ACPI Smart Battery System Overview + Discussion of Implementations of Smart Battery Systems

February 2-4, 1998

Matt Fruin

Intel MHPG MSO Boards

AGENDA

- ACPI Smart Battery System Overview
  - Purpose of OSPM
  - General concept of OSPM and ACPI
  - How Smart Batteries fit into an ACPI system
- Description of ACPI Smart Battery Systems
  - ACPI compliant (general case)
  - SMBus Architecture using Control Methods
- Implementation Issues with SMBus
  - Multiple SMBuses
  - Mixing I2C and SMBus devices

Purpose of OSPM

- Current technology = legacy power management (BIOS-directed)
  - Existing power management (legacy PM) is pushed to its limits
    - all power management implemented in BIOS (SMI code)
      - uses hardware timers and SMI traps to guess when devices or system idle
    - additional power savings are complex and costly
  - Power management is migrating to new platforms
    - current architecture designed for uniprocessor systems
    - add-in cards never intended for power management
  - Plug-and-Play (PNP) mechanism also outdated
    - replaced by bus specifications

Purpose of OSPM, cont.

- Next level of PM = OS-directed Power Management (OSPM)
  - Assigns policy decisions to higher levels of software
    - device drivers determine when devices are idle
    - applications can become involved in power management
    - OS can change its behavior based on power remaining and user input
  - Integrates PM and PNP into OS
  - Advanced Configuration & Power Interface
    - an interface for controlling PM and PNP for devices not compliant with device class and bus class specifications

Purpose of OSPM, cont.

- Migrating power management policy to the OS enables greater power savings
  - Power decisions in system can be made globally, rather than locally
  - OS can make decisions on actions it takes based on how much power is remaining
    - example: what level of sleep to enter
    - example: postponing routine maintanance on HD
  - Enables finer control of CPU low power states
  - I/O operations can be classified as “lazy” (those that don’t have to be performed immediately) or not
    - example: word processing auto-save only when disk is already spinning

OSPM and ACPI

- The OS uses ACPI to interface to system board hardware
  - ACPI provides a standard mechanism for informing the OS what kind of capabilities the hardware has
    - how to enter and exit system power states
    - how to put individual devices into lower power states
    - what capabilities devices have in low power states
    - PM, PNP, and Event management
  - Devices that are compliant with bus class or device class specifications are controlled directly by OS

OSPM and ACPI, cont.

- ACPI overview, cont.
  - ACPI covers devices not compliant with “bus class” and “device class” specifications
    - PCI, audio controllers, network controllers, etc. have standard interfaces defined in the specifications
      - OS directly access these types of devices to perform power management and configuration
      - example: PCI bus power management specification defines mechanisms to put devices in low-power states
    - ACPI focuses on “value-added” additional power management hardware
      - devices not compliant with bus or device class specifications
      - devices implementing extra power control functionality
      - example: IDE hard drive power-down control circuitry

OSPM and ACPI, cont.

- ACPI overview, cont.
  - ACPI events generate System Control Interrupts (SCIs) to notify OS
    - runtime events (occur while system is running)
      - example: notebook computer lid closing
      - example: Smart Battery low battery warning
    - wake events (occur while system is in low power state)
      - example: ring indicate from modem
      - example: Smart Battery critical low battery alarm
    - runtime and wake events must be on separate SCI-capable input lines

OSPM and ACPI, cont.

- ACPI overview, cont.
  - Code is written in ACPI Source Language (ASL)
    - There are three runtime components to ACPI:
      - ACPI tables
      - ACPI registers
      - ACPI BIOS
    - compiled into ACPI Machine Language (AML)
    - AML is interpreted by ACPI OS
      - functions (control methods) are called by OS
      - AML code resides in system ROM
  - Allows system to be customized and differentiated

OSPM and ACPI, cont.

- OS makes power management decisions
  - Standard interfaces are supported directly in “bus class” and “device class” specifications
  - Additional power control hardware is made known to OS through ACPI tables
    - mechanisms to control power management hardware
    - interrupts that inform OS of important power-related events

Smart Battery and ACPI

- An ACPI-compliant SMBus interface allows OS direct access to the battery subsystem
  - ACPI specification defines possible battery subsystem architectures
  - ACPI specification describes how OS communicates with battery subsystem
  - All Smart Battery commands supported for recommended Smart Battery implementation
    - trivially easy to interface to standard configuration
      - supported directly by the operating system
    - more difficult to interface to non-standard configuration (and has less functionality)
      - requires AML (written by OEM / BIOS vendor) to interface between OS and battery subsystem

Smart Battery and ACPI, cont.

- Smart Battery Subsystem Requirements
  - An ACPI-compliant Smart Battery subsystem consists of:
    - An SMBus host controller (CPU to SMBus host controller) interface
      - accessed through an embedded controller (EC)
    - At least one Smart Battery
    - A Smart Battery Charger
    - A Smart Battery Selector (if more than one Smart Battery is supported)
  - ACPI requires an interrupt to notify the system of a change in battery or AC status
    - if no selector, some other device must provide this functionality

Smart Battery and ACPI, cont.

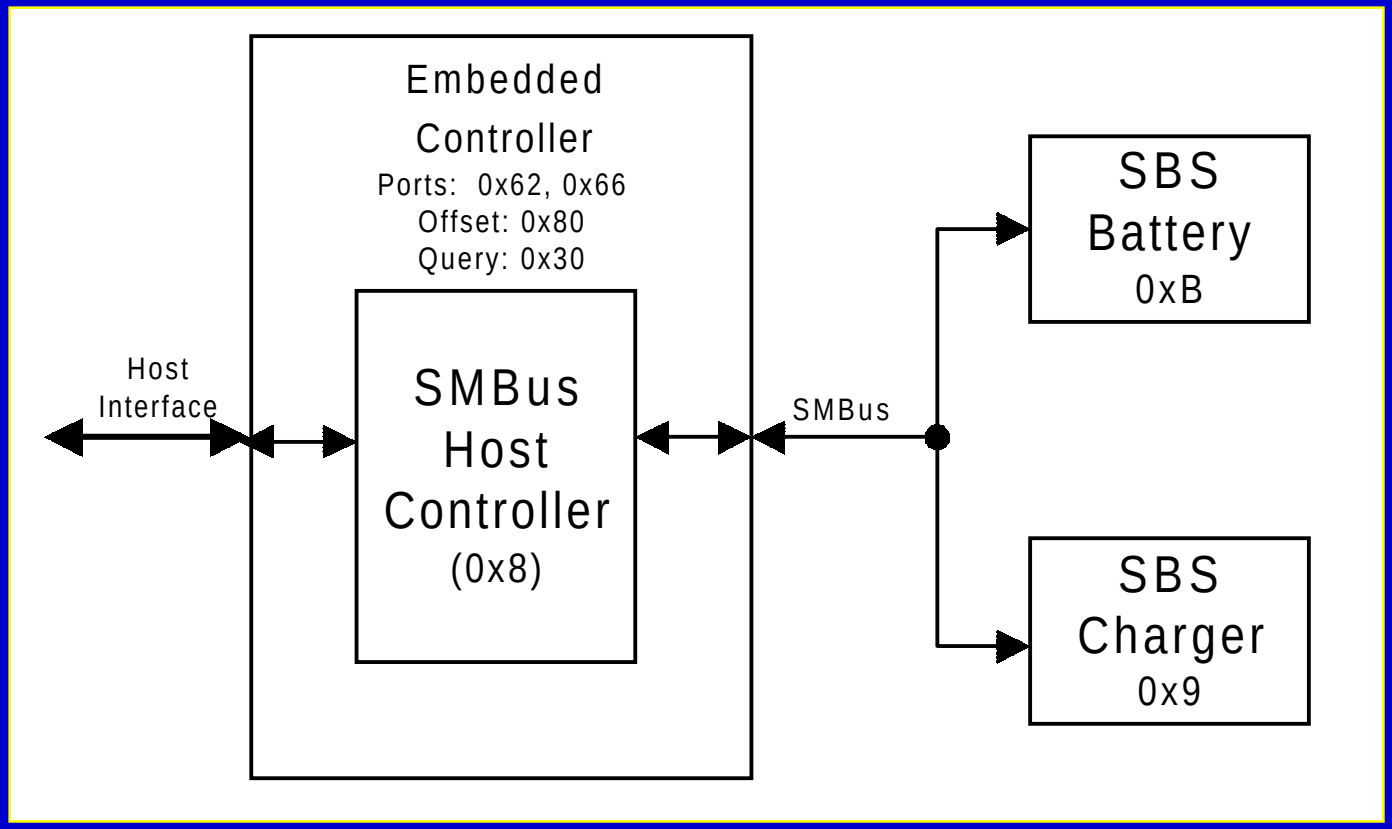
- Smart Battery Subsystem Requirements, cont.
  - Can meet the requirements using SMBus components
    - individual charger and separate selector
    - combined charger / selector device
  - Can also meet requirements by emulating SMBus devices
    - example: charger and / or selector functions provided by a microcontroller
    - ***must*** be software transparent

Smart Battery and ACPI, cont.

- ACPI defines five compatible Smart Battery configurations:
  - Maximum of 1 Smart Battery and no selector
  - Maximum of 1 Smart Battery and a selector
  - Maximum of 2 Smart Batteries and a selector
  - Maximum of 3 Smart Batteries and a selector
  - Maximum of 4 Smart Batteries and a selector

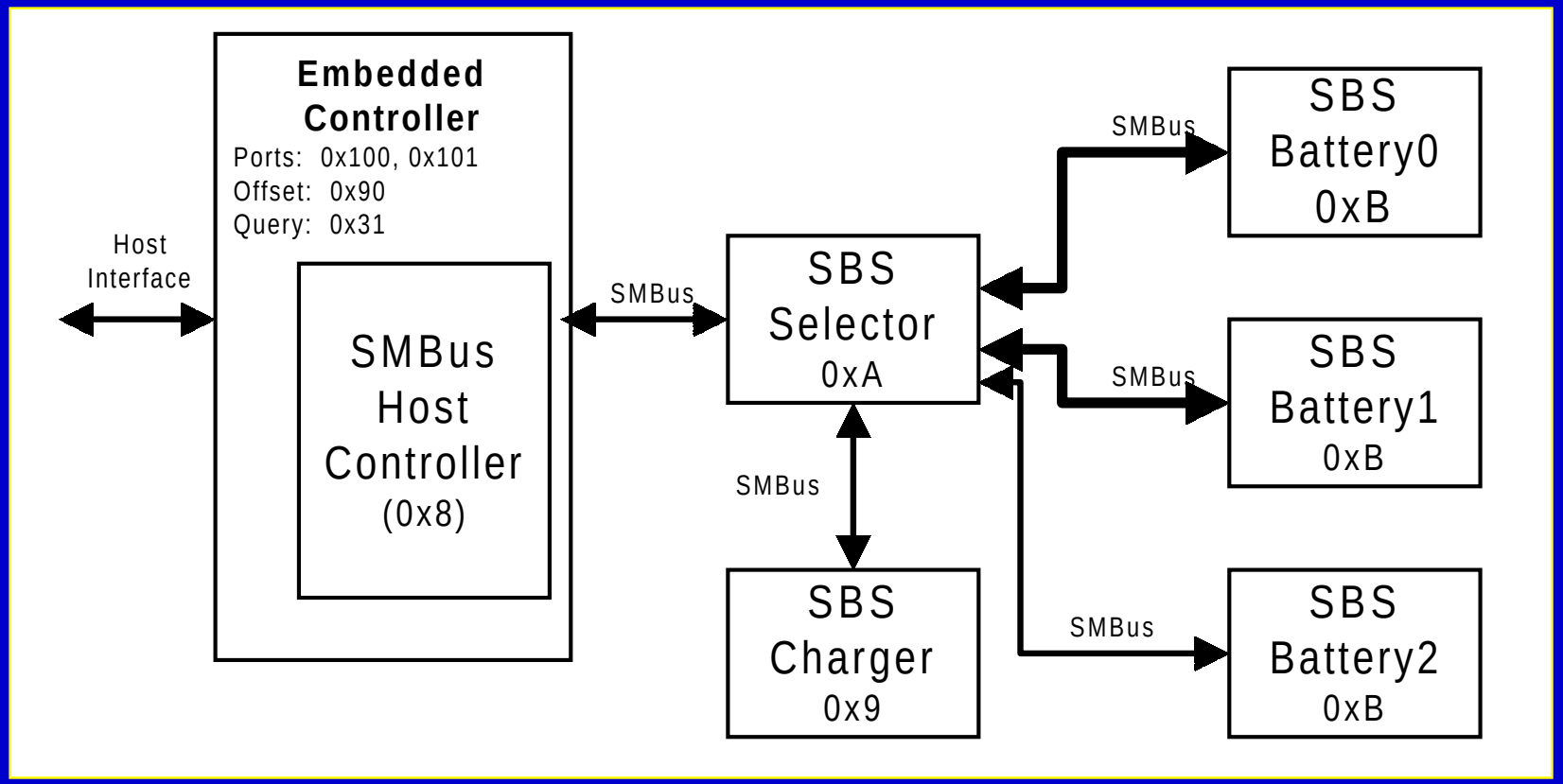
Smart Battery and ACPI, cont.

- Example: single Smart Battery subsystem, no selector (figure 11-2 from ACPI spec)



Smart Battery and ACPI, cont.

- Example: multiple Smart Battery subsystem, with selector (figure 11-3 from ACPI spec)



Smart Battery and ACPI, cont.

- ACPI ASL code is very simple for ACPI-compliant Smart Battery subsystem
  - Specifies where in I/O space SMBus interface (host controller) is located
  - Defines address and query value for OS to interface to SMBus through host controller
  - Specifies interrupt mechanism
  - Indicates what type of Smart Battery system
    - how many batteries
    - whether a selector is present

Smart Batteries and Control Methods

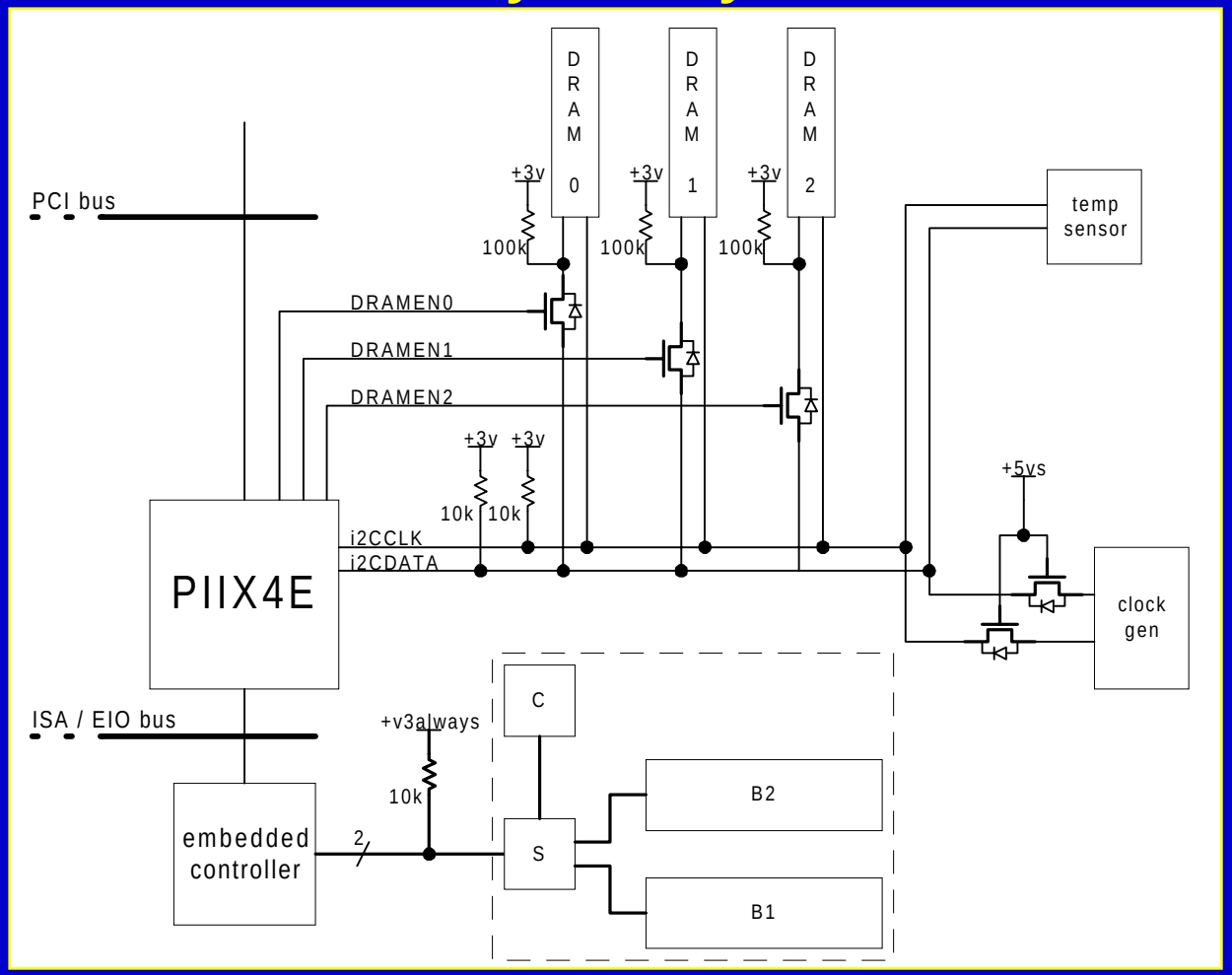
- Control Method Battery (CMBatt)
  - If battery subsystem is not ACPI compliant, must use control methods to interface to battery subsystem
    - supply ASL code for ACPI battery functions:
      - report static information about battery (_BIF object)
      - report present battery status (_BST object)
      - set battery capacity warning (_BTP object)
      - report current power source (_PSR object)
    - requires more effort on part of OEM
    - not as full-featured
      - data is of unknown veracity
      - Lose “At-Rate” functions
  - Don’t necessarily need to use SMBus to communicate with CMBatt subsystem
    - CM can communicate with any hardware

Multiple SMBuses

- May be desirable to have multiple SMBuses in a system
  - Isolate smart battery subsystem from other SMBuses
    - avoid bus collisions from too much traffic
    - isolate I2C devices from SMBus devices
    - avoid leakage problems with one SMBus powered up and another powered down
    - logically group devices by how often they are accessed
    - logically group devices by who accesses them

Multiple SMBuses, cont.

- Trajan reference platform isolates I2C devices from Smart Battery subsystem



Mixing I2C and SMBus Devices

- SMBus specification is based on I2C, with some additional requirements
  - Functional
    - SMBus limits operating frequency between 10kHz and 100kHz (I2C operates down to DC)
    - SMBus defines a timeout condition (none on I2C)
    - SMBus specifies maximum time a slave can hold the bus (not limited in I2C)
    - SMBus defines additional sideband signals: SMBSUS# and SMBALERT#
  - Electrical
    - SMBus defines fixed voltage signaling levels (I2C is usually referenced to VDD)
    - SMBus specifies much smaller pullup current than I2C
  - Protocols
    - SMBus defines data protocols (none defined by I2C)

Mixing I2C and SMBus devices, cont.

- Since I2C and SMBus are so similar, it is possible to mix devices on the same bus
  - Should meet more-stringent SMBus electrical signaling levels
  - Should meet more-restrictive SMBus capacitive loading requirement
  - I2C devices should be “well-behaved”
    - must not indefinitely hang the bus
    - if a danger, isolate charger / battery interface from I2C devices (eg, using a Smart Battery Selector component)
  - Avoid too much traffic seen by Smart Battery devices

SUMMARY

- Next generation operating systems will implement OSPM for power management
- ACPI is an integral part of OS PM
- SMBus is a standard bus defined by ACPI
- Smart Battery is an important part of ACPI
  - An ACPI-compliant Smart Battery subsystem will be directly supported by the operating system
  - Can also use CMBatt battery subsystem