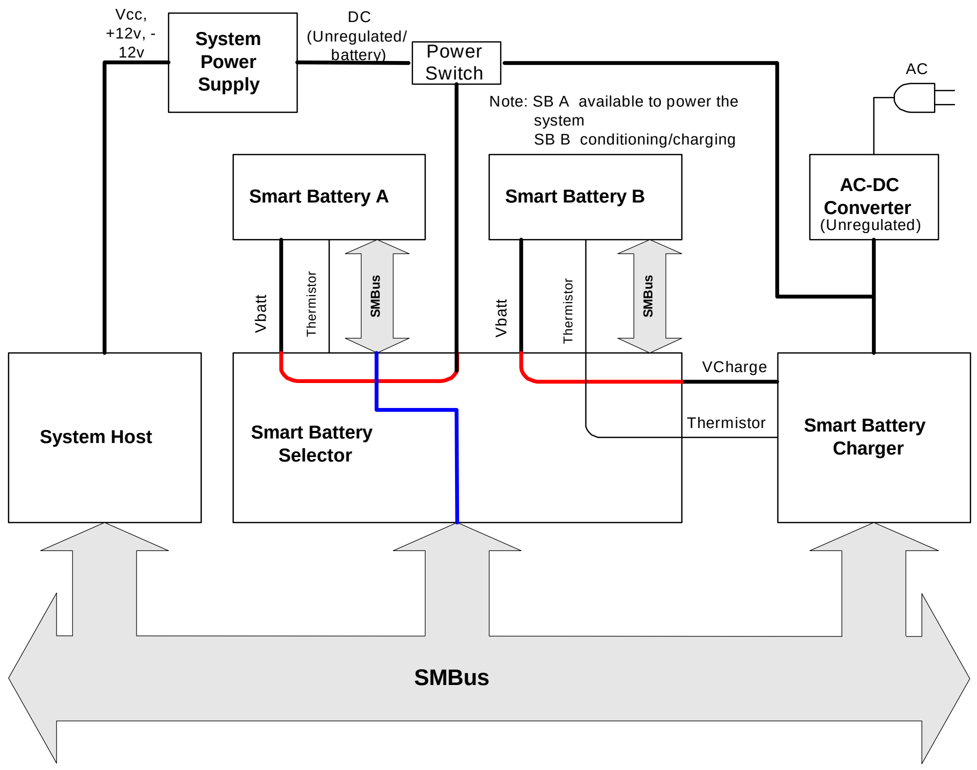
- The selector should provide switch-over functions to autonomously switch between batteries to prevent system failure. In a two-battery system, if the battery in use fails or is removed, the other battery must be selected immediately, without host intervention, and without compromising safety.
- If the selector determines that the configuration has changed (either autonomously or due to user intervention, for example, plugging in an AC wall adapter power source), the host must be notified that a change has occurred. This notification should be independent of the SMBus, for example a state change line, that would cause the host to query the selector's new SelectorState via the SMBus. The SelectorState change line may be connected either to an interrupt or it may be polled. The implementation is at the option of the system's designer. An acceptable alternative would be a selector device that becomes a master device and writes its SelectorState register to the host (SMBus WriteWord to the SMBus Host with the command code set to the selector's address 0x14). In either case, the host is notified by the selector whenever a change occurs.

When operating from an externally supplied power source (AC), there is another set of requirements for a multiple-battery operated system. As in the previous case, there are potentially harmful side-effects if any of the following are ignored while operating from external power:

- The power output from both batteries must be able to be isolated from external power. This is necessary to prevent uncontrolled currents into or out of the batteries.
- The output of the charger must be connected to the appropriate battery. Most (perhaps all) systems will charge the batteries sequentially with only one charger, so the output of that charger must be connected properly.
- The SMBus connection from the battery selected for charging must be connected to the charger's SMBus connection and the SMBus connection from the unselected battery must be isolated from the charger. Because the smart battery uses SMBus commands to properly configure the charger, this data path must match exactly the power path of the charger. It could compromise safety if battery A is connected to the charger power output while battery B is controlling the charge conditions.
- The selected battery's thermistor must be connected to charger. **The thermistor is *the* fail-safe temperature monitor for charging**. Failure to connect the proper thermistor to the charger defeats the fail-safe arrangement.
- The selector must select the proper battery immediately when external power is removed or fails. This must be done autonomously and the SelectorState change mechanism asserted to notify the system host.

Any topology that is in conflict with the above **requirements** has the potential for incorrect or dangerous operation and is therefore unacceptable. Selectors that allow topologies that are in conflict with the above are NOT compliant with the Smart Battery System specifications.

The following diagram illustrates an example of an illegal topology. In this case, Battery B is connected to the charger's power output (the dashed line), but the SMBus (solid line) is connected to Battery A. Such a situation could arise if the system had selected Battery B to be charged, therefore connecting the charger's power output and SMBus to Battery B, and at some later time changed the SMBus connection to Battery A in order to read Battery A's capacity. (Note that in this example implementation the host and charger are always on the same SMBus segment.) If the host then crashes before switching the SMBus connection back to Battery B, Battery A's charging current and voltage requests will be interpreted by the charger as coming from Battery B. Clearly, this dangerous situation must be avoided; for this type of topology the selector must not allow the charger to supply power to one battery while communicating with another. (Of course, the battery pack's internal protection circuitry should avert any dangerous situation, but it should never be relied on as part of the "normal" safety system.)



Example Illegal Selector Topology

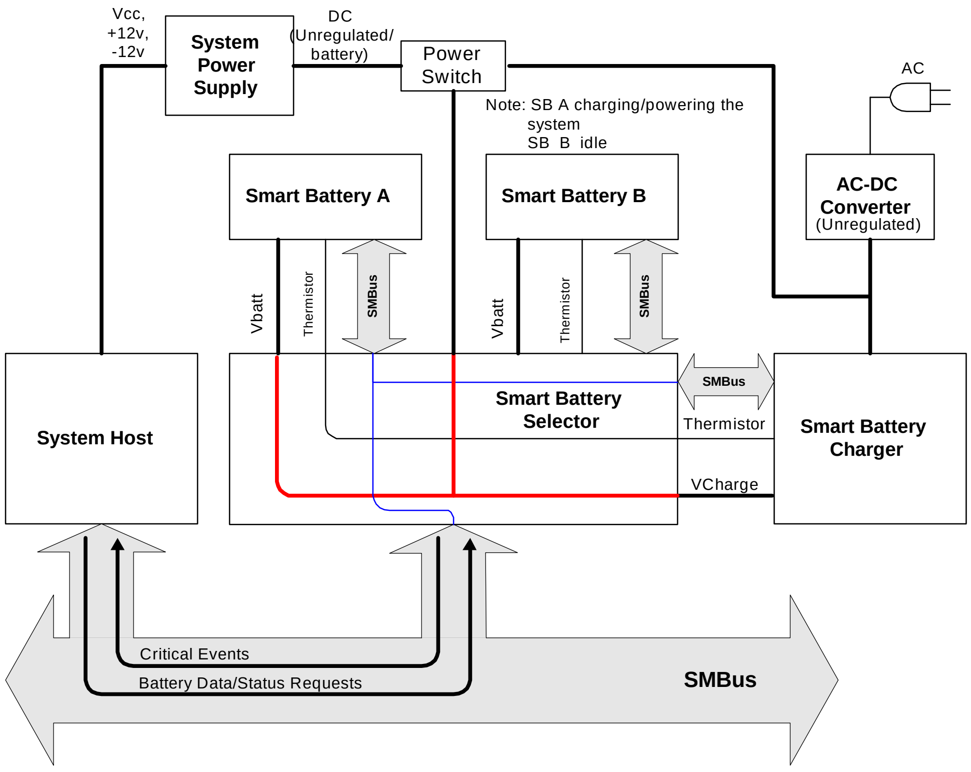
To accomplish these tasks, the selector can be thought of as a device that controls a collection of five data and power switches:

- System input power
- Charger output power
- Battery-to-charger data
- Battery-to-charger thermistor
- Host-to-battery data

Note: See the discussion in section 6.3 for special recommendations for a combined selector and charger component.

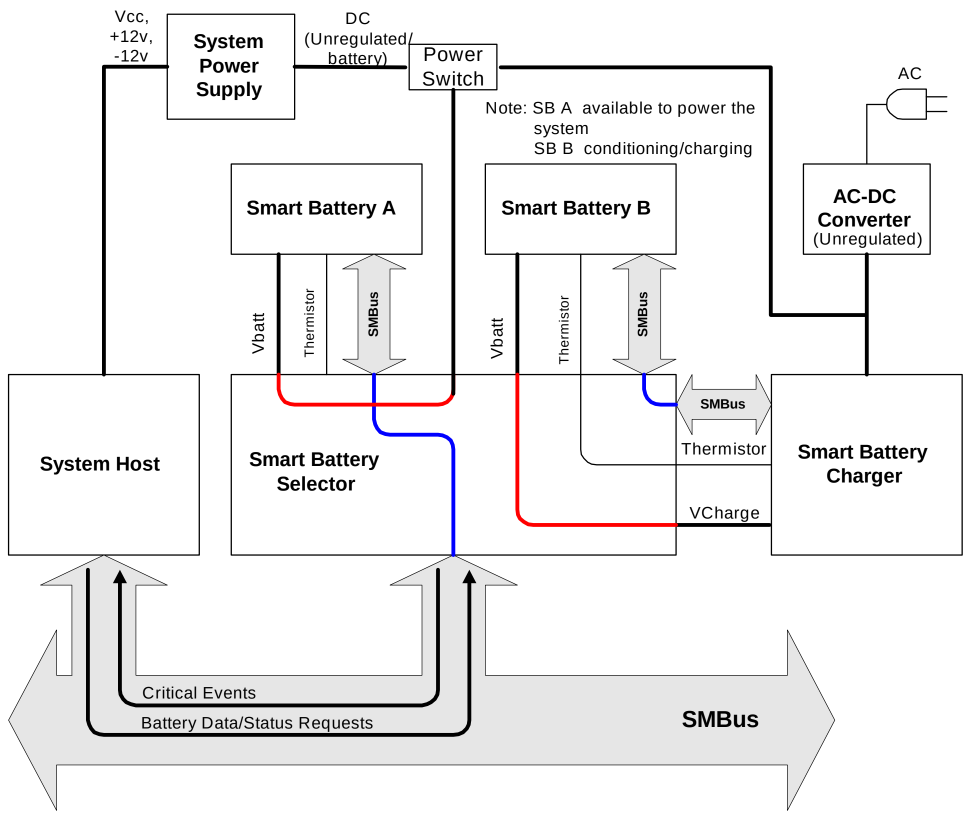
4.2. Smart Battery Selector Model

One possible Smart Battery Selector model is a system consisting of two batteries and smart battery charger in a notebook computer. The diagram below illustrates the typical power and data paths when battery A is acting as the primary battery. This diagram shows a Smart Battery Selector connecting Smart Battery A to both the system and the charger. When AC is present, the Smart Battery can opt to charge itself as required and the system will be powered by the AC source (no power will be drawn from the battery terminals). If AC is not present, the system will be powered by Smart Battery A.



Typical Multiple Smart Battery System

The following diagram presents a variation where Smart Battery A is available to power the system, while Smart Battery B is connected to a charger and being conditioned. Although this is not the most common connection nor necessarily supported by all selectors, it is useful in situations where one battery requires a conditioning cycle while the system must be fully ready to operate.



Typical Multiple Smart Battery System

4.3. Interface Requirements

The software interface consists of the primary read/write control register, an optional read/write control register, and a read-only register. The selector is defined as an SMBus slave-only device, however, it would be possible to construct a device that behaves as a master and sends its SelectorState() to the SMBus host after every change.

4.4. Smart Battery Selector Functional Requirements

The Smart Battery Selector should provide the following services:

- The selector will do a power-on default to connect one battery to power the system and can optionally also connect a battery to the charger.
- The selector will monitor the active battery's terminal voltage and if it falls below a preset minimum, autonomously switch to another battery (if one is present). The selector will update its SelectorState() register to reflect the change and notify the system that its SelectorState has changed. Care should be taken to avoid a situation where the selector "oscillates" between batteries when there are multiple depleted batteries in the system. In this case, it is acceptable for the selector to disconnect all the batteries from the system power supply, effectively shutting down the system.
- The selector will monitor the presence of the active battery and, if it is removed, autonomously switch to another battery (if one is present). The selector will update its SelectorState() register to reflect the change and notify the system that its SelectorState has changed.
- If the system is being powered by AC (POWERED_BY_X set to 0) and AC is removed, the selector will autonomously switch to the next viable battery, update its SelectorState() register to reflect the change, and notify the system that its SelectorState has changed.
- If the optional SelectorPresets()'s USE_NEXT_X register is supported, the selector will give highest priority to the indicated battery on its next autonomous battery switch.
- The charger connection remains until changed by the SMBus Host while the host system is operational. This is true even when the battery connected to the charger is removed or added.
- The battery selector must report whether AC is present or not, and if it changes, the selector will update its SelectorState() register to reflect the change and notify the system that its SelectorState has changed.
- Optionally, it is desirable for there to be some mechanism for the selector to operate in an autonomous manner, independent of high-level control such as that provided by an application or system BIOS, thus allowing the system to charge multiple batteries while the host intelligence is not operational (e.g., when the system is off or suspended). This can be accomplished in many ways, for example: a control bit in one of the Selector's optional OptionalMfgRegister registers, a pin monitoring the system's suspend status signal, or keeping the host alive whenever there is charging power available.

In all cases, the selector's primary purpose is to maintain system power AND minimize interruptions to that power. Its secondary purpose it to inform (or make information available to) the SMBus host about the system's battery SelectorState and changes in that SelectorState.

5. Smart Battery Selector Interface

The following functions are used by the Smart Battery to communicate with a SMBus Host, Smart Battery, 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: Read or Write Word.

Input, Output or Input/Output: A description of the data supplied to or returned by the function.

Whenever the battery selector encounters a valid command with invalid data, it is expected to do nothing and just ignore the data. For example, if an attempt is made to select battery A and B to simultaneously power the system, the selector will just ignore the request. Another example might be a request to connect battery A to the charger and also select battery A to power the system; the selector should also recognize this as an invalid command. This behavior will help to prevent errant commands for setting up conditions that might cause damage to the system, user, or battery.

The following commands describe the interface the system software expects when communicating with a Smart Battery Selector. It is acceptable for the system BIOS, or some other device in the system, to intercept and emulate some or all of the selector commands, provided that the selector interface defined below is maintained to the system software.

5.1. SelectorState (0x01)

Description:

This required function either returns the current state of the selector or allows the state to be altered under programmatic control. The information is broken into four nibbles that report:

- Which battery is communicating with the SMBus Host
- Which power source is powering the system
- Which battery is connected to the smart charger
- Which battery(s) are present.

The selector will have a mechanism to notify the system when there is a change in its state, doing so when:

- A battery is added or removed
- AC power is connected or disconnected
- The selector autonomously switches to another battery (for example, the previous one ran low)

This function provides simultaneous access to four nibble-wide registers of the following types: one for status, two for status and control, and one purely for control. To ensure that a host writing to a specific register nibble does not inadvertently alter the contents of another register nibble, the host must mask register nibbles it doesn't want to modify by writing all 1's (0xF). For example, to set the SMB_X nibble to select the first battery but not change any other values, the host would write 0x1FFF to this function.

Purpose:
Used by the system host to determine the current state of the selector and attached batteries. It also may be used to determine the state of the battery system after the selector notifies the system of a change.

SMBus Protocol: Read or Write Word
Input/Output: word -- bit flags in nibbles mapped as follows:

SMB **(r/w)** Battery connected to System SMBus (comm.)				**POWER_BY** **(r/w)** Battery connected to System PS (power & comm.)				**CHARGE** **(r/w)** Battery connected to Charger (power & comm.)				**PRESENT** **(r)** Battery(s) Present			
D	**C**	**B**	**A**	**D**	**C**	**B**	**A**	**D**	**C**	**B**	**A**	**D**	**C**	**B**	**A**

SMB_X nibble
The read/write SMB_X nibble is used by the SMBus Host either to select which battery to communicate with or to determine which battery it is communicating with. Normally, this nibble selects the same battery as the POWER_BY_X nibble. At most, one bit may be set in this register. When writing a value to the POWER_BY_X nibble, the SMB_X nibble MUST have the same bit set.

For example, an application that displays the remaining capacity of all batteries would use this nibble to select each battery in turn and get its capacity. The bits are defined as follows:

0x8000	SMB_D	Host SMBus is connected to Battery D
0x4000	SMB_C	Host SMBus is connected to Battery C
0x2000	SMB_B	Host SMBus is connected to Battery B
0x1000	SMB_A	Host SMBus is connected to Battery A
0x0000		Host SMBus is not connected to a Battery or to an undetermined battery

Note: At most one bit may be set in this nibble.

POWER_BY_X nibble
The read/write POWER_BY_X nibble is used by the SMBus Host either to select which battery will power the system or to determine which battery is powering the system. It is important to note that any change in the SMB_X nibble is persistent and must be checked to determine that the system is actually communicating with the battery that is supplying power. When selecting which battery will power the system, the POWER_BY_X nibble, the SMB_X nibble MUST have the same bit set. In addition, these bits will be inverted whenever the selector detects the charger is charging a battery, if this feature is supported by the selector as indicated by the CHARGING_INDICATOR bit flag in SelectorInfo().

The bits are defined as follows:

0x0800	POWER_BY_D	Set system to communicate with and be powered by Battery D
0x0400	POWER_BY_C	Set system to communicate with and be powered by Battery C
0x0200	POWER_BY_B	Set system to communicate with and be powered by Battery B
0x0100	POWER_BY_A	Set system to communicate with and be powered by Battery A
0x0000	POWER_BY_AC	Set system to be powered by AC power

Note: If the Selector supports reporting the Charger’s status through this nibble, other valid values are 0x7, 0xB, 0xD, 0xE, and 0xF in addition to 0x0, 0x1, 0x2, 0x4, and 0x8 -- all other bit combinations are not allowed.

When writing to this nibble, the host should write positive logic without concern for whether or not the value should be inverted due to the presence of charging. For example, to set the power_by connection to battery A, even if the charger is charging, the host should write 0x1 to this nibble (and not 0xE).

CHARGE_X nibble
The read/write CHARGE_X nibble is used by the SMBus Host either to select which, if any, battery will be connected to the charger or to determine which, if any, battery is already connected to the charger. In addition, these bits will be inverted whenever AC is present (for example, 0x1 - no AC -> 0xE -AC present). Whether or not AC is present, up to only one bit may be set (i.e., written) to indicate which, if any, battery to charge.

The bits are defined as follows:

0x0080	CHARGE_D	Set to charge Battery D
0x0040	CHARGE_C	Set to charge Battery C
0x0020	CHARGE_B	Set to charge Battery B
0x0010	CHARGE_A	Set to charge Battery A
0x0000		No Battery or undetermined Battery connected to charger

Note: This nibble will be inverted if AC power is present; therefore, valid values are 0x0, 0x1, 0x2, 0x4, 0x8, 0xF, 0xE, 0xD, 0xB, and 0x7 -- all other bit combinations are not allowed.

When writing to this nibble, the host should write positive logic without concern for whether or not the value should be inverted due to the presence of AC power. For example, to set the charger connection to battery 2, even if AC is present, the host should write 0x2 to this nibble (and not 0xD).

PRESENT_X nibble
The read only PRESENT_X nibble is used by the SMBus Host to determine how many and which batteries are present.

The bits are defined as follows:

0x0008	PRESENT_D	Battery D is present
0x0004	PRESENT_C	Battery C is present
0x0002	PRESENT_B	Battery B is present
0x0001	PRESENT_A	Battery A is present

Note: Each bit in the PRESENT_X nibble will be set independently to indicate the presence of a battery.

5.2. SelectorPresets (0x02)

Description:
This optional function allows a system to select the next battery it wants to power the system in the event that the current battery is removed or falls below its cutoff voltage. It also allows the system to select which batteries are OK to use.
Purpose:
Used by the system host to preset expected system behaviors. In a notebook computer where AC is present, the power management system might want to have the battery with the least charge be the next one selected. In that same computer, when AC is not present, the power management system might designate the most fully-charged battery as the next one to use. The OK_TO_USE control allows “bad” batteries to be marked as un-useable. For example, a battery with a protection device in the open-circuit state may be present in the system. In this case, the power management system knows that attempts to extract power from this battery will fail and might cause the system’s power integrity to be compromised.
SMBus Protocol: Read or Write Word

Input/Output: word **--** bit flags in nibbles mapped as follows:

reserved				**reserved**				**USE_NEXT (r/w)** Use this battery next				**OK_TO_USE (r/w)** These battery(s) OK to use			
								D	**C**	**B**	**A**	**D**	**C**	**B**	**A**

USE_NEXT_X nibble
The read/write USE_NEXT_X nibble is used by the SMBus Host either to select which battery it wishes to use next or report which battery it will use next. If no bits are written, the selector will use AND report its default. If the selected battery is not present, the selector will use its default.

The bits are defined as follows:

0x0800	USE_NEXT_D	Set to discharge Battery D next
0x0400	USE_NEXT_C	Set to discharge Battery C next
0x0200	USE_NEXT_B	Set to discharge Battery B next
0x0100	USE_NEXT_A	Set to discharge Battery A next

Note: Zero or one bits of the USE_NEXT_X nibble may be set. If no bit is set or the selected battery is not present, the selector will default to its own (manufacturer specific) “next battery” algorithm.

OK_TO_USE_X nibble
The read/write OK_TO_USE_X nibble is used by the SMBus Host either to select usable batteries or to report which batteries are usable. If a battery is marked NOT OK to use, the selector will NOT allow the battery to be connected to either the system’s power or charger’s power buses. This nibble may also be used to signal that there is NO battery available by marking all the batteries as not OK to use.

The bits are defined as follows:

0x0008	OK_TO_USE_D	Allow the use of Battery D
0x0004	OK_TO_USE_C	Allow the use of Battery C
0x0002	OK_TO_USE_B	Allow the use of Battery B
0x0001	OK_TO_USE_A	Allow the use of Battery A

Note: Each bit in the OK_TO_USE_X nibble may be set independently to indicate that the battery is available for use. The OK_TO_USE_X bit will always be zero when the battery is not present. The OK_TO_USE_X bit will automatically be set whenever the battery is inserted.

5.3. SelectorInfo (0x04)

Description:
The SMBus system host uses this command to determine the capabilities of the selector.
Purpose:
Allows the system host to determine the number of batteries the selector supports as well as the specification revision implemented by the selector.
SMBus Protocol: Read Word
Input/Output: word **--** bit flags in nibbles mapped as follows:

Field	Bits Used	Format	Allowable Values
BATTERIES_ SUPPORTED	0...3	bit flags	BATTERIES_SUPPORTED returns 1's in the bit positions of batteries supported by the selector. For example, a two-battery selector might return 0011 for this nibble, while a four-battery selector would return 1111 for this nibble.
SELECTOR_ REVISION	4...7	encoded nibble	The SELECTOR_REVISION reports the version of the Smart Battery Selector specification supported: 0001 - Version 1.0. All other codes reserved.
CHARGING_ INDICATOR	8	bit flag	The CHARGING_INDICATOR bit indicates whether or not the selector reports the charger's status in the POWER_BY nibble. 1 - Charger status reporting is supported. 0 - The selector does not report the charger's status.
reserved	9..15	undefined	These bits are reserved and will return zero.

Note: It is acceptable for a selector component to expect the system BIOS to intercept and emulate this command rather than implementing it in the actual device itself.

BATTERIES_SUPPORTED nibble
The BATTERIES_SUPPORTED nibble is used by the SMBus Host to determine how many and which batteries the selector can support. This specification is written to allow support for up to four batteries, but due to size or cost constraints a given selector may support less than this number. The bits in this nibble are individually hard-coded by the selector to indicate which battery positions the selector supports.

The bits are defined as follows:

0x0008	PRESENT_D	Battery D is supported by the selector
0x0004	PRESENT_C	Battery C is supported by the selector
0x0002	PRESENT_B	Battery B is supported by the selector
0x0001	PRESENT_A	Battery A is supported by the selector

Note: Each bit in the BATTERIES_SUPPORTED nibble will be set independently to indicate the battery positions supported by the selector.

SELECTOR_REVISION nibble
The SELECTOR_REVISION is an encoded value used to indicate which version of the Smart Battery Selector specification the selector supports. For Revision 1.0 of the Smart Battery Selector specification, the value will be 1.

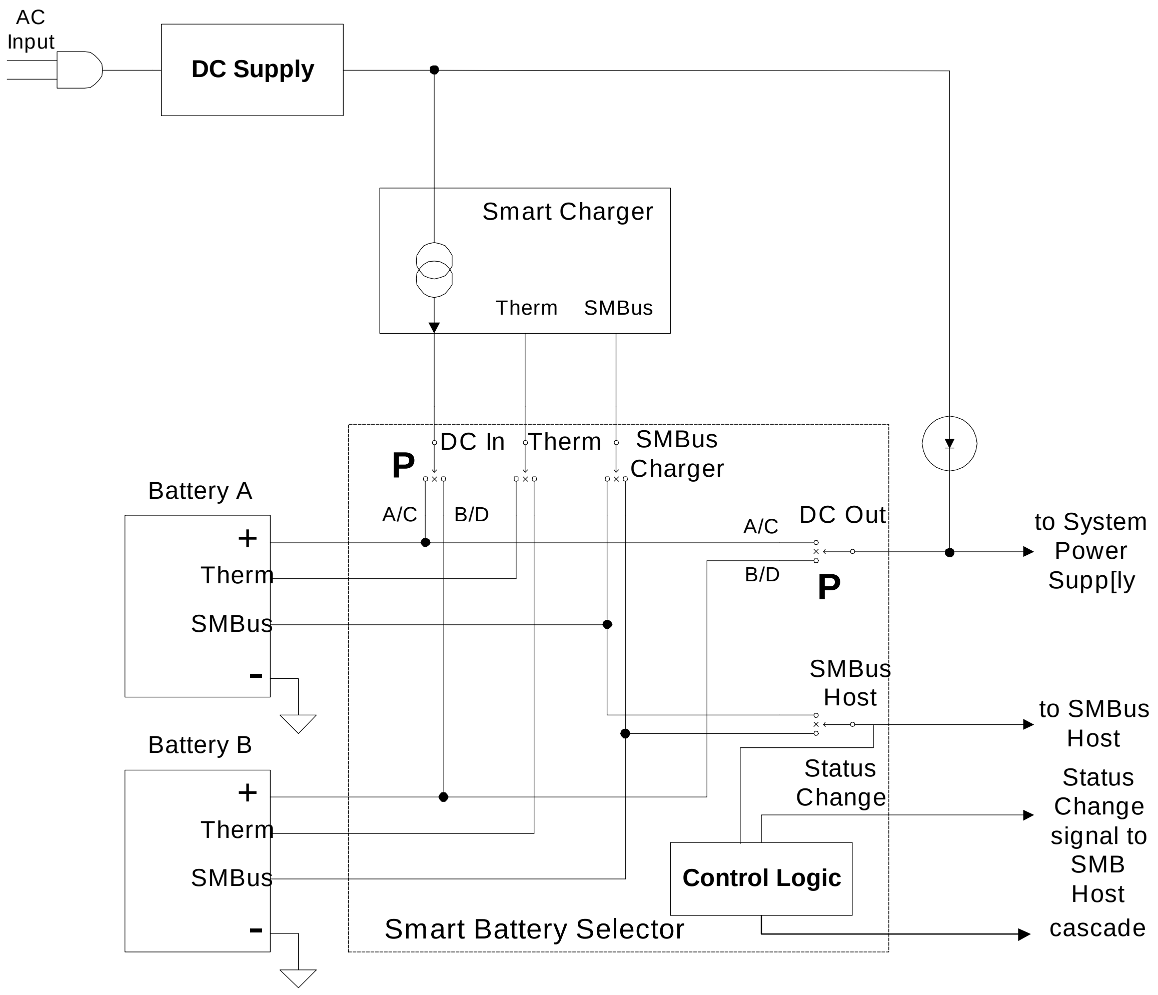
CHARGING_INDICATOR bit flag
The CHARGING_INDICATOR is a bit flag that indicates whether or not the selector reports the charger's status (e.g. the charger is charging a battery or not) in the POWER_BY nibble of SelectorState(). Support of this feature may require a connection between the charger and the selector component. The default is that this feature is supported.

6. Example Implementations

This section is intended to provide a closer look at some possible implementations of a battery selector.

6.1. Sample Battery Selector Diagram

This drawing shows, at a relatively high level, the inner workings of a stand alone battery selector. It is not intended to provide sufficient detail to construct a selector, but rather give a flavor of how it can be done. This particular design will support two batteries, but two devices can be cascaded together to support up to four batteries. The switches labeled "P," are power switches and probably would be external to the device.



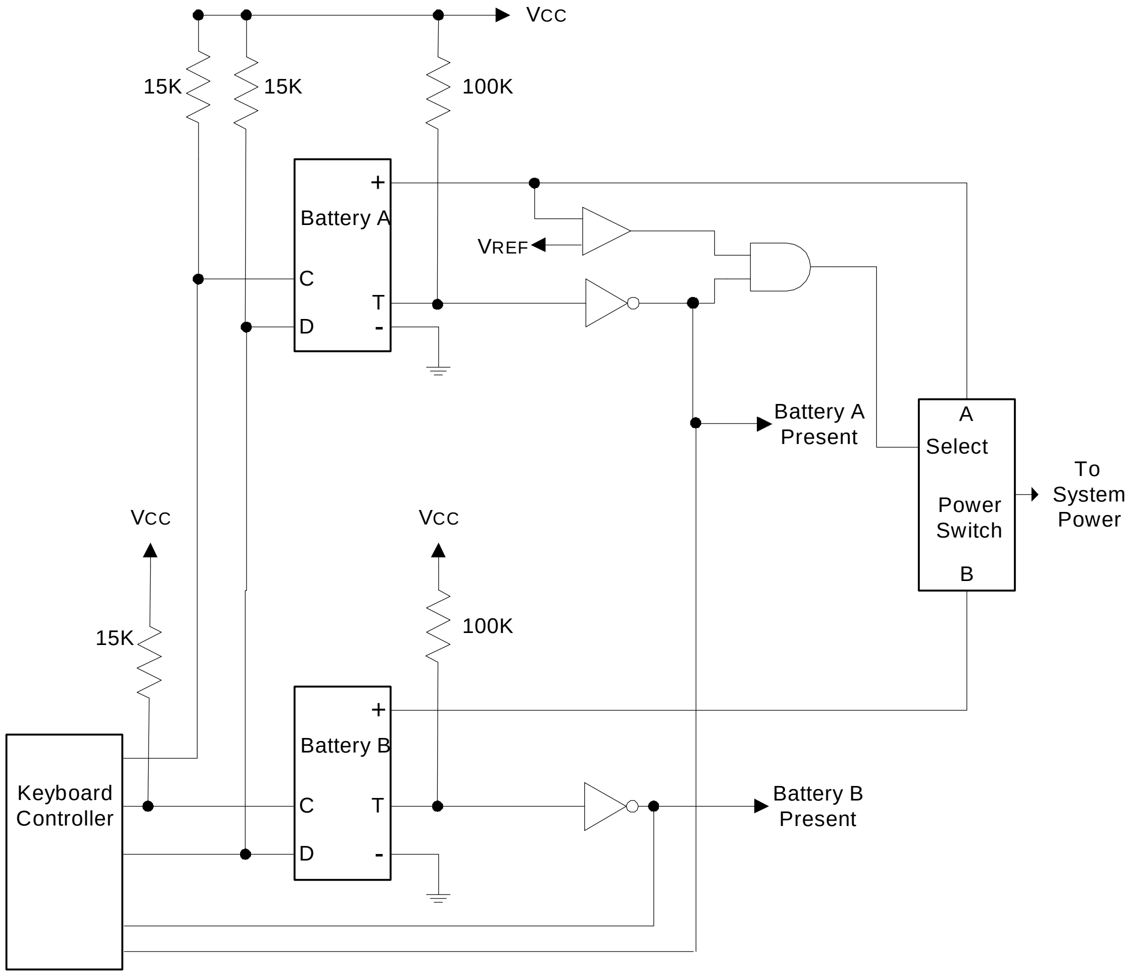
6.2. Battery Selector Emulation

This section illustrates two examples where the keyboard controller emulates the smart battery selector functions: power switching and communications selection. In both of the following examples, a keyboard controller is acting as the system's SMBus host controller.

By intercepting commands directed to a battery selector, the keyboard controller can translate them into an appropriate action emulating a smart battery selector. For example, the keyboard controller in example 6.2.1 can readily report the system's status (for example, which battery(s) is present) or read each battery's characteristics (for example, fuel gauge information). Therefore, to emulate a smart battery selector, all that is needed is a keyboard controller that accepts standard battery selector messages, translates them into appropriate hardware specific actions and returns standard battery selector messages.

6.2.1. Power Switching

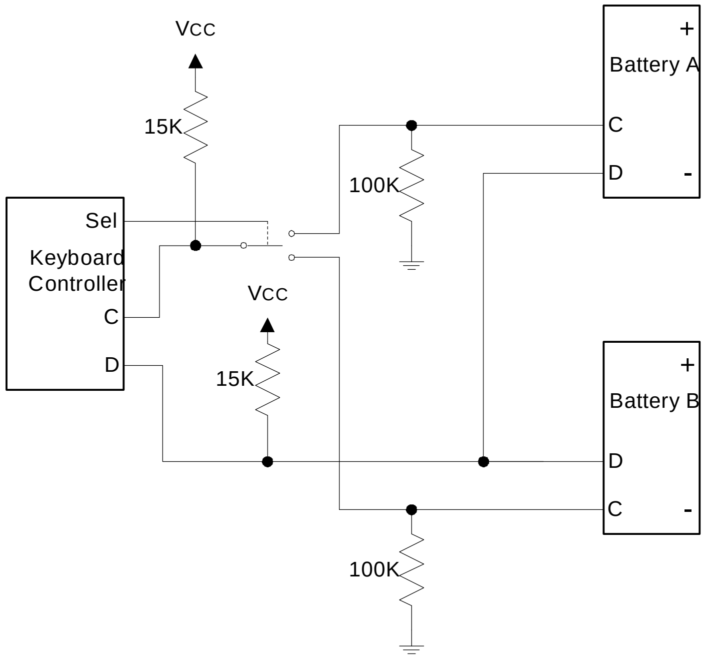
This circuit uses the thermistor pin of the Smart Battery with a logic circuit to detect the presence of a battery. It also detects the output voltage of the battery, so that if the battery is removed or its output falls below the minimum useable voltage battery cutoff voltage (which is equal to VREF), the circuit automatically switches to the other battery.



6.2.2. Battery Communications Selection

There is only one "battery address" in the system, so, for systems with multiple batteries there must be a communications selection process. The following circuit shows one approach: the data lines are tied together and a selector switch, controlled by the keyboard controller, routes the clock to the appropriate battery. With the clock line of the non-selected battery held low, preventing it from responding to signals on the data line, communication with only the selected battery is possible.

This method requires the host (keyboard controller) to handle all battery communications and to periodically communicate with both batteries, since there is no way for the non-selected battery to alert the host that it has encountered a critical condition.



6.3. Combined Charger / Selector Component

Since the Smart Battery Charger and Selector functions are closely related, one possible implementation could merge the operation of these two components into a single device. In this case, the selector commands described in Section 5 are mapped into the charger's address space at command codes 0x21, 0x22, and 0x24 for SelectorState(), SelectorPresets(), and SelectorInfo(), respectively. Merging these two functions guarantees close cooperation between the selector and charger (especially useful for a system designed to charge multiple batteries autonomously without the host's direct control) and may derive cost benefits from reduced component count. Some care should be taken, however, to ensure safe operation in all functional modes when combining the functionality of both components in a single device.

One area of special concern for a combined charger and selector component is its operation when charging a battery. For an implementation where the charger is always on the same SMBus segment as the host (the most likely case), any request by the host to communicate with a battery other than the one being charged will necessarily require disconnecting the charger's SMBus from the battery being charged. In this situation, the charger / selector component must not accept any current or voltage requests from the other battery, and if for some reason the SMBus connection is not restored to the battery being charged, charging must be aborted after the communications timeout period as defined in the Smart Battery Charger Specification (e.g. 140 seconds minimum, 210 seconds maximum).

7. Appendix A The command set in tabular form

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

Smart Battery Selector Functions

Function	Code	Access	Data
reserved	0x00		
SelectorState	0x01	r/w	packed word
(Optional) SelectorPresets	0x02	r/w	packed word
SelectorInfo	0x04	r only	packed word
reserved	0x03, 0x05 - 0x2e		
OptionalMfgFunction5	0x2f	r/w	data
reserved	0x30-0x3b		
OptionalMfgFunction4	0x3c	r/w	word
OptionalMfgFunction3	0x3d	r/w	word
OptionalMfgFunction2	0x3e	r/w	word
OptionalMfgFunction1	0x3f	r/w	word

Notes:

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

###