Reference Design Guide

Smart Battery Gateway

Issued by:	Toradex	Document Type:	Reference Design Guide

Purpose:	This document is a reference design guide for the Smart Battery Gateway

Version:	V1.0

Revision History Date	Version	Remarks
20th June 2012	V1.0	Initial release

1 Smart Battery Gateway Architecture

The idea behind the smart battery gateway is to provide a flexible smart battery interface that is compatible with ACPI compliant operating systems. The gateway interfaces with a standard Award BIOS which provides the host operating system with smart battery support.

The BIOS supports a rich, pre-defined smart battery SM bus compliant communication protocol which the host uses to communicate directly with the smart battery gateway. The smart battery gateway provides two SM bus interfaces; one interface is a dedicated interface with the host controller, and the second interface is used to connect to the smart battery devices which form part of the smart battery solution (e.g. smart battery chargers, smart battery system managers and smart batteries).

The gateway therefore isolates all devices which form the smart battery solution from the host device by physically splitting the SM bus. This has the advantage of increasing robustness of hot plug solutions, when smart battery components may be connected or removed from the SM bus on the fly, whilst providing the flexibility to build a very customized smart battery solution without having to modify the core system BIOS.

The smart battery gateway reference design is built around the Silicon Labs C8051F340 microcontroller. Through customised firmware running on this device, it is possible for system developers to implement smart battery support which is specifically tailored to the requirements of the target application.

A high level architectural overview of the smart battery gateway and associated components is shown in figure 1.

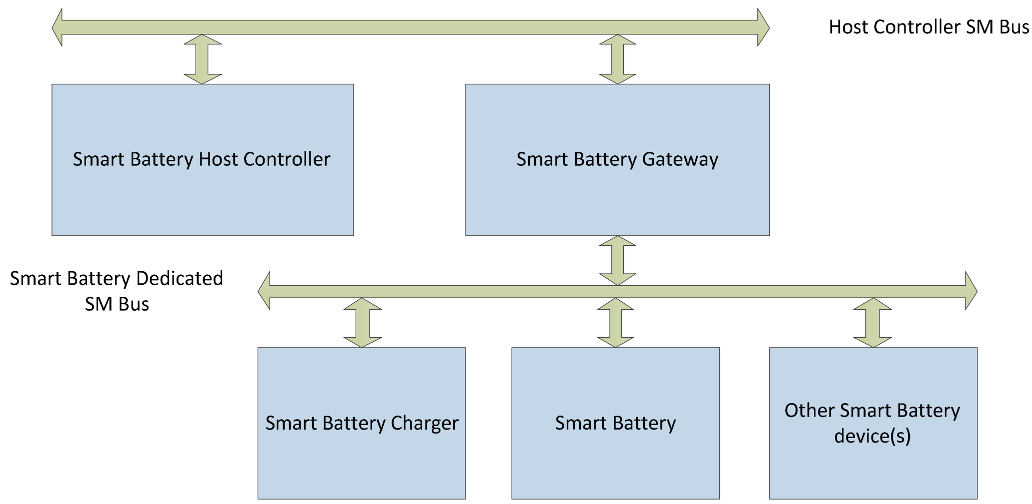

Figure 1: Smart battery architectural overview

2 Host Controller

The host controller is the term used to refer to the host hardware and software platform which interfaces with the smart battery gateway, as shown in figure 2. This consists of a hardware platform (in this case, the Robin Z5xx and Woodpecker Z5xx computer modules), the host operating system (which should be ACPI complaint to support smart battery solutions) and BIOS (Basic Input / Output System).

The host controller requests information relating to the smart battery system status from the smart battery gateway over the primary SM bus interface.

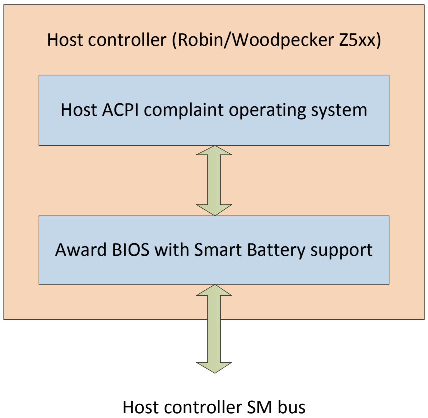

Figure 2: The host controller

The host controller implements a fixed command set which it uses to get smart battery system information from the smart battery gateway and presents it to the host operating system. It follows that the host controller therefore does not need to know about the real architecture of the smart battery system nor the specific device which are used to implement it as it communicates with the smart battery gateway using the same set of commands.

3 Smart Battery Gateway and the Host Controller

The smart battery gateway connects to the host controller through a dedicated SM bus, referred to as the primary SM bus, using the SM bus clock, data and alert signals as shown in figure 3. The smart battery gateway always acts as an SM bus slave, responding to commands issued by the host controller.

The smart battery host controller does not poll the smart battery gateway periodically for battery information. Instead, it relies on the smart battery gateway signalling when it should request updated smart battery information. This is achieved through the smart battery gateway asserting the SM bus alert signal.

It should be noted that the SM bus alert signal as used in this arrangement does not conform to the standard as described in the SM bus specification; that is, no bus arbitration is handled by the host to determine which SM bus devices asserted the SM bus alert signal as in this architecture, this is always the smart battery gateway. However, this is transparent to the rest of the smart battery system when the smart battery is gateway is being used.

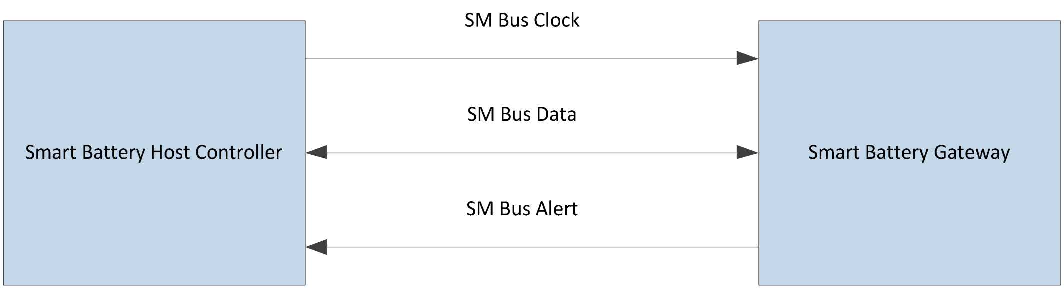

Figure 3: Host controller to smart battery gateway physical interface

As previously mentioned, the host controller implements a fixed command set which it uses to request information relating to the smart battery system status from the smart battery gateway. This command set effectively requires the smart battery gateway to emulate two different smart battery system devices; the smart battery charger and a smart battery.

3.1 Devices and Commands

The host controller implements a subset of smart battery commands which it requires to provide the host operating system with the information in needs to report battery status information. Such commands are issued to the smart battery gateway, which responds with the information required.

The commands are subsets of commands which are required to be supported by either smart batteries or smart battery chargers. The smart battery gateway effectively hides the fact that the host controller is not talking directly to a smart battery or smart battery charger by providing a response to all commands just as the individual smart battery devices would.

The host controller is therefore not aware that is it not communicating directly with either a smart battery or smart battery device, and is instead talking to the smart battery gateway.

The commands which are supported between the host controller and smart battery gateway are as follows:

Smart Battery Device	Device Address	Command	Description
Smart Battery Charger	0x09	0x14	Read charging current
		0x15	Read charging voltage
		0x3F	Read input current
		0x11	Read charger specification info
		0x13	Read charger status
Smart battery	0x0B	0x16	Read battery status
		0x18	Read battery design capacity
		0x10	Read battery full charge capacity
		0x19	Read battery design voltage
		0x0F	Read remaining capacity
		0x09	Read battery voltage
		0x0B	Read average current
		0x04	Write the energy consumption rate used in subsequent calculations
		0x0A	Read battery current

4 Smart Battery Gateway and Smart Battery Devices

As the underlying smart battery architecture and hardware implementation is abstracted from the host controller, the smart battery gateway is free to implement the physical interfaces and command protocols to other smart battery devices in any number of different ways. For example, the smart battery gateway may implement proprietary control signals which allow it to close the charging supply used to charge any smart batteries. Such configurations are very application specific; it should be remembered that the smart battery gateway is essentially a hardware abstraction platform used to interface with a host controller with a fixed interface; hence, the implementation is limited to the resources of the smart battery gateway itself and the requirements of the application.

That said, the reference design implements a software architecture on a defined hardware platform which is suitable for a wide variety of application specific requirements. Whilst the designer may choose to change both hardware and software component parts in their solution, the reference design is intended to provide a good starting point on which to develop such solutions.

4.1 Interfacing to the Smart Battery Gateway

In the reference design, the smart battery gateway makes use of two SM bus interfaces; one dedicated to the host, referred to as the primary SM bus, and the second used for communication with other smart battery SM bus devices, referred to as the secondary SM bus.

As previously described, the interface between the host controller and smart battery gateway is fixed; the host expects the smart battery gateway to support this command set and will use the data returned by the smart battery gateway to report battery status information to the host operating system.

The interface between the smart battery gateway and smart battery devices on the secondary SM bus is not fixed – it is completely implementation specific and depends on the type of smart battery devices which are connected to this bus and how these devices are required to operate.

4.2 Command mapping

In the most simple smart battery system implementation, there is a one-to-one command mapping between the devices addresses and commands issued from the host to the smart battery gateway, and those issued from the smart battery gateway to other smart battery system devices; implicitly, in this scenario, there will be a single smart battery charger and single smart battery connected to the secondary SM bus. This configuration shall be referred to as a *symmetrical smart battery gateway* implementation; all other configurations where there is not a one-to-one command mapping shall be referred to as an *asymmetrical smart battery gateway* implementation.

In asymmetrical smart battery gateway implementations, the command mapping is completely dependent on the architecture of the smart battery devices on the secondary SM bus and the requirements of individual applications.

4.3 Software Architecture

This section describes the smart battery gateway reference design software architecture. Although as already mentioned, alternative approaches can be taken to implement the smart battery gateway, it is beyond the scope of this document to discuss or suggest alternatives. However, this document contains sufficient detail regarding the host to smart battery gateway interface that should the user wish to develop their own solution, this is entirely feasible and not discouraged.

The reference design software architecture makes use of a set of timers for scheduling software tasks and an interrupt driven primary SM bus communication handler.

The primary SM bus interface makes use of an on-chip SM bus peripheral. The secondary SM bus does not use an on-chip SM bus peripheral, but is instead implemented in software using a timer driving interrupt routine and state machine. This is because the C8051F340 device on which the reference design is based only provides a single SM bus hardware peripheral.

All timer and interrupt based events are asynchronous to one another; therefore care should be taken when sharing data between these event handlers to ensure data coherency and integrity. The use of semaphores to ensure effective atomic read/modify/write operations where necessary is strongly recommended.

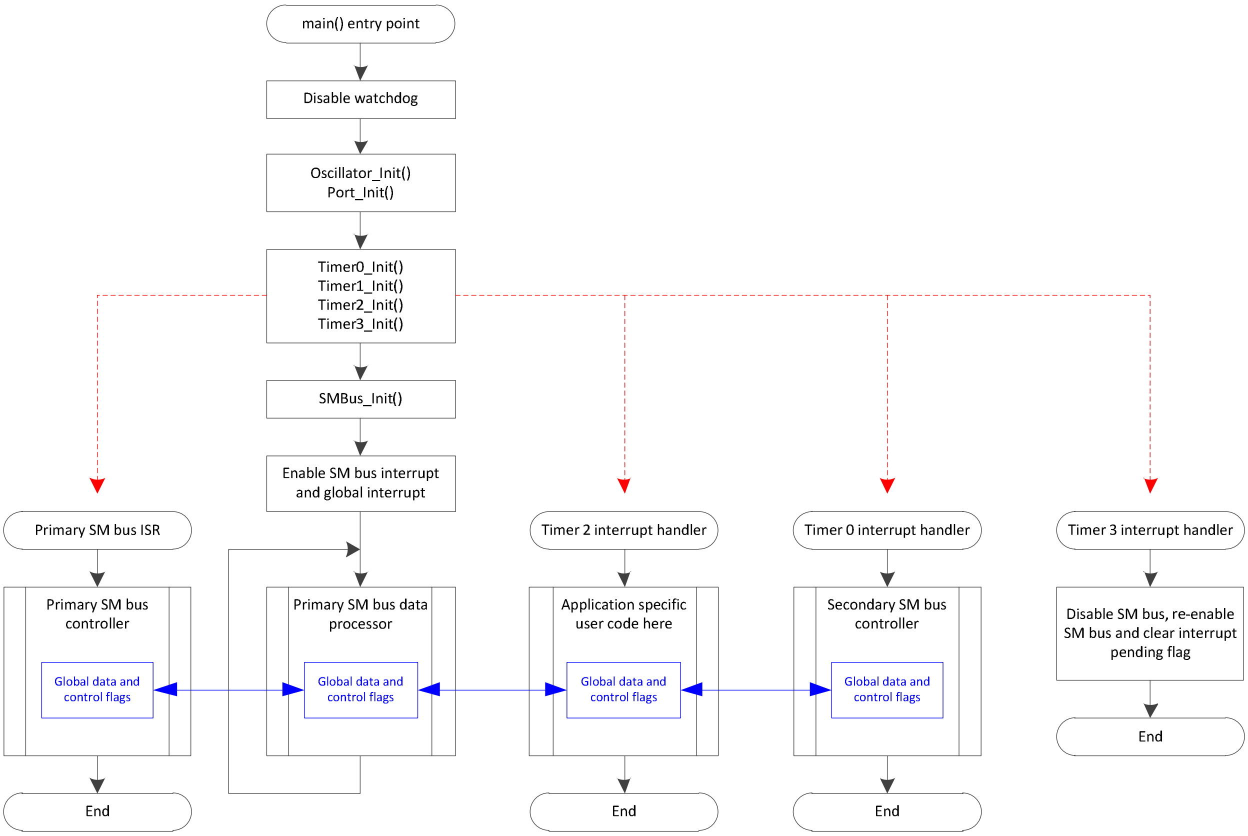

Figure 4: Software architecture

The software implementation makes use of 4 independent timers, an SM bus interrupt handler and a non-terminating main function to implement all required functionality. Each of these is described in detail.

4.3.1 Primary SM bus ISR

This is the interrupt handler for the primary SM bus controller which is generated by hardware when specific events are detected (please refer to the SMB0CN controller register in the C8051F340 document for more details). The handler is responsible for:

- Determining which event generate the interrupt
- Decoding the target device address
- Determining if the transfer is a master read or write
- Signalling the Primary SM bus data processor when data has arrived for processing

4.3.2 Primary SM bus data processor

This is the handler which is driven from the main application thread which processes the data received by the Primary SM bus ISR. It is responsible for providing the Primary SM bus ISR with data it needs to transmit back to the host when a read command is received.

4.3.3 Timer 2 interrupt handler

This handler is invoked by the timer 2 interrupt every millisecond. It is intended that application specific functionality which is responsible for processing data from communication handled by the secondary SM bus controller is placed here. Such functionality should include:

- Scheduling of communication managed by the secondary SM bus controller
- Target address and command control for the secondary SM bus controller
- Processing data received by the secondary SM bus controller

4.3.4 Timer 0 interrupt handler

This handler is invoked by the timer 0 interrupt and is responsible for implementing the software driven secondary SM bus. Communication requests are driven by the secondary SM bus controller.

4.4 Interface Example (LTC1760 Smart Battery System Manager)

The table below shows the commands which are implemented between the host controller and the smart battery gateway.

Smart Battery Device	Device Address	Command	Description
Smart Battery Charger	0x09	0x14	Read charging current
		0x15	Read charging voltage
		0x11	Read charger specification info
		0x13	Read charger status
Smart battery	0x0B	0x16	Read battery status
		0x18	Read battery design capacity
		0x10	Read battery full charge capacity
		0x19	Read battery design voltage
		0x0F	Read remaining capacity
		0x09	Read battery voltage
		0x0B	Read average current
		0x04	Write the energy consumption rate used in subsequent calculations
		0x0A	Read battery current

The table below shows the commands which are required, at a minimum, to be implemented between the LTC1760 smart battery system manager and smart battery and the smart battery gateway.

Smart Battery Device	Device Address	Command	Description
Smart Battery System Manager	0x0A	0x01	Read/write battery system state
		0x02	Read/write battery system state (continued)
		0x04	Read battery system information
		0x3C	Read/write LTC specific features
Smart battery	0x0B	0x14	Read charging current
		0x15	Read charging voltage
		0x16	Read battery status
		0x18	Read battery design capacity
		0x10	Read battery full charge capacity
		0x19	Read battery design voltage
		0x0F	Read remaining capacity
		0x09	Read battery voltage
		0x0B	Read average current
		0x04	Write the energy consumption rate used in subsequent calculations
		0x0A	Read battery current

Below is a key which describes the colour coding used for the different commands.

Key:

Gateway command re-route
Gateway managed command
Gateway autonomous command
Gateway command pass through

4.4.1 Gateway command re-route

A gateway command re-route is when the gateway has received a command from the host directed to a specific device address which is emulated by the gateway. The gateway then effectively re-routes this command by returning data from a device on the secondary SM bus with a different address to that which was specified in the original request from the host.

An example is as follows. The host makes a read request for the charging current (command 0x14) from the smart battery charger (device address 0x09). The smart battery gateway knows that it must obtain the charging current from a smart battery device connected on the secondary SB bus (device address 0x0B). Therefore, when the host requests the charging current, the smart battery gateway returns the charging current obtained from the smart battery.

The gateway should asynchronously read the charging current from the smart battery with respect to the read current request from the host device. This ensures the gateway can respond immediately to a request from the host without suffering from the latency which would be introduced if the gateway had to issue and command and wait for a response from the smart battery.

4.4.2 Gateway managed command

A gateway managed command is where the host issues a command to the gateway, which in turn issues a response which is not a re-routed command response, but is a command response which is completely or partially constructed by the gateway.

An example is as follows. The host makes a read request for the charger status (command 0x13). This command is not support by the smart battery charger attached to the secondary SM bus, so the gateway must construct a response to the host itself. In the simplest form, this may be a hard coded response by the gateway, or it might be a response which the gateway constructs based on other inferred or directly available information from smart battery devices connected to the secondary SM bus. This is dependent upon application specific requirements and it is not within the scope, nor the intent of this document to provide or suggest what such requirements should be.

4.4.3 Gateway command pass through

A gateway command pass through is when the gateway receives a command from the host directed to a specific device address which is being emulated by the gateway. The gateway returns data which it received from issuing a command on the secondary SM bus to a device with the same address; the address specified by the host is the same address that is used by the gateway to obtain the required data, hence the command is effectively passed through – there is a one-to-one mapping.

4.4.4 Gateway autonomous command

A gateway autonomous command is a command which is issued by the smart battery gateway to a smart battery device which resides on the secondary SM bus which is not directly controlled by the host.

An example is as follows. On system start up, a smart battery device, in this example the smart battery system manager, needs to be initialised to a specific configuration; this configuration defines that there is a single smart battery present which is battery 1 and this is the battery communicating with the SM bus gateway, etc.

The smart battery gateway needs to perform this initialisation and does so autonomously without the request of the host controller on power up. As the host controller knows nothing about the underlying hardware implementation, it does not know that this initialisation needs to be done, and so cannot invoke it or control it.

5 Hardware Design Reference

5.1 Interfacing with the LTC1760

This section provides information on interfacing the LTC1760 smart battery system manager with the 8051 based smart battery gateway and Robin/Woodpecker module platforms.

The below simple schematic diagram describes how the Robin module, the C8051 smart battery gateway and the LTC1760 smart battery system manager can be connected together.

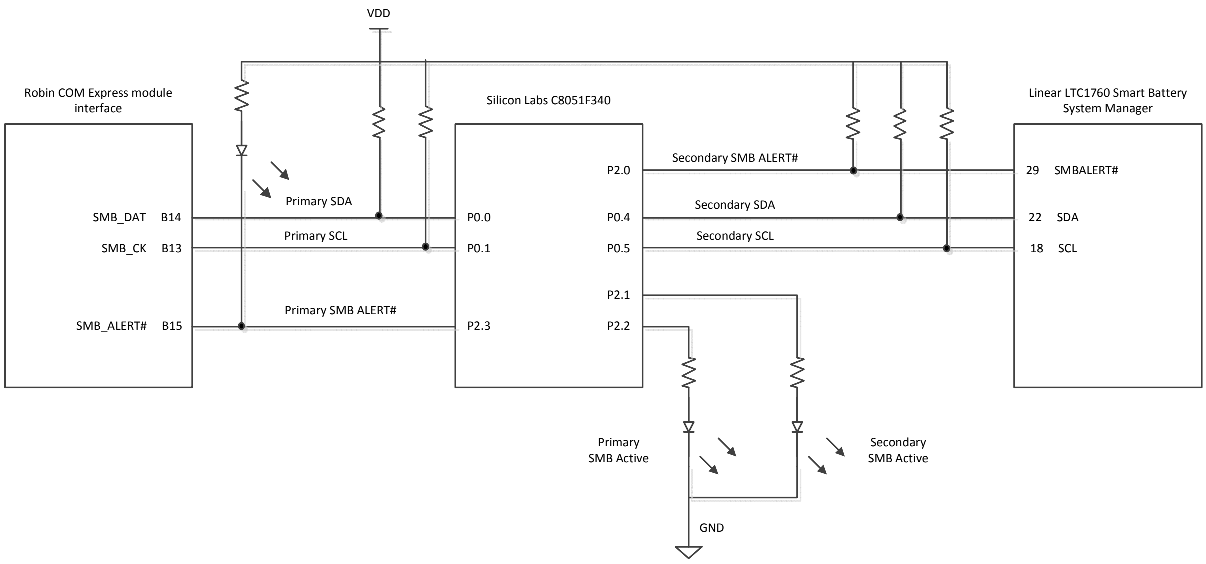

Figure 5: Smart Battery System Components Connections

The above diagram is intended as a reference to described how to interconnect the key hardware devices. The hardware design engineer should carefully follow the design guide for each device respectively to ensure that all design requirements are met.

Important notes:

1) It is strongly recommended that a programming header for the C8051 be placed on the board so that it can be easily programmed and debugged (if necessary) using the Silicon Laboratories USB Debug Adapter. This MUST be included if you are not going to program devices prior to assembling them into the PCB, otherwise there will be no way of programming them.

2) It is strongly recommended that debug headers for each SM bus also be included on the PCB to allow an external bus analyser (such as the Total Phase Beagle I2C analyser) to be attached for debug purposes. If headers cannot be included, it is recommended that at least test points are included for easy access to these signals.