Application Note

Tegra Embedded Controller
Interface Specification 1.0

October 6, 2009
DA_04649-001_11

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com

Table of Contents

List of Tables

List of Figures

Revision History

Version	Date	Description
1.0	27 Aug 2009	Version 1.0 for initial release.

1. Introduction

This specification defines a communication interface between NVIDIA® Tegra Application Processor (AP) and an embedded controller (EC) for netbook applications. Using the interface, the AP can direct the EC to carry out certain netbook system operations. The AP can also query netbook system information from the EC.

The System Management Bus (SMBus) is used as the underlying transport layer for communications between the AP and EC. All of the bus protocols utilized in this document are fully compliant with SMBus specification. Therefore, no Tegra-dedicated SMBus segment is required in the Netbook architecture—Tegra can be connected to any existing segment controlled by EC.

For detailed information on SMBus protocols (including the notation used for protocol depiction) see the following specification:

- System Management Bus (SMBus) Specification, V2.0, August 3, 2000
  http://smbus.org/specs/

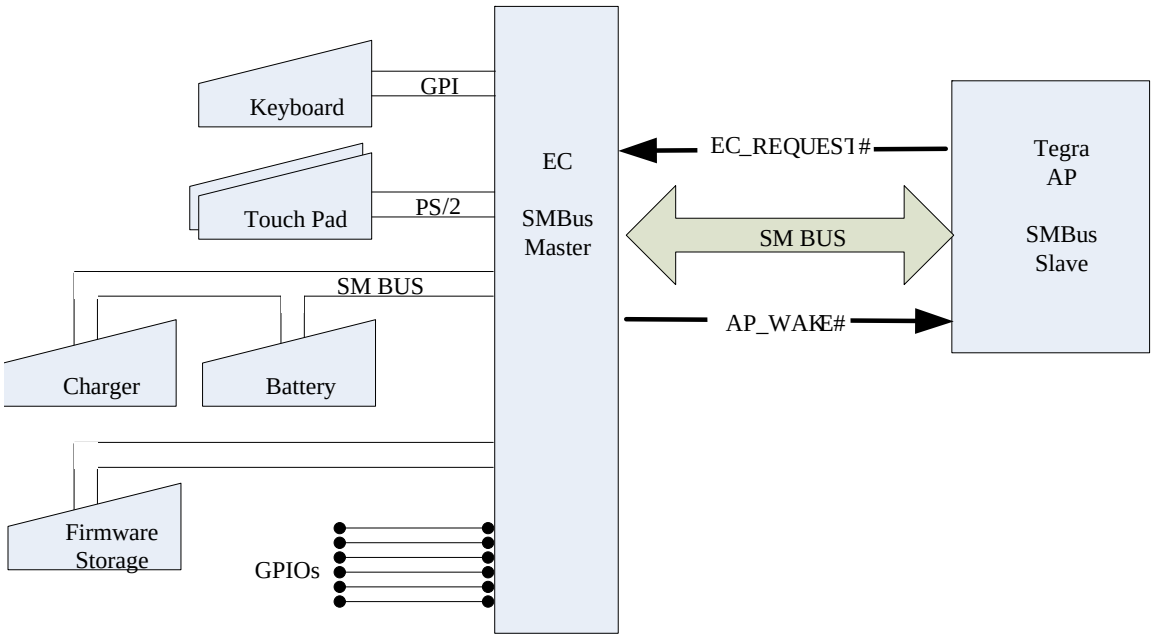
2. System Connectivity

Figure 1 shows the connections between Tegra AP and the EC. In the netbook system, the EC is responsible for controlling the keyboard, touch pad, GPIOs, and battery. In addition, the EC is responsible for waking up the AP from deep sleep in response to certain events (such as key presses). The AP sends commands to the EC, which then carries out the desired operations.

This specification identifies the communication protocol between the AP and the EC. It also specifies the format of messages sent via the protocol. The EC must implement the communication protocol, plus the operations defined by the various messages.

Finally, some EC operations may need to take place without the involvement of the AP. Battery charging, hotkey processing, backlight control, and volume control are possible examples. Such operations are outside the scope of this document but will nevertheless need to be defined and implemented according to the platform requirements.

Figure 1—System Connectivity



3. SMBus Overview

Some implementations of netbook EC do not support SMBus slave mode operations. To avoid this limitation, the EC always acts as an SMBus master in communications with the AP. Correspondingly, the AP always acts as an SMBus slave.

To transfer data from the EC to the AP, the EC simply performs an SMBus block write operation directed to the AP.

To transfer data from the AP to the EC, an active-low side-band signal (EC_REQUEST#) is used. It is asserted by Tegra AP to notify the EC that a request is pending. In response, the EC performs an SMBus block read operation from the AP. The AP deasserts the EC_REQUEST# signal before the block read operation completes. The EC is expected to respond to level changes in the EC_REQUEST# signal rather than to edges so as to avoid missing the first request after system startup.

The AP may choose to enter a low-power sleep state in order to minimize power consumption. In this state, the AP will be unable to respond to any SMBus transactions. The AP notifies the EC (via an SMBus transaction) before entering the sleep state, after which the EC must cease all further SMBus transactions involving the AP.

The EC may be configured to wake the AP in response to certain events. To wake the AP, the EC asserts an active-low side-band signal, AP_WAKE#. The AP responds to level changes in the AP_WAKE# signal, so the EC must keep the signal asserted until the AP confirms that it is awake. When the AP wakes up, it signals its readiness to resume SMBus transactions by asserting EC_REQUEST# and thereby initiating a data transfer to the EC. The EC then deasserts AP_WAKE# and resumes normal SMBus operations, starting with the data transfer initiated by the assertion of EC_REQUEST#.

If the system architecture allows other devices besides the EC to wake up the AP, then the EC must continue to monitor EC_REQUEST# even after the AP has gone to sleep. If the AP is awakened due to some non-EC-related event, assertion of EC_REQUEST# is the only indication to the EC that SMBus operations need to resume.

Any device on the SMBus has a unique 7-bit address. Tegra AP has a default address of **1000_101b**.[1] In the case that another device with that address is connected to the same SMBus segment, the default AP address can be reprogrammed. The mechanism of address re-programming is beyond the scope of this document.

As an SMBus slave device, Packet Error Checking (PEC) support is optional for the Tegra AP. The mechanism for enabling and disabling PEC support, if any, is beyond the scope of this document.

[1] NVIDIA is communicating with SMBus Working Group in order to include Tegra with this address into SMBus device assignment list.

4. Packets

The SMBus block read and block write operations between AP and EC transport three types of packets:

- Requests (from AP to EC)—operations initiated by the AP
- Responses (from EC to AP)—EC-generated response to a request packet
- Asynchronous events (from EC to AP)—EC-generated event

All packets contain a Command field. In addition, Sub-command, Packet Length, Status, and Payload fields may also be present, depending on the packet type.

4.1. Packet Fields

4.1.1. Command Field

The command field is one byte in size and encodes core information about the packet, as shown in Table 1. Event Packets can be variable length or fixed length; whereas Request Packets and Response Packets are always variable length.

The Requestor Tag is assigned by the AP for each Request Packet and echoed back by the EC as part of the corresponding Response Packet. It is used to de-mux the packet stream received from the EC, such that each Response Packet can be connected to its originating Request Packet and routed back to the appropriate requestor. When generating a Response packet, the EC copies the Command field from the Request Packet into the Response Packet verbatim, thus ensuring that the Requestor Tag is echoed back to the AP unchanged.

The Command Type field indicates the general category of operation to be carried out. The Sub-command field (Section 4.1.2) provides further information about the intended operation.

> **Note**: Some Command Type and Event Type values have been set aside for OEM-defined extensions. To ensure compatibility with future revisions of this document, OEMs should not use any of the reserved Command Type or Event Type values.

Table 1—Command Field Structure

Command Field Bit(s)	Description
7:7	Packet Type 0b—Response Packet or Request Packet 1b—Event Packet
for Event Packets (Packet Type is 1b)	
6:5	Transfer Type 0h—Fixed length transfer, Command field + 1 byte 1h—Fixed length transfer, Command field + 2 bytes 2h—Variable length transfer, as specified by Packet Length field (Section 4.1.3) 3h—Reserved
4:4	Error Flag 0b—Successful completion; no error occurred 1b—An error occurred, as specified by the Status field (Section 4.1.4)
3:0	Event Type 0h—Keyboard 1h—Auxiliary Device 0 (PS/2 port 0) 2h—Auxiliary Device 1 (PS/2 port 1) 3h—Auxiliary Device 2 (PS/2 port 2) 4h—Auxiliary Device 3 (PS/2 port 3) 5h—System 6h – GPIO (scalar) 7h – GPIO (vector) 8h – Battery dh – OEM-defined eh – OEM-defined Others—Reserved
for Request Packets and Response Packets (Packet Type is 0b)	
6:4	Requestor Tag—value assigned by AP and echoed back by EC

3:0	Command Type 1h—System Status 2h—Battery Information 3h—GPIO Control 4h—Sleep State Control 5h—Keyboard 6h—Auxiliary Device 7h—EC Control dh – OEM-defined eh – OEM-defined Others—Reserved

4.1.2. Sub-command Field

The Sub-command field is one byte in size and further refines the Command Type. The structure and contents of the Sub-command field depend on the Command Type. Sub-command values are detailed in Sections 6 and 7 below.

When generating a Response Packet, the EC copies the Sub-command field verbatim from the associated Request Packet.

4.1.3. Packet Length Field

The Packet Length field is one byte in size and encodes the size of the packet (in bytes). The Packet Length field must always be specified for variable length packets (Transfer Type is 2h), but must not be specified for fixed length packets (Transfer Type is 0h or 1h). Due to the structure of the SMBus block operations, the Packet Length is computed differently for block read operations (i.e., Request packets) than for block write operations (i.e., Response packets and Event packets). See Section 5 for details of the SMBus block operations. Packet length can never be zero.

4.1.4. Status Field

The Status field indicates whether the requested operation could be completed successfully. Response Packets are required to contain the Status field, whereas Request Packets never contain the Status field, and Event Packets contain the Status field only when the Error Flag is non-zero. Legal values are tabulated below.

> **Note**: Some Status values have been set aside for OEM-defined extensions. To ensure compatibility with future revisions of this document, OEMs should not use any of the reserved Status values.

Table 2—Status Field Structure

Status Field Bit(s)	Description
7:0	Status 00h—Successful completion 01h—PS/2 Auxiliary Port time-out error 02h—PS/2 Auxiliary Port parity error 03h—Hardware resource temporarily unavailable, i.e., requested data from an empty slot (=not connected battery) 04h—Invalid command (or sub-command) 05h—Invalid size. Packet size is incorrect for the specified operation 06h—Invalid parameter. Payload data is incorrect for the specified operation 07h – Unsupported configuration. Requested configurations are not supported by the EC hardware and/or firmware. 08h – Checksum error 09h – Device write error 0ah – Device read error 0bh – Data overflow error 0ch – Data underflow error 0dh – Invalid state error. The EC is not in the proper state to carry out the requested operation d0h-efh – OEM-defined errors ffh—Error Report: any other reason than described above Others—Reserved

4.1.5. Payload Field

The Payload field can be variable length or fixed length depending on the Transfer Type setting in the Command field. The content of the Payload field depends on the Packet Type, Command Type, Sub-command Type, and Event Type. The size of the payload field can be inferred from the Packet Type, Transfer Type, and Packet Length.

5. SMBus Messages

This section describes the SMBus messages corresponding to the three packet types (Request, Response, and Event).

The largest SMBus operation is 34 bytes in length. This corresponds to a Block Write operation with a Byte Count of 32 (Sub-command, Status, and 30-byte Payload) and PEC. This allows for 30 bytes of a battery's identification string to be reported.

5.1. Request Packet Format

Request packets are always sent from the AP to the EC using SMBus block read operations. The EC always acts as an SMBus master and the AP always acts as an SMBus slave.

> **Note**: The SMBus Command must be the value 0x1. Also, the requestor tag value in the (Request) command field must be non-zero.

The diagrams in the following sections follow the convention set out in the SMBus Specification (Version 2.0, August 3, 2000). From Figure 5-1 of that spec, the top row is the number of bits in each field, and the middle row contains the SMBus field names. Grey boxes indicate slave-to-master transfers, whereas white boxes indicate master-to-slave transfers. Abbreviations are shown in the following table.

Table 3—SMBus Packet Protocol Diagram Elements

Abbreviation	Full Name
S	Start Condition
Sr	Repeated Start Condition
Rd	Read (bit value of 1)
W	Write (bit value of 0)
A	Acknowledge ('0' for ACK, '1' for NACK)
P	Stop Condition
PEC	Packet Error Code

Finally, the bottom row in the diagram gives the EC Interface field names. If the EC Interface requires a specific constant value in a given field, the constant is shown rather than the field name.

Figure 2—Request Packet Mapped to SMBus Block Read Operation

1	7	1	1	8	1	1	7	1	1
S	Slave Address	Wr	A	Command	A	Sr	Slave Address	Rd	A
	Tegra SMBus Address	0	0	0x1	0		Tegra SMBus Address	1	0

8	1	8	1	8	1	8	1	
Byte Count	A	Data Byte 1	A	Data Byte 2	A	Data Byte 3	A	…
Packet Length (N)	0	Request Command	0	Sub-command	0	Payload	0	

8	1	8	1	1
Data Byte N	A	PEC[2]	A	P
Payload	0[3]		1	

[2] Packet Error Code (PEC) byte and following Acknowledge bit are optional. PEC byte is calculated over the entire message as defined by SMBus specification.

[3] If the PEC is present, then the preceding Acknowledge will be an ACK, otherwise it will be a NACK.

5.2. Response Packet Format

Response packets are always sent from the EC to the AP using SMBus block write operations. The EC always acts as an SMBus master and the AP always acts as an SMBus slave. The command and sub-command fields are copied verbatim from the corresponding Request packet.

In some cases, the Response simply needs to indicate whether the Request succeeded or failed; no payload data is needed or supplied. This special case of a Response packet is called an Ack packet.

Note: The requestor tag value in the (Response) command field must be non-zero.

Figure 3—Response Packet Mapped to SMBus Block Write Operation

1	7	1	1	8	1	8	1	8	1
S	Slave Address	Wr	A	Command	A	Byte Count	A	Data Byte 1	A
	Tegra SMBus Address	0	0	Response Command	0	Packet Length (N)	0	Sub-Command	0

8	1	8	1		8	1	8	1	1
Data Byte 2	A	Data Byte 3	A	…	Data Byte N	A	PEC[2]	A	P
Status	0	Payload	0		Payload	0		0	

Figure 4—Ack Packet Mapped to SMBus Block Write Operation

1	7	1	1	8	1	8	1	8	1
S	Slave Address	Wr	A	Command	A	Byte Count	A	Data Byte 1	A
	Tegra SMBus Address	0	0	Response Command	0	Packet Length (N)	0	Sub-Command	0

8	1	8	1	1
Data Byte 2	A	PEC[2]	A	P
Status	0		0	

5.3. Event Packet Format

Event Packets are always sent from the EC to the AP and can be either variable length or fixed length. Variable length packets are sent using the SMBus block write operation. Fixed length packets are sent using the SMBus write byte or SMBus write word operation, depending on the payload size (one byte or two bytes). The EC always acts as an SMBus master and the AP always acts as an SMBus slave.

Note: If the Error Flag is non-zero, then the Status field replaces the first byte of the Payload field.

Figure 5—Variable Length Event Packet Mapped to SMBus Block Write Operation

1	7	1	1	8	1	8	1	8	1
S	Slave Address	Wr	A	Command	A	Byte Count	A	Data Byte 1	A
	Tegra SMBus Address	0	0	Event Command	0	Packet Length (N)	0	Payload	0

8	1	8	1		8	1	8	1	1
Data Byte 2	A	Data Byte 3	A	…	Data Byte N	A	PEC[2]	A	P
Payload	0	Payload	0		Payload	0		0	

Figure 6—Fixed Length Event Packet Mapped to SMBus Write Byte Operation

1	7	1	1	8	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	Data Byte 1	A	PEC[2]	A	P
	Tegra SMBus Address	0	0	Event Command	0	Payload	0		0	

Figure 7—Fixed Length Event Packet Mapped to SMBus Write Word Operation

1	7	1	1	8	1	8	1	8	1
S	Slave Address	Wr	A	Command	A	Data Byte 1	A	Data Byte 2	A
	Tegra SMBus Address	0	0	Event Command	0	Payload	0	Payload	0

8	1	1
PEC[2]	A	P
	0	

Figure 8—Variable Length Event Packet with Error Mapped to SMBus Block Write Operation

1	7	1	1	8	1	8	1	8	1
S	Slave Address	Wr	A	Command	A	Byte Count	A	Data Byte 1	A
	Tegra SMBus Address	0	0	Event Command[4]	0	Packet Length (N)	0	Status	0

8	1	8	1		8	1	8	1	1
Data Byte 2	A	Data Byte 3	A	…	Data Byte N	A	PEC[2]	A	P
Payload	0	Payload	0		Payload	0		0	

[4] The Error Flag in the Command Field will be set in this packet, indicating that an error has occurred and that the Status Field will replace the first byte of the Payload.

Figure 9—Fixed Length Event Packet with Error Mapped to SMBus Write Byte Operation

1	7	1	1	8	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	Data Byte 1	A	PEC[2]	A	P
	Tegra SMBus Address	0	0	Event Command[4]	0	Status	0		0	

Figure 10—Fixed Length Event Packet with Error Mapped to SMBus Write Word Operation

1	7	1	1	8	1	8	1	8	1
S	Slave Address	Wr	A	Command	A	Data Byte 1	A	Data Byte 2	A
	Tegra SMBus Address	0	0	Event Command[4]	0	Status	0	Payload	0

8	1	1
PEC[2]	A	P
	0	

6. Request and Response Packets

The supported commands and sub-commands for Request packets from the AP are summarized in the following sections, along with the corresponding Response packets from the EC.

The EC is required to send a response for every request that it receives. If an unrecognized or unsupported command or sub-command is received, an Ack packet with a Status of Invalid Command must be reported.

The EC must ignore all reserved bits in the request packets that it receives. Similarly, the EC must set all reserved bits to zero in the response packets and event packets that it sends.

6.1. System Status

This request is used to query the status of the EC. It is also allowed to convey requests from the EC to the AP, including requests that the AP power down or enter its sleep state.

The system status can be queried manually using the Get System Status command. When event reporting is enabled via the Configure Event Reporting command, the EC automatically sends a System Event packet (see Section 7.1) whenever the system status changes. Event reporting is initially disabled after reset.

Some of the system status fields persist until cleared via the Acknowledge System Status command. These include the AP Power Down Request, AP Suspend Request, and AP Restart Request fields, by which the EC triggers a power state change on the AP.

> **Note**: The acknowledge indicates that the AP has received the request and will begin processing it, not that processing of the request has been completed. Similarly, the EC Reset Notification field is used by the EC to alert the AP that a reset has occurred. This field is set by default when the EC boots up and persists until acknowledged by the AP. The EC will report a Status of Success if the AP acknowledges a non-persistent system status field or a persistent status field that is already cleared.

The AP can also be awakened from its sleep state due to changes in the system status. The Configure Wake command controls whether this functionality is enabled. Further discussion of sleep control can be found in Section 6.4.

The system status commands and sub-commands are tabulated below.

Table 4—Request System Status

SMBus Protocol Byte	Description	Note
Command Byte	01h	System Status
Byte Count	01h	
Data Byte 1 (Sub-Command)	7:0	00h Get System Status 01h Configure Event Reporting 02h Acknowledge System Status fdh Configure Wake

Data Byte 2 – Data Byte N		For sub-command 01h, configuration setting For sub-command 02h, system event types For sub-command fdh, configuration setting Otherwise, no data bytes

6.1.1. Get System Status Response

Table 5—Response Get System Status

SMBus Protocol Byte	Description	Note
Command Byte	01h	System Status
Byte Count	06h	
Data Byte 1 (Sub-Command)	00h	Get System Status
Data Byte 2 (Status)	00h	Successful
Data Byte 3	System State Bits 7-0	See Table 6
Data Byte 4	System State Bits 15-8	
Data Byte 5	OEM System State Bits 7-0	See Table 7
Data Byte 6	OEM System State Bits 15-8	

Table 6—System State Flags

System State Bit(s)	Description
15:5	Reserved (00h)
4	EC Reset Notification 0b – No EC Reset has occurred 1b – An EC Reset has occurred Notification persists until acknowledged
3	AP Power Down Request 0b – EC does NOT request that AP power down 1b – EC requests that AP power down Request persists until acknowledged
2	AP Suspend Request 0b – EC does NOT request that AP enter suspend state 1b – EC requests that AP enter suspend state Request persists until acknowledged

1	AP Restart Request 0b – EC does NOT request that AP restart 1b – EC requests that AP restart Request persists until acknowledged
0	AC Present 0b—No AC (Battery Power) 1b—AC Present No acknowledge required

Table 7—OEM System State Flags

System State Bit(s)	Description
15:0	OEM defined

6.1.2. Acknowledge System Status Request

Table 8— Acknowledge System Status Request

SMBus Protocol Byte	Description	Note
Command Byte	01h	System Status
Byte Count	05h	
Data Byte 1 (Sub-Command)	02h	Acknowledge System Status
Data Byte 2	System State Flag Bits 7-0	See Table 6. Each bit value controls the corresponding System State Flag. 0b – Do not acknowledge 1b – Acknowledge
Data Byte 3	System State Flag Bits 15-8	
Data Byte 4	OEM System State Flag Bits 7-0	See Table 7. Each bit value controls the corresponding OEM System State Flag. 0b – Do not acknowledge 1b – Acknowledge
Data Byte 5	OEM System State Flag Bits 15-8	

6.1.3. Acknowledge System Status Response

Response to an Acknowledge System Status Request is an Ack packet.

6.1.4. Configure Event Reporting Request

Table 9— Configure Event Reporting Request

SMBus Protocol Byte	Description	Note
Command Byte	01h	System Status
Byte Count	06h	
Data Byte 1 (Sub-Command)	01h	Configure Event Reporting
Data Byte 2	Configuration Action	0h – Disable event reporting 1h – Enable event reporting
Data Byte 3	System State Flag Mask Bits 7-0	See Table 6. Each mask bit controls whether the Configuration Action is applied to the corresponding System State Flag. 0b – Do not apply Configuration Action 1b – Apply Configuration Action
Data Byte 4	System State Flag Mask Bits 15-8	
Data Byte 5	OEM System State Flag Mask Bits 7-0	See Table 7. Each mask bit controls whether the Configuration Action is applied to the corresponding OEM System State Flag. 0b – Do not apply Configuration Action 1b – Apply Configuration Action
Data Byte 6	OEM System State Flag Mask Bits 15-8	

6.1.5. Configure Event Reporting Response

Response to a Configure Event Reporting Request is an Ack packet.

6.1.6. Configure Wake Request

Table 10— Configure Wake Request

SMBus Protocol Byte	Description	Note
Command Byte	01h	System Status
Byte Count	06h	
Data Byte 1 (Sub-Command)	fdh	Configure Wake
Data Byte 2	Configuration Action	0h – Disable wake for event 1h – Enable wake for event
Data Byte 3	System Event Type Mask Bits 7-0	See Table 6. Each mask bit controls whether the Configuration Action is applied to the corresponding System Event Type. 0b – Do not apply Configuration Action 1b – Apply Configuration Action
Data Byte 4	System Event Type Mask Bits 15-8	
Data Byte 5	OEM System Event Type Mask Bits 7-0	See Table 7. Each mask bit controls whether the Configuration Action is applied to the corresponding OEM System Event Type. 0b – Do not apply Configuration Action 1b – Apply Configuration Action
Data Byte 6	OEM System Event Type Mask Bits 15-8	

6.2. Battery Information

This request is used to query information about batteries connected to the EC. The EC can be configured to notify the AP when certain battery events occur, such as when a battery is inserted or removed, when the charging state changes, or when a low-power threshold is reached. Notifications take the form of Battery Event packets sent from the EC to the AP.

The number of battery slots in the system (and therefore the maximum number of batteries that can be installed) is reported via the Get System Configuration (Section 6.7.4). Most battery commands are specific to a given battery slot, and the slot number is encoded in the command. It is illegal to reference a slot number greater than the maximum reported by EC Capabilities.

If a command is directed to a legal slot number, but for which the battery is not currently present, a Status of Hardware Resource Temporarily Unavailable (see Table 2).

The Battery Event packet is documented in Section 7.2. The Configure Event Reporting commands control when Battery Event packets are sent. The same information can be polled by the AP via the Get Slot Status and Capacity Gauge command. Event reporting is initially disabled after reset.

Battery events can also trigger waking the AP from its sleep state. Further discussion of sleep control can be found in Section 6.4. The role of the Configure Wake command is described there.

The Set Remaining Capacity Alarm is used to set a threshold for detecting low battery power. When the remaining capacity falls below the threshold value, the Remaining Capacity Alarm bit of the Battery Slot Status is set to one. The Battery Slot Status can be obtained via the Get Slot Status and Capacity Gauge command or (if notifications are enabled via Enable Event Reporting) the Battery Event packet. Normally, these notifications are only sent when the remaining capacity crosses the threshold, but if the capacity is already below the threshold at the time when notifications are enabled, then the EC will proceed to send the notification.

The EC may not allow the system to operate once the battery capacity reaches some critical-low threshold; the AP can query this threshold value via the Get Critical Capacity command. Setting the Remaining Capacity Alarm threshold at or below the Critical Capacity threshold disables the alarm.

Battery capacity information can be reported in units of mAh or 10 mWh, depending on the setting specified via the Set Configuration command.

> **Note**: The EC is not responsible for converting user-specified settings when the capacity units change. Unless the AP re-specifies the settings in the new units, the EC will continue to use the original setting value but with the new units. For example, 5

mAh will become 5*10 mWh. This conversion issue affects the Set Remaining Capacity Alarm and Get Remaining Capacity Alarm commands.

The battery commands and sub-commands are tabulated below. The response for each sub-command is detailed in the ensuing subsections.

Table 11—Request Battery Information

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	N	
Data Byte 1 (Sub-Command)	7:5	Battery Slot Tag 0h—Battery Slot 0 1h—Battery Slot 1 2h—Battery Slot 2 3h—Battery Slot 3 Others—Reserved
	4:0	Battery Operations 00h Get Slot Status and Capacity Gauge 01h Get Voltage 02h Get Remaining Time to Empty 03h Get Current 04h Get Average Current 05h Get Averaging Time Interval 06h Get Remaining Capacity 07h Get Last Full Charge Capacity 08h Get Design Capacity 09h Get Critical Capacity 0ah Get Temperature 0bh Get Manufacturer Name 0ch Get Model 0dh Get Type 0eh Set Remaining Capacity Alarm 0fh Get Remaining Capacity Alarm 10h Set Configuration 11h Get Configuration 12h Configure Event Reporting 1dh Configure Wake Others Reserved

Data Byte 2 – Data Byte N		For sub-command 0eh, alarm threshold value For sub-command 10h, configuration setting For sub-command 12h, configuration setting For sub-command 1dh, configuration setting Otherwise, no data bytes

6.2.1. Get Slot Status and Capacity Gauge Response

Table 12—Get Slot Status and Capacity Gauge Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	00h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Battery Slot Status	See Table 13
Data Byte 4	Battery Capacity Gauge	Battery's relative remaining capacity in %

Table 13—Battery Slot Status

Status Bit(s)	Description
7:4	Reserved (0h)
3	Remaining Capacity Alarm 0b—No alarm 1b—Alarm is set
2:1	Charging state 00b—Battery is idle (self-discharging) 01b—Battery is being charged 10b—Battery is being discharged (powering the system) 11b—Reserved
0	Present State 0b—Battery is not present in the respective slot 1b—Battery is present in the respective slot

6.2.2. Get Voltage Response

Table 14—Get Voltage Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	01h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Present Voltage Bits 7-0	Battery's present voltage (16-bit unsigned value, in mV)
Data Byte 4	Present Voltage Bits 15-8	

6.2.3. Get Remaining Time to Empty Response

Table 15—Get Remaining Time to Empty Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	02h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Time to Empty Bits 7-0	Estimated remaining time to empty for discharging battery at present rate (16-bit unsigned value, in minutes) 0FFFFh if battery is not discharging
Data Byte 4	Time to Empty Bits 15-8	

6.2.4. Get Current Response

Table 16—Get Current Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	03h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Present Current Bits 7-0	Battery's present current at the terminals (16-bit signed value, in mA) 0 to 32,767 if charging 0 to -32,768 if discharging
Data Byte 4	Present Current Bits 15-8	

6.2.5. Get Average Current Response

Table 17—Get Average Current Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	04h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Rate Bits 7-0	Battery's average current at the terminals (16-bit signed value, in mA) 0 to 32,767 if charging 0 to -32,768 if discharging
Data Byte 4	Rate Bits 15-8	

6.2.6. Get Averaging Time Interval Response

Table 18—Get Averaging Interval Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	05h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Time Interval Bits 7-0	Time interval over which averaged values are computed (16-bit unsigned value, in msec).
Data Byte 4	Time Interval Bits 15-8	

6.2.7. Get Remaining Capacity Response

Table 19—Get Remaining Capacity Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	06h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Capacity Bits 7-0	Battery's remaining capacity (16-bit unsigned value, in *Capacity Units*)
Data Byte 4	Capacity Bits 15-8	

6.2.8. Get Last Full Charge Capacity Response

Table 20—Get Last Full Charge Capacity Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	07h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Capacity Bits 7-0	Battery's capacity when fully charged last time (16-bit unsigned value, in *Capacity Units*)
Data Byte 4	Capacity Bits 15-8	

6.2.9. Get Design Capacity Response

Table 21—Get Design Capacity Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	08h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Capacity Bits 7-0	Battery's design capacity (16-bit unsigned value, in *Capacity Units*)
Data Byte 4	Capacity Bits 15-8	

6.2.10. Get Critical Capacity Response

Table 22—Get Critical Capacity Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	09h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Capacity Bits 7-0	Battery's critical capacity (16-bit unsigned value, in *Capacity Units*)
Data Byte 4	Capacity Bits 15-8	

6.2.11. Get Temperature Response

Table 23—Get Temperature Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	0ah
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Temperature Bits 7-0	Battery's temperature (16-bit unsigned value, in 0.1°K)
Data Byte 4	Temperature Bits 15-8	

6.2.12. Get Manufacturer Response

Table 24—Get Manufacturer Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	N = 03h-20h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	0bh
Data Byte 2 (Status)	00h	Successful
Data Byte 3–Data Byte N	ASCII string	Manufacturer name is up to 30 characters (may not be null-terminated)

6.2.13. Get Model Response

Table 25—Get Model Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	N = 03h-20h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	0ch
Data Byte 2 (Status)	00h	Successful
Data Byte 3–Data Byte N	ASCII string	Battery model is up to 30 characters (may not be null-terminated)

6.2.14. Get Type Response

Table 26—Get Type Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	
Byte Count	N = 03h-20h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	0dh
Data Byte 2 (Status)	00h	Successful
Data Byte 3–Data Byte N	ASCII string	Battery type (commonly battery chemistry) is up to 30 characters (may not be null-terminated).

6.2.15. Set Remaining Capacity Alarm Request

Table 27—Set Remaining Capacity Alarm Request

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	0eh
Data Byte 3	Capacity Bits 7-0	Threshold for remaining capacity alarm (16-bit unsigned value, in *Capacity Units*)
Data Byte 4	Capacity Bits 15-8	

6.2.16. Set Remaining Capacity Alarm Response

Response to a Set Battery Low Capacity Alarm Request is an Ack packet.

6.2.17. Get Remaining Capacity Alarm Response

Table 28—Get Remaining Capacity Alarm Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	04h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	0fh
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Capacity Bits 7-0	Threshold for remaining capacity alarm (16-bit unsigned value, in *Capacity Units*)
Data Byte 4	Capacity Bits 15-8	

6.2.18. Set Configuration Request

Table 29—Set Configuration Request

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	02h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	10h
Data Byte 2	Configuration Setting	See Table 30

Table 30—Battery Configuration Settings

Battery Configuration Bit(s)	Description
7:1	Reserved (0b)
0	*Capacity Units* 0b capacity values are reported in units of mAh 1b capacity values are reported in units of 10 mWh Initial setting in mAh (0b).

6.2.19. Set Configuration Response

Response to a Set Battery Configuration Request is an Ack packet.

6.2.20. Get Configuration Response

Table 31—Get Configuration Response

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	03h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	11h
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Configuration Setting	See Table 30

6.2.21. Configure Event Reporting Request

Table 32— Enable Event Reporting Request

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	02h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	12h
Data Byte 2	Configuration Action	0h – Disable event reporting 1h – Enable event reporting
Data Byte 3	Battery Event Mask	See Table 33. Each mask bit value controls whether the Configuration Action is applied to the specified Battery Event Type. 0b – Do not apply Configuration Action 1b – Apply Configuration Action

Table 33—Battery Event Type

System State Bit(s)	Description
7:3	Reserved (00h)
2	Remaining Capacity Alarm
1	Charging State
0	Present State

6.2.22. Configure Event Reporting Response

Response to a Configure Event Reporting Request is an Ack packet.

6.2.23. Configure Wake Request

Table 34— Configure Wake Request

SMBus Protocol Byte	Description	Note
Command Byte	02h	Battery Information
Byte Count	03h	
Data Byte 1 (Sub-Command)	7:5	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
	4:0	1dh
Data Byte 2	Configuration Action	0h – Disable wake for event 1h – Enable wake for event
Data Byte 3	Battery Event Mask	See Table 33. Each mask bit controls whether the Configuration Action is applied to the corresponding Battery Event Type. 0b – Disable wake for event 1b – Enable wake for event

6.2.24. Configure Wake Response

Response to a Configure Wake Request is an Ack packet.

6.3. GPIO Control

This request is used to configure GPIO operation, allowing the AP to control devices connected to the EC's GPIO pins. The GPIO pins can be configured as inputs or outputs, and notifications can be sent to the AP whenever certain input signal changes occur.

Selection of the physical EC GPIO pins to be controlled by the AP, the capabilities of those pins, their initial configuration, the mapping of logical GPIO numbers to physical EC GPIO pins, and the mapping of logical signal levels to signal polarity at the physical pins is outside the scope of this document. These platform-specific details are left to the system integrator to define and the EC firmware developer to implement.

The EC reports the total number of GPIO pins controllable by the AP via the Get System Configuration request (Section 6.7.4). Legal logical GPIO numbers range from zero to one less than the total number of AP-controllable GPIO pins.

> **Note**: The maximum number of GPIO pins is effectively limited to 232; this is governed by the maximum payload length for an SMBus block read (32 bytes) and the maximum overhead for a vector-type command (3 bytes; Set Pin Request), thus limiting the bit vector length to 29 bytes or 232 bits.

The Configure Pin command is used to select the pin's operation mode as input, output, tri-state (high impedance), or unused. For input pins, a trigger type is also specified; whenever the trigger event occurs, a notification is sent to the AP, provided that event reporting is enabled (details below).

> **Note**: Setting pins as "unused" allows the EC to reclaim the pin for its own purposes; if also not needed by the EC, the pin and associated hardware logic can be powered down. If an unsupported pin configuration is specified, the Unsupported Configuration error (Table 2) is reported in the response and the pin configuration is not changed.

The output drive level of GPIO pins can be set using the Set Pin command. For GPIO pins configured as outputs, the new drive level goes into effect immediately. For non-output GPIO pins, the output drive level is latched and goes into effect if the pin is later reconfigured as an output.

> **Note**: The output drive levels are just logical values; the system integrator and/or EC firmware developer determines how these logical values map to output signal polarity (i.e., low versus high for push-pull pins, low versus high-impedance for open-drain pins).

The input state of each GPIO can also be read using the Get Pin command. For push-pull output pins, the reported value is the output drive level. For input pins and open-drain output pins, the reported value is the input signal level. For unused and tri-stated pins, the reported value is undefined.

Note: The input signal levels are just logical values; the system integrator and/or EC firmware developer determines how these logical values map to input signal polarity.

The EC can be configured to detect certain input events via the Event Trigger Type setting of the Configure Pin command. Edge triggered events are latched when they occur, allowing them to be reported later. The latch is cleared and re-enabled when the edge triggered event is eventually reported. Level triggered events are not latched; only the current ("live") state of a level trigger can be reported. Thus, while it's possible to have an unreported edge triggered event, there is no such thing as an unreported level triggered event.

If auto-reporting is enabled (by specifying Report Enable as 01h with the Configure Event Reporting command), the EC sends a GPIO Event packet when a trigger event occurs. For a given GPIO pin, each trigger event must be acknowledged (by the Acknowledge Event Report command) before the next one can be reported. Unnecessary acknowledge commands are ignored. Auto-reporting can be cancelled using the Configure Event Reporting command and setting Report Enable to 00h. The Disable Event Reporting command cancels auto-reporting. Auto-reporting is initially disabled after reset.

The AP can also explicitly query the EC for trigger events via the Get Event Report command. For edge triggered events the latched state is reported, whereas for level triggered events the current ("live") trigger state is reported. If no trigger is enabled for a given GPIO pin, then a zero (i.e., no trigger event detected) is reported.

If an unreported edge triggered event is already pending when event reporting is enabled, it is reported by sending a GPIO Event packet. Since there can be no unreported level trigger interrupts, the corresponding situation cannot arise for level trigger events.

GPIO trigger events can also waking the AP from its sleep state. Further discussion of sleep control can be found in Section 6.4. The role of the Configure Wake command is described there.

Many GPIO operations come in two flavors – scalar and vector. Scalar operations apply to a single GPIO pin, whereas vector operations apply to multiple GPIO pins. For scalar operations, the logical GPIO pin number is specified explicitly. For vector operations, the set of logical GPIO numbers is specified indirectly via a bit vector. For each logical GPIO number "N", the Nth bit of the bit vector is set to "1"; all other bits are set to "0".

The bits in the bit vector are numbered in increasing order from the least significant bit (LSB) to most significant bit (MSB) within each byte, and starting from the first byte of the bit vector. Thus, bit 0 of the vector is the LSB of the first byte, bit 7 is the MSB of the first byte, and bit 8 is the LSB of the 2nd byte, and so on. The number of bits in the bit vector does not have to match the number of GPIO pins on the EC (as reported by the Get System Configuration request, Section 6.7.4). If the bit vector is short, the missing

upper bits are assumed to be zero. If the bit vector is long, the excess upper bits must be ignored for requests and must be set to zero for responses.

Format of the Request and Response packets are given below.

Table 35— GPIO Control Requests

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	7:0	Scalar sub-commands 00h—Configure Pin 01h—Set Pin (scalar) 02h – Get Pin (scalar) 03h – Configure Event Reporting (scalar) 04h – Acknowledge Event Report (scalar) 06h – Get Event Report (scalar) 1dh – Configure Wake (scalar) Vector sub-commands 21h—Set Pin (vector) 22h – Get Pin (vector) 23h – Configure Event Reporting (vector) 24h – Acknowledge Event Report (vector) 26h – Get Event Report (vector) 3dh – Configure Wake (vector) Others Reserved
Data Byte 2 – Data Byte N		For all sub-commands, parameter data For scalar sub-commands, pin number For vector sub-commands, bit vector

6.3.1. Configure Pin Request

Table 36—Request GPIO Configure Pin (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	4	

Data Byte 1 (Sub-Command)	00h	Configure Pin (scalar)
Data Byte 2	7:5	Mode 00h—Input 01h—Output 02h – Tri-state 03h – Unused
	4:2	Event Trigger Type 00h -- none 01h -- rising edge 02h -- falling edge 03h -- any edge 04h -- lo level 05h -- hi level 06h – level change
	1:0	Pull Enable 00h – no pull enabled 01h – pull down enabled 10h – pull up enabled
Data Byte 3	7:6	Output Drive Type 00h – Push-pull 01h – Open drain
	5:5	Schmitt Trigger 00h – Disable 01h – Enable
Data Byte 4		Logical GPIO number

6.3.2. Configure Pin Response (scalar)

Response to a Configure Pin (scalar) Request is an Ack packet. The Invalid Configuration error code is returned if the EC cannot accommodate the requested pin configuration.

6.3.3. Set Pin Request (scalar)

Table 37—Request GPIO Set Pin (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	3	
Data Byte 1 (Sub-Command)	01h	Set Pin (scalar)
Data Byte 2		Pin Drive Level 00h Logical low 01h Logical high
Data Byte 3		Logical GPIO number

6.3.4. Set Pin Response (scalar)

Response to a Set Pin (scalar) Request is an Ack packet.

6.3.5. Set Pin Request (vector)

Table 38—Request GPIO Set Pin (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	21h	Set Pin (vector)
Data Byte 2		Pin Drive Level 00h Logical low 01h Logical high
Data Byte 3 – Data Byte N		Bit Vector of GPIO pins to set 0b Do not set pin level 1b Set pin level

6.3.6. Set Pin Response (vector)

Response to a Set Pin (vector) Request is an Ack packet.

6.3.7. Get Pin Request (scalar)

Table 39—Request GPIO Get Pin (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	2	
Data Byte 1 (Sub-Command)	02h	Get Pin (scalar)
Data Byte 2		Logical GPIO number

6.3.8. Get Pin Response (scalar)

Table 40—Response GPIO Get Pin (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	3	
Data Byte 1 (Sub-Command)	02h	Get Pin (scalar)
Data Byte 2 (Status)	00h	Successful
Data Byte 3		Pin Level 0b Logical low 1b Logical high For input and output pins, value indicates current logical level at the pin. For other pin configurations, value is undefined.

6.3.9. Get Pin Request (vector)

Table 41—Request GPIO Get Pin (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	22h	Get Pin (vector)
Data Byte 2 – Data Byte N		Bit Vector of GPIO pins to report 0b Do not report pin level 1b Report pin level

6.3.10. Get Pin Response (vector)

Table 42—Response GPIO Get Pin (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	22h	Get Pin (vector)
Data Byte 2 (Status)	00h	Successful
Data Byte 3 – Data Byte N		Bit Vector of Pin Levels 0b Logical low 1b Logical high For input and output pins, bit value indicates current logical level at the pin. For other pin configurations, bit value is undefined.

6.3.11. Configure Event Reporting Request (scalar)

Table 43—Request GPIO Configure Event Reporting (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	2	
Data Byte 1 (Sub-Command)	03h	Configure Event Reporting (scalar)
Data Byte 2		Report Enable 00h Disable reporting 01h Enable reporting
Data Byte 3		Logical GPIO number

6.3.12. Configure Event Reporting Response (scalar)

Response to a Configure Event Reporting (scalar) Request is an Ack packet.

6.3.13. Configure Event Reporting Request (vector)

Table 44—Request GPIO Configure Event Reporting (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	23h	Configure Event Reporting (vector)
Data Byte 2		Report Enable 00h Disable reporting 01h Enable reporting
Data Byte 2 – Data Byte N		Bit Vector of GPIO pins for which Report Enable is to be applied 0b Do not apply 1b Apply

6.3.14. Configure Event Reporting Response (vector)

Response to an Enable Event Reporting (vector) Request is an Ack packet.

6.3.15. Acknowledge Event Report Request (scalar)

Table 45—Request GPIO Acknowledge Event Report (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	2	
Data Byte 1 (Sub-Command)	04h	Acknowledge Event Report (scalar)
Data Byte 2		Logical GPIO number

6.3.16. Acknowledge Event Report Response (scalar)

Response to an Acknowledge Event Report (scalar) Request is an Ack packet.

6.3.17. Acknowledge Event Report Request (vector)

Table 46—Request GPIO Acknowledge Event Report (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	24h	Acknowledge Event Report (vector)
Data Byte 2 – Data Byte N		Bit Vector of GPIO pins for which event reports are to be acknowledged 0b Do not acknowledge 1b Acknowledge

6.3.18. Acknowledge Event Report Response (vector)

Response to an Enable Event Reporting (vector) Request is an Ack packet.

6.3.19. Get Event Report Request (scalar)

Table 47—Request GPIO Get Event Report (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	2	
Data Byte 1 (Sub-Command)	06h	Get Event Report (scalar)
Data Byte 2		Logical GPIO number

6.3.20. Get Event Report Response (scalar)

Table 48—Response GPIO Get Event Report (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	3	
Data Byte 1 (Sub-Command)	06h	Get Event Report (scalar)
Data Byte 2 (Status)	00h	Successful
Data Byte 3		Trigger Event Status 0b No trigger event detected 1b Trigger event detected

6.3.21. Get Event Report Request (vector)

Table 49—Request GPIO Get Event Report (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	26h	Get Event Report (vector)
Data Byte 2 – Data Byte N		Bit Vector of GPIO pins to report 0b Do not report trigger event 1b Report trigger event

6.3.22. Get Event Report Response (vector)

Table 50—Response Get Event Report (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	26h	Get Event Report (vector)
Data Byte 2 (Status)	00h	Successful
Data Byte 3 – Data Byte N		Bit Vector of Trigger Events detected 0b No trigger event detected 1b Trigger event detected

6.3.23. Configure Wake Request (scalar)

Table 51—Request GPIO Configure Wake (scalar)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	2	
Data Byte 1 (Sub-Command)	1dh	Configure Wake (scalar)
Data Byte 2		Wake Enable 00h Disable wake 01h Enable wake
Data Byte 3		Logical GPIO number

6.3.24. Configure Wake Response (scalar)

Response to a Configure Wake (scalar) Request is an Ack packet.

6.3.25. Configure Wake Request (vector)

Table 52—Request GPIO Configure Wake (vector)

SMBus Protocol Byte	Description	Note
Command Byte	03h	GPIO Control
Byte Count	N	
Data Byte 1 (Sub-Command)	3dh	Configure Wake (vector)
Data Byte 2		Wake Enable 00h Disable wake 01h Enable wake
Data Byte 2 – Data Byte N		Bit Vector of GPIO pins for which Wake Enable is to be applied 0b Do not apply 1b Apply

6.3.26. Configure Wake Response (vector)

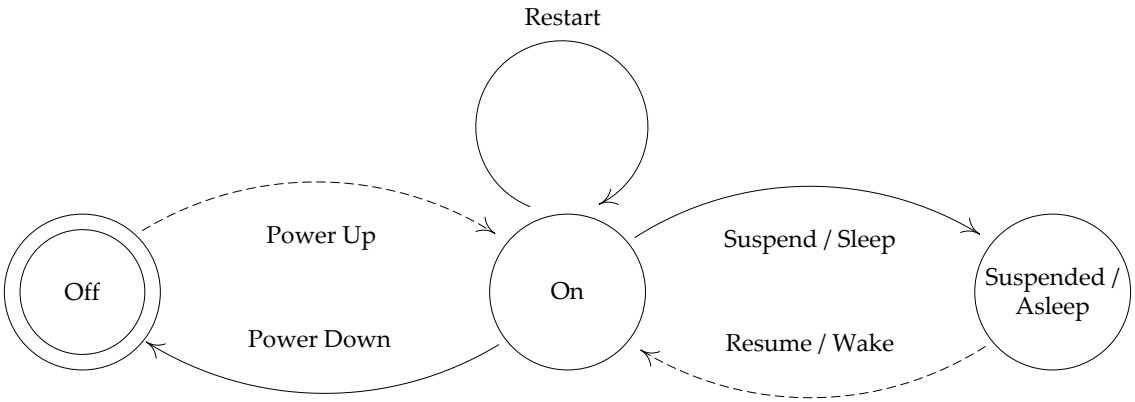
Response to a Configure Wake (vector) Request is an Ack packet.

6.4. Sleep State Control

The section describes the transition between power states, and the requests that drive the process. The low-level electrical signaling between the AP and the EC during these transitions is detailed in Section 3.

Figure 11 shows the supported power states and transitions for the AP. The solid lines indicate state transitions initiated by the AP and communicated to the EC via Request Packets sent over the SMBus. The dashed lines indicate state transitions initiated from outside the AP (but not necessarily by the EC) and communicated to the AP via side-band signals. For example, the EC can trigger the Resume / Wake transition by asserting the AP_WAKE# signal. The system architecture may include any number of wake sources in the system, but only discussion of the EC as a wake source is within the scope of this document. Triggering for the Power Up transition (regardless of its source) is also outside the scope of this document.

Figure 11—AP Power States and Power State Transitions



The AP initiates a power state change by disabling event reporting, as it may not be able to process (asynchronous) Event packets while preparing to enter the new power state. Table 53 lists the commands for enabling and disabling reporting for each event type. In addition, the Global Configure Event Reporting command can be used to enable and disable reporting for all events. Once reporting has been disabled for all EC functional units, the AP notifies the EC of the power state change by issuing an AP Power Down, AP Suspend, or AP Restart command. The EC sends a response for the command and then ceases further SMBus operations involving the AP.

Table 53—Event Reporting Commands

Event Type	Enable Reporting Command	Disable Reporting Command
System	Configure Event Reporting (Section 6.1)	Configure Event Reporting (Section 6.1)
Battery	Configure Event Reporting (Section 6.1)	Configure Event Reporting (Section 6.1)
GPIO	Configure Event Reporting (Section 6.3)	Configure Event Reporting (Section 6.3)
Keyboard	Enable (Section 6.5)	Disable (Section 6.5)
PS/2 Auxiliary Device	Auto-receive N Bytes (Section 6.6)	Cancel Auto-receive (Section 6.6)

For the Power Down and Restart commands, the EC now re-initializes its AP-visible state in preparation for the next AP boot cycle. In addition, the EC resets itself and reloads its firmware after the first Power Down following a firmware update attempt (Section 6.7.6). Depending on the system architecture, the Power Down and Restart commands may perform auxiliary operations such as shutting off the power to the AP, resetting the AP, or the like. Any such auxiliary operations are determined by the system designer and are outside the scope of this document.

For the Suspend command, the EC begins monitoring for designated events such that the AP can be awakened when needed. When a wake event is detected, the EC wakes the AP by asserting and holding the AP_WAKE# signal. The EC also latches the wake event for subsequent reporting to the AP. The wake events (if any) must be specified by the AP before issuing the Suspend command. In general, the same types of events that can be reported with an Event packet can also trigger wake-up of the AP. Each major functional unit in the EC firmware has a Configure Wake command for its events. Wake is disabled for all events after reset.

Asserting the AP_WAKE# signal causes the AP to wake up. When ready to resume communications with the EC, the AP initiates a normal data transfer, the first step of which is to assert the EC_REQUEST# signal. Once the EC sees this signal, it deasserts AP_WAKE# and resumes normal SMBus operations involving the AP. In order to see the EC_REQUEST# signal, the EC must continue to monitor the signal even when the AP is asleep (or powered down or restarting), unless the system architecture imposes some limitations on when EC_REQUEST# can be asserted. For example, if the EC controls power to the AP, then the AP won't be able to assert EC_REQUEST# until after being powered up by the EC; thus, the EC need not monitor EC_REQUEST# until after powering up the AP. Similarly, if the EC is the sole device capable of waking the AP, then the EC need not monitor the EC_REQUEST# signal until after it asserts AP_WAKE#. The Non-EC Wake Support capabilities bit (Table 70) must be set accordingly.

After resuming SMBus communications, the AP may issue any number of commands to the EC.

> **Note**: Event reporting is still disabled at this point, as the AP may not yet be ready to process (asynchronous) event packets. When ready, the AP restarts event reporting by issuing the event reporting commands in Table 53 and/or the Global Configure Event Reporting command. The EC will then report the latched wake event by sending the appropriate Event packet to the AP, provided that reporting is enabled for the type of event that triggered the wake.

It is left to the system designer and EC firmware implementer to determine whether non-wake events are to be detected while reporting is disabled or the AP is asleep, and reported later after reporting has been re-enabled. Similarly, the system designer specifies when wake event detection begins and end. Regardless of when detection begins, the EC cannot assert AP_WAKE# until after sending its response for the AP Suspend command.

The EC firmware should be designed to minimize power consumption while the AP is asleep or powered down, so far as is possible while carrying out its duty to detect wake events (if any are enabled) and monitor the `EC_REQUEST#` signal (if any devices other than the EC can wake the AP).

The power control commands are tabulated below. All of these commands return an Ack packet as response.

Table 54—Request Sleep Control Command

SMBus Protocol Byte	Description	Note
Command Byte	04h	Sleep control command
Byte Count	01h	
Data Byte 1 (Sub-Command)		00h Global Configure Event Reporting 01h AP Power Down 02h AP Suspend 03h AP Restart Others Reserved
Data Byte 2		For sub-command 00h, configuration setting Otherwise, no data bytes

6.4.1. Global Configure Event Reporting Request

Table 55— Global Configure Event Reporting Request

SMBus Protocol Byte	Description	Note
Command Byte	04h	Sleep control command
Byte Count	02h	
Data Byte 1 (Sub-Command)	00h	Global Configure Event Reporting
Data Byte 2		Report Events 00h – Disable reporting 01h – Enable reporting

6.4.2. Global Configure Event Reporting Response

Response to a Global Configure Event Reporting Request is an Ack packet.

6.5. Keyboard Control

This request is used to control the keyboard connected to the EC. The EC is expected to scan the key matrix, perform debouncing and phantom key processing, generate make/break scan codes, buffers the scan codes, and ultimately send each scan code to the AP as a Keyboard Event packet. The make/break scan codes are reported using Scan Code Set 1. The EC should perform no typematic processing; driver code running on the AP will supply any needed auto-repeat functionality.

In response to a Reset sub-command, the keyboard controller will load default settings, begin scanning, and send scan codes to the AP via Keyboard Event packets. A Disable sub-command stops keyboard scanning, loads default settings, and awaits further instructions. The Enable sub-command resumes scanning and reporting of scan codes after a Disable sub-command. Each of the sub-commands generates an Ack packet in response.

If the EC's output buffer (for scan codes) overflows, the EC will report a special scan code denoting the overflow condition. The special scan code is 0xff for Scan Code Set 1 and 0x00 for Scan Code Set 2. Any scan codes reported to the AP prior to the overflow marker must be correct (i.e., uncorrupted). The output buffer must be cleared when any sub-command is received, and no Keyboard Event packets can be sent while a sub-command is being processed.

> **Note**: When scanning is enabled, the response packet will always be sent prior to the first event packet. Similarly, when scanning is disabled, the last event packet will always be sent prior to the response packet.

On power-up, the EC will be in the "disable" state. That is, it will behave as though a Reset had been performed, immediately followed by a Disable. No Keyboard Event packets can be sent to the AP until a Reset or an Enable sub-command is received.

By default, scan codes are reported using the Scan Code Set 1 encoding. The Set Scan Code Set sub-command is supported for future expansion; currently, only Scan Set 1 is supported. The Get Scan Code Set sub-command reports the currently-selected scan code set in the payload of its Response Packet.

Keyboard events can also wake the AP from its sleep state. The wake can be triggered by any key or by a restricted set of special keys. It is left to the system designer and EC firmware implementer to determine which keys belong to the restricted set, if any. The EC can be configured to report the scan code for the key that triggered the wake. In this case, the scan code is placed in the EC's output buffer such that it will be reported to the AP after the next Enable command is received. Consequently, it is required that the Enable command not clear the EC's output buffer.

Finally, the system designer specifies whether scanning is terminated after the wake event has been detected, or whether normal scanning is resumed automatically. In the latter case, scan codes will be placed in the EC's output buffer, where they'll remain until

reporting is enabled via the subsequent Enable command. Further discussion of sleep control can be found in Section 6.4. The role of the Configure Wake command is described there.

Table 56—Request Keyboard Command

SMBus Protocol Byte	Description	Note
Command Byte	05h	KBC command
Byte Count	N	02h for Set LEDs and Set scan code set 01h otherwise
Data Byte 1 (Sub-Command)	7:0	ffh Reset f4h Enable f5h Disable f1h Get scan code set f0h Set scan code set edh Set LEDs 03h Configure Wake 04h Configure Wake Key Reporting Others Reserved
Data Byte 2		For sub-command f0h, scan code set 01h is the only supported value For sub-command edh, LED indicators 7:3 reserved, must be zero 2:2 scroll lock indicator (0=off, 1=on) 1:1 num lock indicator (0=off, 1=on) 0:0 caps lock indicator (0=off, 1=on) For sub-command 03h, configuration setting For sub-command 04h, configuration setting Otherwise, no data byte

6.5.1. Configure Wake Request

Table 57— Configure Wake Request

SMBus Protocol Byte	Description	Note
Command Byte	05h	KBC command
Byte Count	03h	
Data Byte 1 (Sub-Command)	03h	Configure Wake
Data Byte 2	Configuration Action	0h – Disable wake for event 1h – Enable wake for event
Data Byte 3	Keyboard Event Type Mask	See Table 59. Each bit value controls whether the Configuration Action is applied to the corresponding Keyboard Event Type. 0b – Disable wake for event 1b – Enable wake for event

Table 58—Keyboard Event Type

System State Bit(s)	Description
7:2	Reserved (00h)
1	Special Key Press
0	Any Key Press

6.5.2. Configure Wake Response

Response to a Configure Wake Request is an Ack packet.

6.5.3. Configure Wake Key Reporting Request

Table 59— Configure Wake Request

SMBus Protocol Byte	Description	Note
Command Byte	05h	KBC command
Byte Count	02h	
Data Byte 1 (Sub-Command)	04h	Configure Wake Key Reporting
Data Byte 2	Report Wake Key	Specifies whether scan code for wake key is placed in keyboard output buffer for reported to AP 00h – Do not report scan code 01h – Report scan code Default is 00h

6.5.4. Configure Wake Key Reporting Response

Response to a Configure Wake Key Reporting Request is an Ack packet.

6.6. Auxiliary Device Control

This request is used to control PS/2 devices connected to the EC, typically a touchpad or other pointer device (e.g., “eraser head”). Hot-plugging is not supported; all PS/2 devices are required to be present at system boot time and to remain permanently installed while the system is running.

Commands exist to read and write data to a specified PS/2 port. The send-command command sends one byte of data to the PS/2 port, then accepts N bytes of data from the PS/2 port and reports them to the AP. The receive command accepts N bytes of data from the PS/2 port and reports them to the AP. Any PS/2 bus errors encountered while carrying out these commands must be reported to the AP by setting the Status Field of the Response Packet. In particular, the EC must check for PS/2 bus time-outs for both of these commands and report time-out errors via the Status Field.

An auto-receive command allows streaming data from a PS/2 port to be read by the AP without polling. The auto-receive command causes the EC to read a user-specified number of bytes of data from the PS/2 port and send them to the AP. The data is transferred as a single Auxiliary Device Event packet to minimize SMBus bandwidth utilization and to aid in maintaining data stream synchronization. Once the data is sent to the AP, the cycle repeats automatically until canceled.

Auxiliary Device events can also be used to wake the AP from its sleep state. If wake is enabled, the EC enables the specified PS/2 port and waits for the attached PS/2 device to begin driving the bus. Once the PS/2 device begins driving, the EC immediately inhibits the bus and proceeds to wake the AP. The EC must inhibit the bus before transfer of the first byte of data is complete, thus forcing the PS/2 device to retransmit from the beginning when the PS/2 port is subsequently re-enabled (as commanded by the AP). Since no PS/2 data was received by the EC, none can be reported to the AP. Further discussion of sleep control can be found in Section 6.4. The role of the Configure Wake command is described there.

The AP will typically perform a sequence of read and write operations to identify the type of device attached to a given PS/2 port. Once the device has been identified, the AP will know the size of data packets transmitted by the device in streaming mode (for example, 3 bytes for a legacy PS/2 mouse). The AP will then configure the PS/2 device for streaming operation and invoke auto-receive (specifying the device’s packet size as an argument). Eventually, auto-receive may be cancelled if the AP needs to stop or reconfigure the PS/2 device.

On initial power-up and after most commands have been completed (see Table 60), the EC inhibits communication on each PS/2 interface. This will be referred to as the non-accepting state. The one exception is for the auto-receive command, which leaves the PS/2 interface in the accepting state. In the accepting state, the EC performs the following sequence of operations:

1. Enable PS/2 interface.

2. Receive N bytes of data from PS/2 device (N = user-specified value).
3. Inhibit PS/2 interface.
4. Send the data to the AP.
5. Wait for data transfer to complete.
6. Go to step 1.

If a parity error occurs during step 2 of the cycle, processing should continue as normal except that the error will be reported in step 4. In particular, the Error Flag must be set in the Command Field, and the Status Field must be populated with the proper status code (see Table 2). After the error has been reported, the accepting state cycle continues as if no error had occurred.

When a new command is received on a given PS/2 port, auto-receive will be cancelled on that port, if it is in effect. The new command cannot be delayed indefinitely. This means that, for example, step 2 in the accepting state cycle may have to be terminated early if the PS/2 device fails to transmit the expected amount of data. In this case, the partial data must be discarded (not sent to the AP) and the PS/2 interface inhibited. If, on the other hand, complete data is received by the EC, then steps 3-5 can be carried out normally. Either way, the accepting state cycle is now terminated, any final event packet has already been sent to the AP, and the PS/2 interface is in the non-accepting state.

At this point, the newly-received command can be processed and its resulting response packet sent back to the AP. The PS/2 interface will enter the accepting or non-accepting state, depending on the command.

> **Note**: When auto-receive is initiated, the response packet will always be sent prior to the first event packet. Similarly, when auto-receive is terminated, the last event packet will always be sent prior to the response packet.

Since Auxiliary Device Events are likely to be the most frequent packets on the system, it is important to minimize transfer overhead and reduce SMBus bandwidth utilization where possible. One optimization is to avoid sending the first byte of a 3-byte legacy PS/2 mouse sample. The first byte contains the button state and X/Y movement direction, both of which are likely to change slowly relative to the X/Y location data in bytes 2 and 3. The resulting 2-byte sample can be transferred as a fixed size packet which uses the SMBus write word operation, thus saving an additional byte of overhead (Packet Length). This optimization is enabled and disabled via the Set Compression sub-command. Compression only applies to auto-receive events. The first event packet will never be compressed, because there is no valid previous value for byte 1 yet.

If a PS/2 parity error or time-out error occurs, it must be reported in the corresponding response or event packet.

Table 60—Final PS/2 Interface State Per Command

Command	Final PS/2 State
Send command	Non-accepting
Receive N bytes	Non-accepting
Auto-receive N bytes	Accepting
Cancel auto-receive	Non-accepting

Auto-receive and cancel auto-receive commands return Ack packets; other commands return Response packets as documented in the following subsections.

Table 61—Request Auxiliary Device Command

SMBus Protocol Byte	Description	Note
Command Byte	06h	PS/2 Auxiliary Device command
Byte Count	N	
Data Byte 1 (Sub-Command)	7:6	PS/2 port select 0h PS/2 port 0 1h PS/2 port 1 2h PS/2 port 2 3h PS/2 port 3
	5:0	01h Send command (send one byte, then receive N bytes) 02h Receive N bytes 03h Auto-receive N bytes 04h Cancel auto-receive 05h Set compression 3dh Configure Wake Others Reserved
Data Byte 2		For sub-command 01h, data byte to write to device For sub-command 02h, N, number of bytes to receive (1 to 15) For sub-command 03h, N, number of bytes to receive (1 to 15) For sub-command 05h 0h Disable Compression 1h Enable Compression For sub-command 3dh, configuration action Otherwise, no data byte

Data Byte 3		For sub-command 01h, N, number of bytes to receive (0 to 15) For sub-command 3dh, auxiliary device event type mask Otherwise, no data byte

6.6.1. Send Command Response

Table 62—Read Byte Response

SMBus Protocol Byte	Description	Note
Command Byte	06h	PS/2 Auxiliary Device command
Byte Count	M	
Data Byte 1 (Sub-Command)	X1h	X = 0h for PS/2 port 0 X = 4h for PS/2 port 1 X = 8h for PS/2 port 2 X = ch for PS/2 port 3
Data Byte 2 (Status)	00h	Successful
Data Byte 3–Data Byte M		N data bytes read from device

6.6.2. Receive N Bytes Response

Table 63—Send Sample Response

SMBus Protocol Byte	Description	Note
Command Byte	06h	PS/2 Auxiliary Device command
Byte Count	M	
Data Byte 1 (Sub-Command)	X2h	X = 0h for PS/2 port 0 X = 4h for PS/2 port 1 X = 8h for PS/2 port 2 X = ch for PS/2 port 3
Data Byte 2 (Status)	00h	Successful
Data Byte 3–Data Byte M		N data bytes read from device

6.6.3. Configure Wake Request

Table 64— Configure Wake Request

SMBus Protocol Byte	Description	Note
Command Byte	06h	PS/2 Auxiliary Device command
Byte Count	02h	
Data Byte 1 (Sub-Command)	7:6	PS/2 port select 0h PS/2 port 0 1h PS/2 port 1 2h PS/2 port 2 3h PS/2 port 3
	5:0	3dh
Data Byte 2	Configuration Action	0h – Disable wake for event 1h – Enable wake for event
Data Byte 2	Auxiliary Device Event Type Mask	See Table 65. Each bit value controls whether the Configuration Action is applied to the corresponding Auxiliary Device Event Type. 0b – Do not apply Configuration Action 1b – Apply Configuration Action

Table 65—Auxiliary Device Event Type

System State Bit(s)	Description
7:2	Reserved (00h)
0	Any Auxiliary Device Event

6.6.4. Configure Wake Response

Response to a Configure Wake Request is an Ack packet.

6.7. System Control

This request is used to configure and control the EC itself, rather than any peripherals connected to the EC.

Table 66—Request System Control

SMBus Protocol Byte	Description	Note
Command Byte	07h	System command
Byte Count	N	
Data Byte 1 (Sub-Command)	7:0	00h Reset EC 01h Self Test 02h No-op 10h Get EC Interface Spec Version 11h Get System Capabilities 12h Get System Configuration 14h Get EC Product Name 15h Get EC Firmware Version 20h Initialize Generic Configuration 21h Send Generic Configuration Bytes 22h Finalize Generic Configuration 30h Initialize firmware update 31h Send firmware bytes 32h Finalize firmware update 33h Poll firmware update 40h Get firmware size 41h Read firmware bytes Others Reserved
Data Byte 2–Data Byte N		No data for sub-commands 00h-15h, 20h, 22h, 30h, 32h, 33h, 40h, 41h

6.7.1. Reset EC

This command triggers a full reset of the EC. The EC must send an Ack Packet before performing the actual reset.

The AP is responsible for disabling events and draining Response Packet queue before requesting an EC reset. If the AP fails to do so, the EC is permitted to cancel any pending packets waiting to be sent to the AP such that the reset command will not be stalled indefinitely waiting for the Ack Packet to be sent to the AP.

6.7.2. Self Test

The command initiates a self-test on the EC. Implementation of the self test is EC vendor specific. After the self test has completed, the EC must be restored to its power-on reset state, with the exception that the EC is free to initiate SMBus transfers with the AP (it doesn't have to wait until the AP asserts `EC_REQUEST#`). The EC must send an Ack Packet reporting the success or failure of the self test.

Since the self test will generally cause any pre-existing system state information to be lost, the self test is only allowed to be performed immediately after EC reset. EC Capabilities requests are also permitted before performing a self test.

6.7.3. No Operation (No-op)

This command causes no side effects other than to return an Ack Packet. It can be used by the AP to verify that the EC is still operating and responsive.

6.7.4. Get Capabilities

These commands request the capabilities and configuration of the EC and its firmware.

6.7.4.1. Get EC Interface Spec Version Response

Table 67—Response Get EC Interface Spec Version

SMBus Protocol Byte	Description	Note
Command Byte	07h	System command
Byte Count	03h	
Data Byte 1 (Sub-Command)	10h	Get EC Interface Spec Version
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Version Number (10h = version 1.0)	Version of this specification EC is compliant with: the major version is specified in the high nibble, the minor version in the low nibble. For this revision of the spec, the Version Number is 10h.

6.7.4.2. Get System Capabilities Response

Table 68—Response Get System Capabilities

SMBus Protocol Byte	Description	Note
Command Byte	07h	System command
Byte Count	07h	
Data Byte 1 (Sub-Command)	11h	Get System Capabilities
Data Byte 2 (Status)	00h	Successful
Data Byte 3	GPIO allocation	Number of EC GPIOs controllable by Tegra
Data Byte 4	System Capabilities Bits 7-0	See Table 69
Data Byte 5	System Capabilities Bits 15-8	
Data Byte 6	OEM System Capabilities Bits 7-0	See Table 70
Data Byte 7	OEM System Capabilities Bits 15-8	

Table 69—System Capabilities Bits

System Capabilities Bit(s)	Description
15:3	Reserved (0b)
2	Fixed-Size Event Packet support 0b – Not supported 1b -- Supported
1	Non-EC Wake support (as per Section 6.4) 0b – Not supported 1b -- Supported
0	Runtime Generic Configuration support (as per Section 6.7.5) 0b – Not supported 1b -- Supported

Table 70—OEM System Capabilities Bits

OEM System Capabilities Bit(s)	Description
15:0	OEM Defined

6.7.4.3. Get System Configuration Response

Table 71—Response Get System Configuration

SMBus Protocol Byte	Description	Note
Command Byte	07h	System command
Byte Count	06h	
Data Byte 1 (Sub-Command)	12h	Get System Configuration
Data Byte 2 (Status)	00h	Successful
Data Byte 3	System Configuration Bits 7-0	See Table 72
Data Byte 4	System Configuration Bits 15-8	
Data Byte 5	OEM System Configuration Bits 7-0	See Table 73
Data Byte 6	OEM System Configuration Bits 15-8	

Table 72—System Configuration Bits

System Configuration Bit(s)	Description
15:6	Reserved (0b)
5:4	Number of PS/2 Ports 0h—One Port 1h—Two Ports 2h—Three Ports 3h—Four Ports
3:0	Number Battery Slots (= maximum number of batteries in the system) 0h—No Battery Slots 1h—One Battery Slot 0 2h—Two Battery Slots 1 and 2 3h—Three Battery Slots 0, 1 and 2 4h—Four Battery Slots 0, 1, 2 and 3 Others—Reserved

Table 73—OEM System Configuration Bits

OEM System Configuration Bit(s)	Description
15:0	OEM Defined

6.7.4.4. Get EC Product Name Response

This command returns a unique string that identifies the EC system configuration, which encompasses the EC manufacturer, EC model number, connectivity, and other configuration parameters. Besides general reporting purposes, this command is used during the firmware update process to determine whether a new firmware version is compatible with the existing system configuration. See Section 6.7.6 for more details.

Table 74—Get EC Product Name Response

SMBus Protocol Byte	Description	Note
Command Byte	07h	System command
Byte Count	N = 03h-20h	
Data Byte 1 (Sub-Command)	14h	Get EC Product Name
Data Byte 2 (Status)	00h	Successful
Data Byte 3–Data Byte N	ASCII string	EC product name is up to 30 characters (may not be null-terminated)

6.7.4.5. Get EC Firmware Version Response

Table 75—Get EC Firmware Version Response

SMBus Protocol Byte	Description	Note
Command Byte	07h	System command
Byte Count	N = 03h-20h	
Data Byte 1 (Sub-Command)	02h	Get EC Firmware Version
Data Byte 2 (Status)	15h	Successful
Data Byte 3	Minor Version bits 7-0	Minor Version Number
Data Byte 4	Minor Version bits 15-8	
Data Byte 5	Major Version bits 7-0	Major Version Number
Data Byte 6	Major Version bits 15-8	

6.7.5. Generic Configuration

The EC may optionally support runtime download of configuration information from the AP, as indicated by the Runtime Generic Configuration field reported by the Get System Capabilities command (see Table 69). For example, the layout of a key matrix as well as the scan code value to be reported for each key could be specified at runtime using this mechanism. This command provides a transport mechanism for such information, but does not specify the content or internal format of the information. The data content and internal format are defined by the EC vendor. If supported, this command must be permitted at any time after EC reset and before first use of the functionality being configured.

The configuration data consists of a header and a body. The header is used by the AP to identify and validate the configuration information; only the body data is actually sent to the EC. The format and content of the header are defined below. The header contains a Configuration ID parameter, which identifies the type of configuration data contained in the body. The Configuration ID value from the header is passed to the EC as an argument to the Initialize Generic Configuration command.

The format and content of the body are defined by the EC vendor and are opaque to the AP, except that the body must contain a trailing CRC-32 checksum value. The checksum

is computed using the CRC-32 algorithm from IEEE 802.3 ($x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1$).

The EC vendor is responsible for providing tools and documentation to assist system integrators in generating the configuration information (both content and format). The vendor may wish to leverage pre-existing tools that only address the opaque body content portion of the configuration information. In this case, the EC vendor is additionally responsible to provide tools and documentation to assist system integrators in transforming the raw body content into the format needed by the Generic Configuration commands.

The header contents are detailed in Table 76. The header contains a trailing CRC-32 checksum (just like the body), allowing the AP to validate the integrity of the header contents. Integrity is checked by computing a running checksum that covers the header contents, excluding the trailing checksum bytes. If the running checksum value is not equal to the trailing CRC value, then integrity has been compromised. If the header is found to be intact, then the integrity of the body can also be checked in a similar manner.

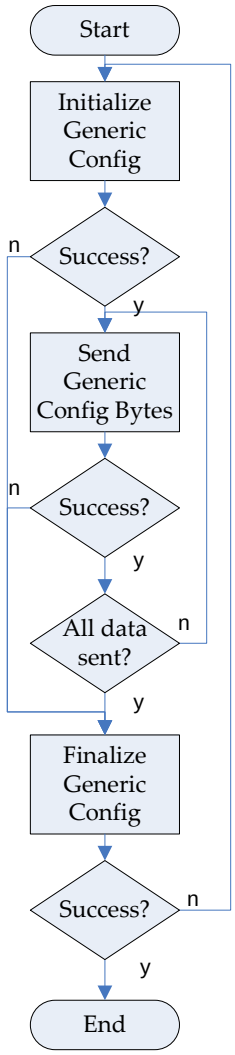
The remaining information in the header can be used by the AP to identify the configuration data and to decide whether to proceed with the update. Specifically, the Get EC Interface Spec Version, Get EC Product Name, and Get EC Firmware Version commands provide useful information that the AP can then compare with the header contents. In general, the EC Interface Spec Version and EC Product Name in the header must match the corresponding values reported by the currently-running EC firmware; otherwise, the AP will reject the configuration information.

Configuration information is downloaded to the EC in a 3-stage process – initialize, send bytes, and finalize – as illustrated in Figure 12. First, the Initiate Generic Configuration command is sent. This prepares the EC to receive the configuration data and also resets its running checksum value. Next, the body contents are transferred to the EC via a series of Send Generic Configuration Bytes commands. Each such command reports the cumulative number of bytes received by the EC. As bytes are received, the EC updates its running checksum value. Once all the body data (including trailing checksum) have been transferred in this manner, the AP performs a Finalize Generic Configuration command. This triggers the EC to perform any final checks on the configuration data. One required check is for the EC to verify that the running checksum value is zero, reporting a Status of Checksum error otherwise. The EC may optionally perform additional checks, since it has greater knowledge of the internal format of the body contents than does the AP. If the EC detects no error, it reports a Status of Success to the AP.

Table 76—Generic Configuration Package Header

Byte Number	Description	Note
0 – 3	Magic Number	ASCII string “cnfg”; not null-terminated
4	EC Interface Spec Version	Major/minor version; same format as in Get EC Interface Spec Version Response, Section 6.7.4.1
5	Reserved	Must be 00h
6 – 35	EC Product Name	ASCII string; same format as in Get EC Product Name Response, Section 6.7.4.4
36 – 39	EC Firmware Version	Major/minor version; same format as in Get EC Firmware Version Response, Section 6.7.4.5
40 – 43	Configuration ID	OEM-defined value specifying the type of configuration data contained in this package.
44 – 47	Body Length	Length of opaque data including its trailing checksum; first byte is least significant, last byte is most significant
48 – 51	CRC	Checksum computed over the above header data; first byte is least significant, last byte is most significant

Figure 12—Generic Configuration Procedure



6.7.5.1. Initialize Generic Configuration Request

Table 77—Request Initialize Generic Configuration

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	5	
Data Byte 1 (Sub-Command)	20h	Initialize Generic Configuration
Data Byte 2	Configuration ID Bits 7 - 0	Configuration ID OEM-defined value specifying the type of configuration data to follow.
Data Byte 3	Configuration ID Bits 15 - 8	
Data Byte 4	Configuration ID Bits 23 - 16	
Data Byte 5	Configuration ID Bits 31 - 24	

6.7.5.2. Initialize Generic Configuration Response

Response to an Initialize Generic Configuration request is an Ack packet.

6.7.5.3. Send Generic Configuration Bytes Request

Table 78—Request Send Generic Configuration Bytes

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	N	
Data Byte 1 (Sub-Command)	21h	Send Generic Configuration Bytes
Data Byte 2 – Data Byte N		Generic configuration data bytes

6.7.5.4. Send Generic Configuration Bytes Response

Table 79—Response Send Generic Configuration Bytes

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	6	
Data Byte 1 (Sub-Command)	21h	Send Generic Configuration Bytes
Data Byte 2 (Status)		00h – Successful 08h -- Checksum error
Data Byte 3	Bytes Received Bits 7 - 0	Cumulative Number of Bytes Received First byte is least significant; last byte is most significant.
Data Byte 4	Bytes Received Bits 15 - 8	
Data Byte 5	Bytes Received Bits 23 - 16	
Data Byte 6	Bytes Received Bits 31 - 24	

6.7.5.5. Finalize Generic Configuration Request

Table 80—Request Finalize Generic Configuration

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	1	
Data Byte 1 (Sub-Command)	22h	Finalize Generic Configuration

6.7.5.6. Finalize Generic Configuration Response

Table 81—Response Finalize Generic Configuration

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	2	
Data Byte 1 (Sub-Command)	22h	Finalize Generic Configuration
Data Byte 2 (Status)		00h -- Successful 08h -- Checksum error 09h -- Device write error 0bh – Data overflow error 0ch – Data underflow error

6.7.6. Firmware Update

This command allows the Tegra AP to download and install new EC firmware. The AP sees the EC firmware only as a contiguous block of data. The EC manages all details as to where and how the EC firmware data is stored, whether a backup copy of the firmware is maintained, etc.

Firmware update takes place in five stages, as shown in Figure 13:

1. The AP notifies the EC to prepare for firmware update by issuing the Initialize Firmware Update command.

2. The AP sends the firmware image to the EC through a series of Send Firmware Bytes commands.

3. The AP completes the firmware update by issuing the Finish Firmware Update command.

4. The AP optionally reads back the new firmware image from the EC. See Section 6.7.7 for details.

5. The AP powers down by issuing the Configure Global Event Reporting command to disable event reporting (optional), followed by the AP Power Down command (not optional). The EC then resets itself and begins executing the new firmware image in time for the next system boot.

The EC is free to reject any extraneous commands received between steps 1 and 5 above, by responding with an Invalid State error. The EC is also free to stop reporting events during the firmware update process. These restrictions are intended to limit the footprint of the EC-side firmware update code, such that the code can fit entirely in the EC's onboard memory. In this way, firmware updates can be supported even for EC architectures that normally execute directly out of the firmware data store.

The firmware update must be distributed by the EC firmware developer as a package consisting of a header and a body. The header is used only by the AP and contains information that identifies the update. The AP ultimately strips off the header and sends only the body to the EC. The EC must write the entire body contents to its data store (including the trailing checksum value, discussed below).

The header and body each contain a trailing checksum, which can be used to verify their integrity. The checksums are computed using the CRC-32 algorithm from IEEE 802.3 ($x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1$). Aside from the trailing checksum, the body contents are entirely opaque to the AP.

The header contents are detailed in Table 82. After validating the header, the AP can compare its contents with information reported by the currently-running firmware image before deciding whether to proceed with the update. Specifically, the Get EC Interface Spec Version, Get EC Product Name, and Get EC Firmware Version commands are useful for this purpose. In general, the EC Interface Spec Version and EC Product Name in the header must match the corresponding values reported by the currently-running EC firmware; otherwise, the AP will reject the new firmware image.

Once the AP decides to proceed with the update, it issues an Initialize Firmware Update command. This command prepares the EC to update its firmware store. The EC initializes its running CRC value and its counter of firmware data bytes received.

Note: The EC must respond to the AP within the communication time-out period specified in this document (Section 8). If the storage device for the EC firmware is slow, though, storage operations performed by the EC may not finish before a response is due to the AP. To accommodate this situation, the AP enters a polling loop after receiving a response from the EC (provided that the EC doesn't report an error). The AP issues the Poll Firmware Update command, to which the EC reports that it is ready, busy, or has encountered an error. Polling continues as long as the EC reports busy. Whenever the EC reports ready or that it has encountered an error, polling terminates. Unlike the AP, the EC has detailed knowledge of the particular storage device installed in the system. The EC is encouraged to make use of this information to report an error if the storage device does not respond within a suitable period of time.

Next, the AP begins sending the body contents to the EC using the Send Firmware Bytes command. With this command, the Tegra AP transfers between 1 and 30 bytes of firmware data to the EC. The EC maintains a running count of the number of data bytes received from the AP, which it reports back to the AP in the Send Firmware Bytes response packet. The AP can use this information to determine whether any packets have been dropped. The EC must also compute a running CRC value over the cumulatively received data (excluding the final 4 bytes received, i.e., the trailing CRC value), which will be used to verify the integrity of the received firmware data.

The EC may optionally buffer the received data before writing it to the data store. If writing to the data store, it is the EC's responsibility to verify the correctness of the written data before sending a response back to the AP. If the write or subsequent verification fails, the EC will report a Status value of Device Write Error. After the first write error, the EC is free to fail any subsequent Send Firmware Bytes commands.

Once the AP has sent all of the body contents (including the trailing CRC) it issues the Finalize Firmware Update command to complete the update process. If any errors were reported by the preceding Send Firmware Bytes commands, Finalize Firmware Update must also report an error; otherwise, the EC proceeds to write any remaining buffered data to its data store and check the cumulative CRC value against the trailing CRC value (final 4 bytes of the downloaded data). If the CRC values do not match, then a transmission error has occurred and the EC reports a Status of Checksum Error.

Finally, if the amount of data received by the EC (as reported in its response to the final Send Firmware Bytes command) doesn't match the size of the body, then the update has failed regardless of the Status reported by Finalize Firmware Update.

Finally, the AP again polls the EC using the Poll Firmware Update command, allowing the EC extra time to complete any final storage operations.

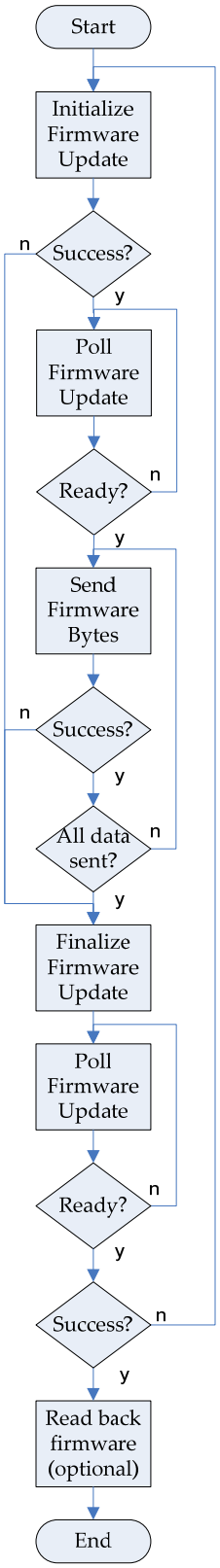
The AP can now optionally read back the firmware image (specifically, the body contents including trailing CRC), presuming the update succeeded. The AP is free to return to Initialize Firmware Update and repeat the cycle from the beginning, regardless of whether the previous attempt succeeded or failed. Finally, whenever a firmware update

is attempted (whether it succeeds or not), the EC is responsible for resetting itself after the next AP Power Down command so that the new firmware will be loaded and executed when the system is subsequently booted up.

Table 82—Firmware Update Package Header

Byte Number	Description	Note
0 – 3	Magic Number	ASCII string “updt”; not null-terminated
4	EC Interface Spec Version	Major/minor version; same format as in Get EC Interface Spec Version Response, Section 6.7.4.1
5	Reserved	Must be 00h
6 – 35	EC Product Name	ASCII string; same format as in Get EC Product Name Response, Section 6.7.4.4
36 – 39	EC Firmware Version	Major/minor version; same format as in Get EC Firmware Version Response, Section 6.7.4.5
40 – 43	Body Length	Length of opaque data including its trailing checksum; first byte is least significant, last byte is most significant
44 – 47	CRC	Checksum computed over the above header data; first byte is least significant, last byte is most significant

Figure 13—Firmware Update Procedure



6.7.6.1. Initialize Firmware Update Request

Table 83—Request Initialize Firmware Update

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	1	
Data Byte 1 (Sub-Command)	30h	Initialize Firmware Update

6.7.6.2. Initialize Firmware Update Response

Response to an Initialize Firmware Update request is an Ack packet.

6.7.6.3. Send Firmware Bytes Request

Table 84—Request Send Firmware Bytes

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	N	
Data Byte 1 (Sub-Command)	31h	Send Firmware Bytes
Data Byte 2 – Data Byte N		Firmware data bytes

6.7.6.4. Send Firmware Bytes Response

Table 85—Response Send Firmware Bytes

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	4	
Data Byte 1 (Sub-Command)	31h	Send Firmware Bytes
Data Byte 2 (Status)		00h – Successful 08h -- Checksum error 09h -- Device write error 0ch – Data underflow error
Data Byte 3	Bytes Received Bits 7 - 0	Cumulative Number of Bytes Received First byte is least significant; last byte is most significant.
Data Byte 4	Bytes Received Bits 15 - 8	
Data Byte 5	Bytes Received Bits 23 - 16	
Data Byte 6	Bytes Received Bits 31 - 24	

6.7.6.5. Finalize Firmware Update Request

Table 86—Request Finalize Firmware Update

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	1	
Data Byte 1 (Sub-Command)	32h	Finalize Firmware Update

6.7.6.6. Finalize Firmware Update Response

Table 87—Response Finalize Firmware Update

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	2	
Data Byte 1 (Sub-Command)	32h	Finalize Firmware Update
Data Byte 2 (Status)		00h -- Successful 08h -- Checksum error 09h -- Device write error 0bh – Data overflow error 0ch – Data underflow error

6.7.6.7. Poll Firmware Update Request

Table 88—Request Poll Firmware Update

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	1	
Data Byte 1 (Sub-Command)	33h	Poll Firmware Update

6.7.6.8. Poll Firmware Update Response

Table 89—Response Poll Firmware Update

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	3	
Data Byte 1 (Sub-Command)	33h	Poll Firmware Update
Data Byte 2 (Status)		00h -- Successful 08h -- Checksum error 09h -- Device write error 0bh – Data overflow error 0ch – Data underflow error
Data Byte 3		EC ready flag 00h EC busy 01h EC ready

6.7.7. Firmware Read

These commands enable the AP to read a firmware image installed on the EC. The EC must provide the data in the exact form as used by the Send Firmware Bytes, such that a read followed by an update (write) will return the system exactly back to its starting

state. The read-back firmware image must contain the same trailing checksum value as originally provided by the AP when the image was installed. This enables the AP to verify the integrity of the read-back image.

If no firmware update has been attempted before the read attempt, the EC will provide the currently-executing firmware image. If performed after a successful firmware update, the EC will provide the newly-downloaded firmware image. Reading of the firmware image is not supported following an unsuccessful update; the data returned by the EC is undefined in this case.

To begin reading the firmware image, the AP issues a Get Firmware Size command. This returns the size of the firmware image in bytes and also resets the EC's read pointer.

The AP then begins reading the firmware data by invoking the Read Firmware Bytes command, which returns between 1 and 30 bytes of data. The response must contain at least one byte of firmware data, unless the entire firmware image has already been transferred. In this latter case, the EC returns zero data bytes along with a Status of Data Underflow. If a read error occurs, the EC reports a Status of Device Read Error; this error is sticky and remains in effect until the next Get Firmware Size command is issued.

6.7.7.1. Get Firmware Size Request

Table 90—Request Get Firmware Size

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	1	
Data Byte 1 (Sub-Command)	40h	Get Firmware Size

6.7.7.2. Get Firmware Size Response

Table 91— Get Firmware Size Response

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	6	
Data Byte 1 (Sub-Command)	40h	Get Firmware Size
Data Byte 2 (Status)	00h	Successful
Data Byte 3	Firmware Size Bits 7-0	Firmware image size in bytes First byte is least significant; last byte is most significant.
Data Byte 4	Firmware Size Bits 15-8	
Data Byte 5	Firmware Size Bits 23-16	
Data Byte 6	Firmware Size Bits 31-24	

6.7.7.3. Read Firmware Bytes Request

Table 92—Request Read Firmware Bytes

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	1	
Data Byte 1 (Sub-Command)	41h	Read Firmware Bytes

6.7.7.4. Read Firmware Bytes Response

Table 93—Response Read Firmware Bytes

SMBus Protocol Byte	Description	Note
Command Byte	07h	System Control
Byte Count	N	
Data Byte 1 (Sub-Command)	41h	Read Firmware Bytes
Data Byte 2 (Status)		00h – Successful 08h – Checksum error 0ah – Device read error
Data Byte 3 – Data Byte N		Firmware data bytes

7. Asynchronous Event Packets

The supported Event packets sent from the EC to the AP are summarized in the sections below.

7.1. System Event

The System Event is sent when there is a change in the System State Bits.

Table 94—System Event

SMBus Protocol Byte	Description	Note
Command Byte	c5h	System Event, variable length
Byte Count	02h	
Data Byte 1	System State Bits 7-0	See Table 6
Data Byte 2	System State Bits 15-8	
Data Byte 3	OEM System State Bits 7-0	See Table 7
Data Byte 4	OEM System State Bits 15-8	

7.2. Battery Event

The Battery Event is sent when there is a change in the Battery Slot Status Bits. The fixed-length Event packet format is shown below; the variable-length format is also supported.

Table 95—Battery Event, Fixed Length

SMBus Protocol Byte	Description	Note
Command Byte	a8h	Battery Event, fixed length
Data Byte 1	Battery Slot	0h for Battery in Slot 0 1h for Battery in Slot 1 2h for Battery in Slot 2 3h for Battery in Slot 3
Data Byte 2	Battery Slot Status	See Table 13

7.3. Keyboard Events

The Keyboard Event is sent when key events are enabled and when a key press event has occurred (see Section 6.5). Currently, each Event packet contains the complete scan code for exactly one key press event.

Since Keyboard Events occur relatively frequently, it is important to minimize SMBus bandwidth utilization where possible by eliminating data transfer overhead. As such, there is a fixed length packet format that allows a 1-byte scan code to be transferred with one byte of overhead (rather than three bytes of overhead for the variable length format). Since 1-byte scan codes are the common case for Scan Set 1 scan codes, using fixed length packets results in a significant bandwidth savings. The variable length format provides for full flexibility in dealing the with the multi-byte Scan Set 1 scan codes, which occur less frequently.

For multi-byte scan codes, the first byte of the scan code is reported in the first byte of the payload, and so on. For example, the Scan Set 1 make code for the left arrow key is 0xe0 0x4b, so the first byte of the payload would be 0xe0 and the second byte of the payload would be 0x4b.

Table 96—Keyboard Event, Variable Length

SMBus Protocol Byte	Description	Note
Command Byte	c0h	Keyboard Event, variable length
Byte Count	N	
Data Byte 1-N (Payload)		Scan code for one key; see Section 6.5 for a discussion of scan code formats

Table 97—Keyboard Event, Fixed Length (1-Byte Scan Code)

SMBus Protocol Byte	Description	Note
Command Byte	80h	Keyboard Event, fixed length
Data Byte 1 (Payload)		Single-byte scan code for one key; see Section 6.5 for a discussion of scan code formats

Table 98—Keyboard Event, Fixed Length (2-Byte Scan Code)

SMBus Protocol Byte	Description	Note
Command Byte	a0h	Keyboard Event, fixed length
Data Byte 1 (Payload)		Low-order byte of two-byte scan code for one key; see Section 6.5 for a discussion of scan code formats
Data Byte 2 (Payload)		High-order byte of two-byte scan code for one key; see Section 6.5 for a discussion of scan code formats

7.4. Auxiliary Device Events

The Auxiliary Device Event is sent when auto-receive is enabled for a PS/2 port and the attached PS/2 device is transmitting a stream of data (see Section 0). Currently, each Event packet contains exactly one complete sample from the PS/2 device, where sample-size is specified by the AP.

Since Auxiliary Device Events are the most frequent events being generated on the system, it is important to minimize SMBus bandwidth utilization where possible by eliminating data transfer overhead. As such, there is a fixed length packet format that allows a 2-byte sample to be transferred with one byte of overhead (rather than three bytes of overhead for the variable length format). Using the fixed length format can result in significant SMBus bandwidth utilization savings for legacy PS/2 mice when compression is enabled. The variable length format provides for full flexibility when compression is disabled, the first byte of the sample changes value, or the sample size is larger than 3 bytes.

Finally, both variable length and fixed length error events exists to notify the AP of communication or timing error while auto-receiving data samples.

Table 99—Auxiliary Device Event, Variable Length

SMBus Protocol Byte	Description	Note
Command Byte	cXh	Auxiliary Device Event, variable length X = 1h for PS/2 port 0 X = 2h for PS/2 port 1 X = 3h for PS/2 port 2 X = 4h for PS/2 port 3
Byte Count	N	
Data Byte 1–N (Payload)		N data bytes, where N is specified as the argument to auto-receive sub-command

Table 100—Auxiliary Device Event, Fixed Length (2-Byte Payload)

SMBus Protocol Byte	Description	Note
Command Byte	aXh	Auxiliary Device Event, variable length X = 1h for PS/2 port 0 X = 2h for PS/2 port 1 X = 3h for PS/2 port 2 X = 4h for PS/2 port 3
Data Byte 1 (Payload)		Device sample byte
Data Byte 2 (Payload)		Device sample byte

Table 101—Auxiliary Device Error Event, Fixed Length

SMBus Protocol Byte	Description	Note
Command Byte	8Xh	Auxiliary Device Event, fixed length with error X = 1h for PS/2 port 0 X = 2h for PS/2 port 1 X = 3h for PS/2 port 2 X = 4h for PS/2 port 3
Data Byte 1 (Status)		Error code information

7.5. GPIO Events

The GPIO Events are sent when trigger auto-reporting is enabled (see Enable Event Reporting, Sections 6.3.11 and 6.3.13) and a trigger event is detected on a GPIO pin. The GPIO Event packets come in scalar and vector flavors. Note that all the fixed-length and variable-length variants are supported, even if not explicitly shown here.

EC firmware implementers are only required to support one GPIO Event packet type, though SMBus bandwidth can be optimized by supporting both types and selecting between them based on the number of outstanding trigger events to be reported.

Table 102—GPIO Event (scalar), Fixed Length

SMBus Protocol Byte	Description	Note
Command Byte	86h	GPIO Event (scalar), fixed length
Data Byte 1		Logical GPIO number of triggered pin

Table 103—GPIO Event (vector), Variable Length

SMBus Protocol Byte	Description	Note
Command Byte	C7h	GPIO Event (vector), variable length
Byte Count	N	
Data Byte 1 – Data Byte N		Bit Vector of Trigger Events detected 0b No trigger event detected 1b Trigger event detected

8. EC Behavioral Rules

This section describes EC behavioral requirements and recommendations for proper communication with Tegra AP over SMBus interface.

After EC power-on reset, the EC must be ready to receive requests from the AP; however, the EC must wait to receive the first request from the AP before initiating any SMBus transfers involving the AP. In this way, the AP-side driver is guaranteed to be ready to accept SMBus transfers from the EC.

This same process is repeated when the AP enters deep sleep. Once notified that the AP is entering deep sleep, the EC must hold any further SMBus transfers involving the AP until after the AP wakes up and initiates its first request. Note that the EC is required to send an acknowledgement for the AP's deep sleep notification before stopping SMBus communications.

EC should generally transfer packets in the following priority order (from high to low):

1. Response packets
2. Request packets
3. Event packets

This should avoid deadlock conditions since the Response packet queue is guaranteed to drain if no new Request packets are received. It should also avoid starvation since both Request and Response packets are processed before Event packets. If too many Request/Responses are performed, though, Event packets may be delayed. The EC firmware must handle this condition gracefully. Auxiliary Device Events already degrade gracefully, because new events cannot accumulate until previous events have already been sent to the AP. Keyboard Events should also degrade gracefully as there is already a mechanism in place to indicate EC-internal key press buffer overflow.

The AP may NACK an SMBus operation at any time if it is unable to process the operation at that time. The EC is required to retry the NACK'ed operation after a delay of 10 msec. If the operation still cannot be completed after 10 retry attempts, the EC must inhibit any further SMBus operations until such time as the AP makes a request. Once the AP makes its request, SMBus operations are once again enabled and the EC must resume retrying the NACK'ed operation. This cycle of retrying the operation and inhibiting the bus continues until the operation is completed successfully.

After EC Power On Reset, EC must be ready to accept Tegra request messages over SMBus and respond to them with the following time-out limits:

- After the EC_REQUEST signal is asserted, the EC is expected to read the pending Request Packet from the AP within 20 msec (typical), 500 msec (max). After 600 msec, the AP will conclude that an error has occurred.

- After the EC receives a Request Packet, the EC must send a corresponding Response Packet to the AP within 20 msec (typical), 500 msec (max). After 600 msec, the AP will conclude that an error has occurred.

The EC must be able to handle the case where the AP makes additional requests before the EC has yet sent responses for previous requests. One possible EC implementation would be to serialize request/response pairs by ignoring the EC_REQUEST signal until after sending a response for the current request; only then would the next request be read from the AP. Should the EC elect to process multiple requests concurrently, requests sharing a common Command Type must be queued and processed sequentially in the order received.

Appendix A—List of Commands

Command	Sub-command
01h System Status	00h Get System Status 01h Configure Event Reporting 02h Acknowledge System Status fdh Configure Wake
02h Battery Information	Battery Slot Tag, bits 7:5 0h Battery Slot 0 1h Battery Slot 1 2h Battery Slot 2 3h Battery Slot 3 Others—Reserved Battery Operations, bits 4:0 00h Get Slot Status and Capacity Gauge 01h Get Voltage 02h Get Remaining Time to Empty 03h Get Current 04h Get Average Current 05h Get Averaging Time Interval 06h Get Remaining Capacity 07h Get Last Full Charge Capacity 08h Get Design Capacity 09h Get Critical Capacity 0ah Get Temperature 0bh Get Manufacturer Name 0ch Get Model 0dh Get Type 0eh Set Remaining Capacity Alarm 0fh Get Remaining Capacity Alarm 10h Set Configuration 11h Get Configuration 12h Configure Event Reporting 1dh Configure Wake Others Reserved

Command	Sub-command
03h GPIO Control	Scalar sub-commands 00h Configure Pin 01h Set Pin (scalar) 02h Get Pin (scalar) 03h Configure Event Reporting (scalar) 04h Acknowledge Event Report (scalar) 06h Get Event Report (scalar) 1dh Configure Wake (scalar) Vector sub-commands 21h Set Pin (vector) 22h Get Pin (vector) 23h Configure Event Reporting (vector) 24h Acknowledge Event Report (vector) 26h Get Event Report (vector) 3dh Configure Wake (vector) Others Reserved
04h Sleep Control	00h Global Configure Event Reporting 01h AP Power Down 02h AP Suspend 03h AP Restart Others Reserved
05h Keyboard Control	ffh Reset f4h Enable f5h Disable f1h Get scan code set f0h Set scan code set edh Set LEDs 03h Configure Wake 04h Configure Wake Key Reporting Others Reserved

Command	Sub-command
06h PS/2 Auxiliary Device Control	PS/2 Port Select, bits 7:6 0h PS/2 port 0 1h PS/2 port 1 2h PS/2 port 2 3h PS/2 port 3 PS/2 Port Operation, bits 5:0 01h Send command (send one byte, then receive N bytes) 02h Receive N bytes 03h Auto-receive N bytes 04h Cancel auto-receive 05h Set compression 3dh Configure Wake Others Reserved
07h System Control	00h Reset EC 01h Self Test 02h No-op 10h Get EC Interface Spec Version 11h Get System Capabilities 12h Get System Configuration 14h Get EC Product Name 15h Get EC Firmware Version 20h Initialize Generic Configuration 21h Send Generic Configuration Bytes 22h Finalize Generic Configuration 30h Initialize firmware update 31h Send firmware bytes 32h Finalize firmware update 33h Poll firmware update 40h Get firmware size 41h Read firmware bytes Others Reserved