Advanced Platform Management Link (APML) Specification

Table of Contents

List of Figures

List of Tables

Revision History

Revision 1.02

- Initial document release.

1 Overview

The Advanced Platform Management Link (SBI) is an SMBus v2.0 compatible 2-wire processor slave interface. APML is also referred as the sideband interface (SBI).

APML is used to communicate with the Remote Management Interface (see section 4.3 [SBI Remote Management Interface (SB-RMI)] on page 11) and the Temperature Sensor Interface (see 4.2 [SBI Temperature Sensor Interface (SB-TSI)] on page 11).

1.1 Intended Audience

This document provides the APML behavioral definition and associated design notes. It is intended for platform designers and for programmers involved in the development of management subsystem firmware. It assumes prior experience in personal computer platform design.

1.2 Reference Documents

- *Fr6 (1207) Processor Functional Data Sheet*, #45603.
- *Socket G34 Processor Functional Data Sheet*, #45937.
- *AMD Family 10h Processor Electrical Data Sheet*, #40014.
- *System Management Bus (SMBus) specification*. www.smbus.org.
- *The I²C-Bus Specification*. www.semiconductors.philips.com/products/interface_control/i2c/

1.3 Numbering Conventions

- **Binary numbers**. Binary numbers are indicated by appending a “b” at the end, e.g., 0110b.
- **Decimal numbers**. Unless specified otherwise, all numbers are decimal. Note: this rule does not apply to the register mnemonics; register mnemonics all utilize hexidecimal numbering.
- **Hexidecimal numbers**. Hexidecimal numbers are indicated by appending an “h” to the end, e.g., 45f8h.
- **Underscores in numbers**. Underscores are used to break up numbers to make them more readable. They do not imply any operation. E.g., 0110_1100b.

1.4 Definitions

- **ARA**. Alert response address.
- **EC**. Embedded controller.
- **KBC**. Keyboard controller.
- **lsb**. Least significant bit.
- **LSB**. Least significant byte.
- **Master or SMBus Master**. The device that initiates and terminates all communication and drives the clock, SCL.
- **msb**.Most significant bit.
- **MSB**. Most significant byte.
- **PEC**. Packet error code.
- **POR**. Power on reset.
- **RMI**. Remote management interface.
- **RTS**. Remote temperature sensor, typical examples are *ADM1032*, *LM99*, *MAX6657*, *EMC1002*.
- **SBI**. Sideband interface.
- **Slave or SMBus slave**. The slave cannot initiate SMBus communication and cannot drive the clock but can

drive the data signal SDA and the alert signal ALERT_L.
- **TSI**. Temperature sensor interface.

1.5 Terminology

Table 1 shows terminology found in the register descriptions.

Table 1: Terminology in register descriptions

Terminology	Description
Read or read-only	Capable of being read by software. Read-only implies that the register cannot be written by software.
Write	Capable of being written by software.
Read-write	Capable of being written by software and read by software.
Set-or-cleared-by-hardware	Register bit is set high or cleared low by hardware.
Write-1-to-clear	Software must write a 1 to the bit in order to clear it. Writing a 0 to these bits has no effect.
Write-0-to-clear	Software must write a 0 to the bit in order to clear it. Writing a 1 to these bits has no effect.
Write-1-only	Software can set the bit high by writing a 1 to it. Writes of 0 have no effect. Cleared by hardware.
Reserved	Field is reserved for future use. Software is required to preserve the state read from these bits when writing to the register. Software may not depend on the state of reserved fields nor on the ability of such fields to return the state previously written.
Cold reset	The field state is not affected by a warm reset (even if the field is labled “cold reset: X”); it is placed into the reset state when PWROK is deasserted.

2 SBI Bus Characteristics

The SBI largely follows SMBus v2.0. This section describes the exceptions.

2.1 Physical Layer Characteristics

The SIC and SID pins differ from the SMBus specification with regard to voltage. Systemboard voltage translators are necessary to convert the SIC and SID pin voltage levels to that of the SMBus specification.

SBI supports frequencies of 100 KHz, 400 KHz, and 3.4 MHz over SIC. In order to operate at 3.4 MHz, the I2C-defined high-speed mode command must be issued to the processor from the master.

2.1.1 SMBus Protocol Support

The SBI follows SMBus protocol except:
- The processor does not implement SMBus master functionality.
- Only 7-bit SMBus addresses are supported.
- The SBI implements the Send Byte/Receive Byte, Read Byte/Write Byte, Block Read/Block Write and Block Write-Block Read Process Call SMBus protocols.
  - The Send Byte/Receive Byte SMBus protocol is only supported by SB-TSI.
- Packet error checking (PEC) is not supported by SB-TSI.

- Address Resolution Protocol (ARP) is not implemented.
- Cumulative clock extensions are not enforced.

2.1.2 I2C Support

The processor supports higher I2C-defined speeds as specified in 2.1 [Physical Layer Characteristics] on page 8. The processor supports the I2C master code transmission in order to reach the high-speed bus mode. Multiple SBI commands may be sent within a single high-speed mode session. Ten-bit addressing is not supported.

3 SBI Processor Information

3.1 SBI Processor Pins

Up to six processor pins are used for SBI support: two for data transfer, three for address determination and one for an interrupt output.

The Serial Interface Clock (SIC) and Serial Interface Data (SID) pins function as the SMBus clock and data pins respectively. The SMBus alert pin (ALERT_L) is used to signal interrupts to the SMBus master. Products that include the address select pins (SA[2:0]) use the pins to initialize the SMBus slave address (see Table 11 on page 22).

3.2 Processor States

SBI responds to SMBus traffic except when:
- PWROK is deasserted (and for a brief period after it is deasserted).

Access to internal processor state using SB-RMI is not supported under the following conditions:
- The processor is in the stop-grant state.
- During cold and warm resets.
- During the APIC spin loop.
- When the HDT interface is in PDM (DBRdy pin is asserted).
- [The Core Enable Status Register 0] SB-RMI 04, [The Core Enable Status Register 1] SB-RMI 05, [The APIC Spin Loop Status Register 0] SB-RMI 06, and [The APIC Spin Loop Status Register 1] SB-RMI 07 registers indicate which cores are accessible by APML. Commands sent to non-enabled cores or cores in the APIC spin loop will receive a Core Not Enabled status return code. Broadcast commands will return Core Not Enabled if any core is either not enabled or in the APIC spin loop.

4 SBI Protocols

4.1 SBI Modified Block Write-Block Read Process Call

SBI uses a modified SMBus PEC-optional Block Write-Block Read Process Call protocol. The change from the SMBus protocol is support for an optional intermediate PEC byte and Ack after the Ack for Data Byte M.This PEC byte covers the data starting with the Slave Address through Data Byte M and is controlled by SB-RMI 01[PECEn]. This is the only modification to the standard SMBus PEC-optional Block Write-Block Read Process Call as defined by the *SMBus Specification*. The PEC byte after Data Byte N covers all previous bytes excluding the first PEC byte. Figure 1 shows the transmission protocol. Each byte in the protocol is sent with the most significant bit first (bit[7]). The master may reset the bus by holding the clock low for 25ms as specified by the *SMBus Specification*.

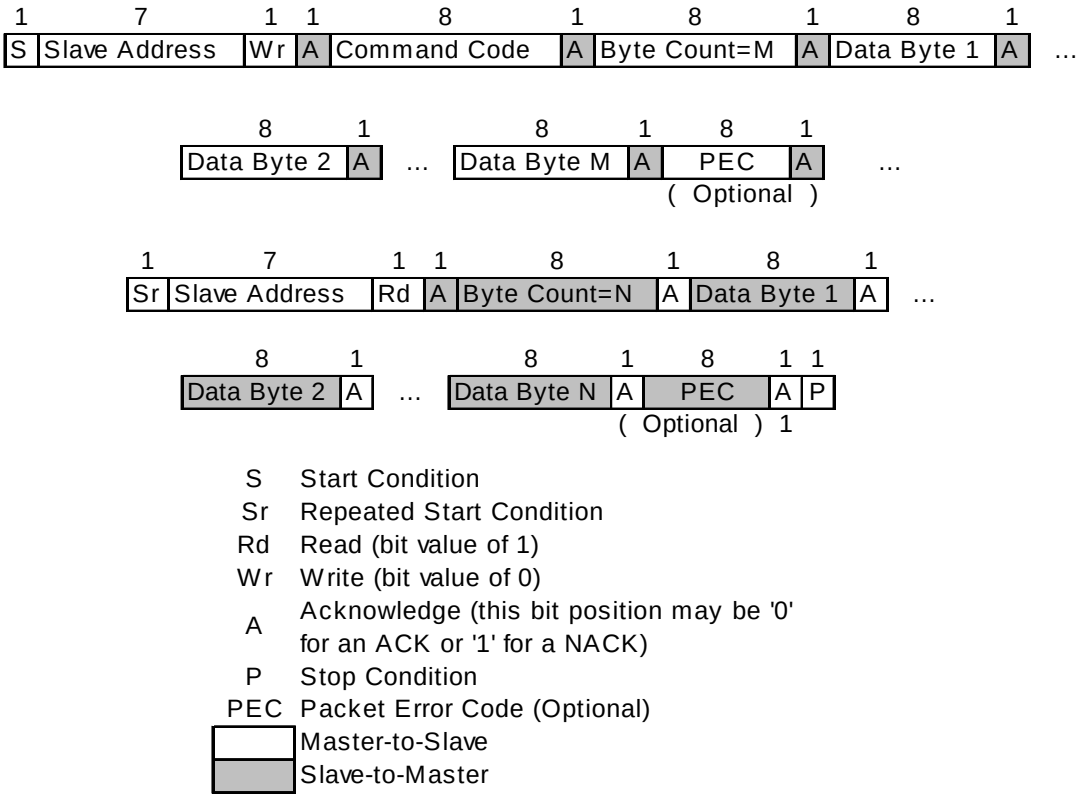


Figure 1: SBI modified block write-block read process call transmission protocol

4.2 SBI Temperature Sensor Interface (SB-TSI)

SB-TSI is used to access the internal temperature sensor and to specify temperature thresholds. SB-TSI functionality is defined in the *SBI Temperature Sensor Interface (SB-TSI) Specification*, #40821.

4.3 SBI Remote Management Interface (SB-RMI)

SB-RMI provides an interface for an external SMBus master that can be used to perform tasks such as monitoring the processor MCA registers, monitoring the current P-state or controlling the maximum P-state allowed. SB-RMI supports signaling Alert_L when a MCE is received by any core, or when software sets SB-RMI 02[SwAlertSts].

4.3.1 SB-RMI Processor State Access

Table 2 describes the functions for accessing processor state. See the *BIOS and Kernel Developer's Guide* of the processor family for additional information about the processor registers.

Table 2: SB-RMI functions

Function	Description	Core Specific[1]
CPUID	Access to CPUID using read CPUID command. General purpose registers are not altered unlike a processor CPUID instruction. See the *BIOS Kernel and Developer's Guide* of the processor family for more information about CPUID functions.	Y
HTC	Register read or write command to register address C001_003Eh to access the Hardware Thermal Control (HTC) Register (F3x64).	N
Current P-state	Register read command to register address C001_0063h to access the P-State Status Register (MSRC001_0063).	Y
Set P-state limit	Register read or write command to register address C001_0072h to access the SBI P-state Limit Register (F3xC4).	N
Current P-state limit	Read command to register address C001_0061h to access the P-State Current Limit Register (MSRC001_0061).	N
MCA Registers	Register read or write command using the MSR address as the register address to access MSR0000_0179, MSR0000_017A, MSR0000_017B, MSR0000_0400 through MSR0000_0417, and MSRC000_04[0A:08].	Y
1. Functions that are not core specific must use SB-RMI 41[CoreNum] as the core in the command.		

4.3.1.1 SB-RMI Read Processor Register and Read CPUID Commands

SB-RMI read processor register and read CPUID commands are performed using the SBI Modified Block Write-Block Read Process Call. If an SMBus timeout occurs before the data is returned, a read data/status command (see Table 5 on page 14) can be issued to read the data from the previous command. The previous command must be complete before a new command can be issued.

Table 3: SB-RMI read processor register command protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address
2	Command	73h	Read CPUID/Read Register Command Format
3	WrDataLen	07h	7 Bytes
4	WrData 1	0Xh	Number of bytes to read from register. Valid values are 1 through 8.
5	WrData 2	86h	Read Register command
6	WrData 3	000X_XXX0b	Bits [4:1] select the core to access. 0h = Core 0, 1h = Core 1, 2h = Core 2 3h = Core 3, 4h = Core 4, 5h = Core 5 6h = Core 6, 7h = Core 7, 8h = Core 8 9h = Core 9, Ah = Core 10, Bh = Core 11 Ch = Core 12, Dh = Core 13, Eh = Core 14 Fh = Core 15
7	WrData 4	XXh	Register Address [7:0] from Table 2

Table 3: SB-RMI read processor register command protocol

Byte	Byte Name	Value	Notes
8	WrData 5	XXh	Register Address [15:8] from Table 2
9	WrData 6	XXh	Register Address [23:16] from Table 2
10	WrData 7	XXh	Register Address [31:24] from Table 2
11	PEC	XXh	Optional PEC byte
12	Slave Address	0111_XXX1b	Read Address
13	RdDataLen	0Xh	Number of bytes returned = WrData 1 + 1.
14	Status	XXh	Status Code
15	RdData 1	XXh	Register Data [7:0]
16	RdData 2	XXh	Register Data [15:8] Optional
17	RdData 3	XXh	Register Data [23:16] Optional
18	RdData 4	XXh	Register Data [31:24] Optional
19	RdData 5	XXh	Register Data [39:32] Optional
20	RdData 6	XXh	Register Data [47:40] Optional
21	RdData 7	XXh	Register Data [55:48] Optional
22	RdData 8	XXh	Register Data [63:56] Optional
23	PEC	XXh	Optional PEC byte

Table 4: SB-RMI read CPUID command protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address
2	Command	73h	Read CPUID/Read Register Command
3	WrDataLen	08h	8 Bytes
4	WrData 1	08h	Number of CPUID bytes to read.
5	WrData 2	91h	Read CPUID Command
6	WrData 3	000X_XXX0b	Bits [4:1] select the core to access. 0h = Core 0 1h = Core 1 2h = Core 2 3h = Core 3 4h = Core 4 5h = Core 5 6h = Core 6 7h = Core 7 8h = Core 8 9h = Core 9 Ah = Core 10 Bh = Core 11 Ch = Core 12 Dh = Core 13 Eh = Core 14 Fh = Core 15
7	WrData 4	XXh	CPUID function [7:0]
8	WrData 5	XXh	CPUID function [15:8]
9	WrData 6	XXh	CPUID function [23:16]
10	WrData 7	XXh	CPUID function [31:24]
11	WrData 8	0000_000Xb	0b = Return ebx:eax 1b = Return edx:ecx
12	PEC	XXh	Optional PEC byte
13	Slave Address	0111_XXX1b	Read Address

Table 4: SB-RMI read CPUID command protocol

Byte	Byte Name	Value	Notes
14	RdDataLen	09h	Number of bytes returned.
15	Status	XXh	Status Code
16	RdData 1	XXh	eax or ecx [7:0]
17	RdData 2	XXh	eax or ecx [15:8]
18	RdData 3	XXh	eax or ecx [23:16]
19	RdData 4	XXh	eax or ecx [31:24]
20	RdData 5	XXh	ebx or edx [7:0]
21	RdData 6	XXh	ebx or edx [15:8]
22	RdData 7	XXh	ebx or edx [23:16]
23	RdData 8	XXh	ebx or edx [31:24]
24	PEC	XXh	Optional PEC byte

Table 5: SB-RMI read data/status command protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address
2	Command	72h	Read Data/Status Command Format
3	WrDataLen	01h	1 Byte of write data
4	WrData 1	0Xh	Number of bytes to read from register. Valid values are 1 through 8.
5	PEC	XXh	Optional PEC byte
6	Slave Address	0111_XXX1b	Read Address
7	RdDataLen	0Xh	Number of bytes returned = WrData 1 + 1.
8	Status	XXh	Status Code
9	RdData 1	XXh	Register Data [7:0] Optional
10	RdData 2	XXh	Register Data [15:8] Optional
11	RdData 3	XXh	Register Data [23:16] Optional
12	RdData 4	XXh	Register Data [31:24] Optional
13	RdData 5	XXh	Register Data [39:32] Optional
14	RdData 6	XXh	Register Data [47:40] Optional
15	RdData 7	XXh	Register Data [55:48] Optional
16	RdData 8	XXh	Register Data [63:56] Optional
17	PEC	XXh	Optional PEC byte

4.3.1.2 SB-RMI Write Processor Register Command

Writing processor registers from SB-RMI uses two SBI Modified Block Write-Block Read Process Call commands. The first command loads the address of the register to be written into the processor. The register address loaded by this command is stored on a per-core basis. The second command writes the data to the processor register using the stored register address. The read data/status command can be used to determine that

the command completed if an SMBus timeout occurs (see Table 5 on page 14). The previous command must be complete before a new command can be issued.

Table 6: SB-RMI load address command protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address
2	Command	71h	Write Register/Load Address Command Format
3	WrDataLen	06h	6 Bytes
4	WrData 1	81h	Load Address Command
5	WrData 2	000X_XXXXb	Bits [4:1] select the core to access. 0h = Core 0 1h = Core 1 2h = Core 2 3h = Core 3 4h = Core 4 5h = Core 5 6h = Core 6 7h = Core 7 8h = Core 8 9h = Core 9 Ah = Core 10 Bh = Core 11 Ch = Core 12 Dh = Core 13 Eh = Core 14 Fh = Core 15 Bit 0 indicates that register write targets all cores. 0b = Bits [4:1] select core. 1b = Load address command targets all cores.
6	WrData 3	XXh	Register Address [7:0] from Table 2
7	WrData 4	XXh	Register Address [15:8] from Table 2
8	WrData 5	XXh	Register Address [23:16] from Table 2
9	WrData 6	XXh	Register Address [31:24] from Table 2
10	PEC	XXh	Optional PEC byte
11	Slave Address	0111_XXX1b	Read Address
12	RdDataLen	01h	Number of bytes returned.
13	Status	XXh	Status Code
14	PEC	XXh	Optional PEC byte

Table 7: SB-RMI write processor register command protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address
2	Command	71h	Write Register/Load Address Command Format
3	WrDataLen	0Xh	Total number of WrData bytes sent by the master. The total number of bytes written to the register (WrDataLen-2) must match the size of the register that is being written or undefined data will be written into the register.
4	WrData 1	87h	Write Register Command

Table 7: SB-RMI write processor register command protocol

Byte	Byte Name	Value	Notes
5	WrData 2	000X_XXXX	Bits [4:1] select the core to access. 0h = Core 0 1h = Core 1 2h = Core 2 3h = Core 3 4h = Core 4 5h = Core 5 6h = Core 6 7h = Core 7 8h = Core 8 9h = Core 9 Ah = Core 10 Bh = Core 11 Ch = Core 12 Dh = Core 13 Eh = Core 14 Fh = Core 15 Bit 0 indicates that register write targets all cores. 0b = Bits [4:1] select core. 1b = Write targets all cores.
6	WrData 3	XXh	Register Data [7:0]
7	WrData 4	XXh	Register Data [15:8] Optional
8	WrData 5	XXh	Register Data [23:16] Optional
9	WrData 6	XXh	Register Data [31:24] Optional
10	WrData 7	XXh	Register Data [39:32] Optional
11	WrData 8	XXh	Register Data [47:40] Optional
12	WrData 9	XXh	Register Data [55:48] Optional
13	WrData 10	XXh	Register Data [63:56] Optional
14	PEC	XXh	Optional PEC byte
7 + WrDataLen	Slave Address	0111_XXX1b	Read Address
8 + WrDataLen	RdDataLen	01h	Number of bytes returned.
9 + WrDataLen	Status	XXh	Status Code
10 + WrDataLen	PEC	XXh	Optional PEC byte

4.3.1.3 SB-RMI Protocol Status Codes

The legal values for the Status byte of the SB-RMI processor state accesses are shown in Table 8.

Table 8: SB-RMI protocol status codes

Status Code Value	Name	Description
00h	Success	Command.
10h	Command Aborted	The processor core targeted by the command could not start the command before an SMBus timeout occurred and was aborted by the processor. This status code will never occur if SB-RMI 01[TimeoutDis] = 1.
11h 12h	Command Timeout	Command did not complete before an SMBus timeout occurred. This status code will never occur if SB-RMI 01[TimeoutDis] = 1.
2Xh	Interface Busy	SB-RMI interface is busy.
40h	Unknown Command Format	The value in the Command Format field is not recognized.
41h	Invalid Read Length	The value in RdDataLen is less than 1 or greater than 32.

Table 8: SB-RMI protocol status codes

Status Code Value	Name	Description
42h	Excessive Data Length	The sum of the RdDataLen and WrDataLen is greater than 32 and RdDataLen is greater than or equal to 1 and less than or equal to 31.
44h	Invalid Core	Invalid core selected.
45h	Unsupported Command	Command not supported by the processor.

4.3.2 SB-RMI Register Access

The SB-RMI registers can be read or written from the SMBus interface using the SMBus defined PEC-optional Read Byte and Write Byte protocols with the SB-RMI register number in the command byte or the PEC-optional Block Read and Block Write protocols with the first SB-RMI register number to be accessed in the command byte. SB-RMI registers 10h-4Fh use either Read/Write Byte or Block Read/Write protocols as specified by SB-RMI 01[BlkRWEn]. The SB-RMI interface supports Block Reads and Block Writes of up to 32 bytes as specified by SB-RMI 03[RdSize]. Bytes are returned in ascending register order starting with the first SB-RMI register in the command byte.

4.3.2.1 SB-RMI Register List

Reads to unimplemented registers return 00h. Writes to unimplemented registers are discarded.

SB-RMI 00 Interface Revision Register

Reset: 02h.

Bits	Description
7:0	**Revision: SB-RMI revision**. Read-only. This field specifies the APML specification revision that the product is compliant to. See also the *BIOS and Kernel Developer's Guide* of the processor family. 02h = 1.0.

SB-RMI 01 Control Register

Reset: 61h.

Bits	Description
7	**PECEn: packet error checking enable**. Read-write. 0=Intermediate PEC is disabled.1=Intermediate PEC is enabled. This only controls the intermediate PEC of the SBI Modified Block Write-Block Read Process Call.
6:5	Reserved.
4	**SwAlertMask: software alert mask**. Read-write. 0=Alert_L signaling is enabled when SB-RMI 02SwAlertSts is set. 1=Alert_L signaling is disabled when SB-RMI 02SwAlertSts is set.
3	**BlkRWEn: block read/write enable**. Read-write. 0=SMBus accesses to SB-RMI[4F:10] use the byte read/write protocol. 1=SMBus accesses to SB-RMI[4F:10] use the block read/write protocol.
2	**TimeoutDis: SB-RMI timeout disable**. Read-write. 1=SMBus defined timeouts are disabled. If the SB-TSI interface is also in use, SMBus timeouts should be enabled or disabled in a consistent manner on both interfaces. The SB-TSI timeout setting is used by SB-RMI until the SMBus interface can determine which interface is targeted by the transaction.

1	**AraDis: SB-RMI ARA disable**. Read-write. 1=Sending of an ARA response is disabled. 0=Sending of an ARA response is enabled.
0	**AlertMask: SB-RMI alert mask**. Read-write; set-by-hardware if AraDis=0 and a successful ARA is sent. 1=Alert_L signaling disabled. 0=Alert_L is asserted if any unmasked event is present in the [The Alert Status Registers] SB-RMI 1[F:0] or if SB-RMI 02[SwAlertSts]=1 and SwAlertMask=0.

SB-RMI 02 Status Register

Reset: 00h.

Bits	Description
7:2	Reserved.
1	**SwAlertSts: SB-RMI software alert status**. Write-one-to-clear from the SMBus interface; Read-write from the processor. 1=Software generated alert event.
0	**AlertSts: SB-RMI alert status**. Read-Only; set-by-hardware; cleared-by-hardware. 1=Alert event present in one of the [The Alert Status Registers] SB-RMI 1[F:0].

SB-RMI 03 Read Size Register

Reset: 01h. This register specifies the number of bytes to return when using the block read protocol to read SB-RMI[4F:10].

Bits	Description
7:6	Reserved.
5:0	**RdSize: read size**. Read-write specifies the number of bytes to return when using the block read protocol. 00h = Reserved. 01h = 1 byte. 02h = 2 bytes. ... 1Fh = 31 bytes. 20h = 32 bytes 3Fh-21h = Reserved.

SB-RMI 04 Core Enable Status Register 0

Reset: 00h. This register specifies the core enable status for cores 0-7.

Bits	Description
7:0	**CoreEnStat: Core Enable Status**. Read-only. Bit vector for cores 0-7. 1=Core is enabled.

Bit	Description	Bit	Description
0	core 0.	4	core 4.
1	core 1.	5	core 5.
2	core 2.	6	core 6.
3	core 3.	7	core 7.

SB-RMI 05 Core Enable Status Register 1

Reset: 00h. This register specifies the core enable status for cores 8-15.

Bits	Description
7:0	**CoreEnStat: Core Enable Status**. Read-only. Bit vector for cores 8-15. 1=Core is enabled. Bit Description Bit Description 0 core 8. 4 core 12. 1 core 9. 5 core 13. 2 core 10. 6 core 14. 3 core 11. 7 core 15.

SB-RMI 06 APIC Spin Loop Status Register 0

Reset: 00h. This register specifies the APIC spin loop status for cores 0-7.

Bits	Description
7:0	**APICStat: APIC spin loop status**. Read-only. Bit vector for cores 0-7. 1=core is in APIC spin loop. Unimplemented cores have the value of 0. Bit Description Bit Description 0 core 0. 4 core 4. 1 core 1. 5 core 5. 2 core 2. 6 core 6. 3 core 3. 7 core 7.

SB-RMI 07 APIC Spin Loop Status Register 1

Reset: 00h. This register specifies the APIC spin loop status for cores 8-15.

Bits	Description
7:0	**APICStat: APIC spin loop status**. Read-only. Bit vector for cores 0-7. 1=core is in APIC spin loop. Unimplemented cores have the value of 0. Bit Description Bit Description 0 core 8. 4 core 12. 1 core 9. 5 core 13. 2 core 10. 6 core 14. 3 core 11. 7 core 15.

SB-RMI 1[F:0] Alert Status Registers

Reset: 00h. SB-RMI 10h is associated with core 0. SB-RMI 11h is associated with core 1. SB-RMI 1Fh is associated with core 15. The number of Alert Status Registers present depends on the number of core that the processor implements.

Bits	Description
7:1	Reserved.
0	**MceStat: MCE status**. Write-1-to-clear; set-by-hardware. 1=MCE occurred on core.

SB-RMI 2[F:0] Alert Mask Registers

Reset: 00h. SB-RMI 20h is associated with core 0. SB-RMI 21h is associated with core 1. SB-RMI 2Fh is associated with core 15.The number of Alert Mask Registers present depends on the number of core that the processor implements.

Bits	Description
7:1	Reserved.
0	**MceAlertMsk: MCE alert mask**. Read-write. 1=Alert signaling disabled for MCE (SB-RMI 1[F:0][MceStat]) from the core.

SB-RMI 3[7:0] Outbound Message Registers

Reset: 00h. These registers are used for sending messages from software running on the processor to the SMBus master.

Bits	Description
7:0	**OutBndMsg: outbound message data**. Read-write from the processor; Read-only from the SMBus interface.

SB-RMI 3[F:8] Inbound Message Registers

Reset: 00h. These registers are used for sending messages from the SMBus master to software running on the processor.

Bits	Description
7:0	**InBndMsg: inbound message data**. Read-write from the SMBus interface; Read-only from the processor.

SB-RMI 40 Software Interrupt Register

Reset: 00h. This register is used by the SMBus master to generate an interrupt to the processor to indicate that a message is available. The local vector table offset for the interrupt is specified by The SBI Control Register F3x1E4 in the *BIOS and Kernel Developer's Guide* of the processor family.

Bits	Description
7:1	Reserved
0	**SwInt: software interrupt**. Read, write-1-only from the SMBus interface; Read, write-1-to-clear from the processor. 1=Processor interrupt generated.

SB-RMI 41 RMI Core Number

Reset: 00h. This register indicates the core number that must be used for non core-specific RMI functions. See Table 2.

Bits	Description

7:4	Reserved
3:0	**CoreNum: core number**. Read-only. Specifies a logical core number. 00h = core 0. 01h = core 1. 02h = core 2. ... 0Fh = core 15.

4.3.2.2 SB-RMI Register Block Read

The following example shows a block read to SB-RMI 1[F:0] using the SMBus Block Read protocol. In the example, SB-RMI 01[BlkRWEn]=1 and SB-RMI 03=08h and core 1 has an MCE event logged.

Table 9: SB-RMI register block read protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address
2	Command	10h	Indicates starting register SB-RMI 10.
3	Slave Address	0111_XXX1b	Read Address
4	Byte Count	08h	Number of bytes returned.
5	Data Byte 1	00h	Value of SB-RMI 10h.
6	Data Byte 2	01h	Value of SB-RMI 11h.
7	Data Byte 3	00h	Value of SB-RMI 12h.
8	Data Byte 4	00h	Value of SB-RMI 13h.
9	Data Byte 5	00h	Value of SB-RMI 14h.
10	Data Byte 6	00h	Value of SB-RMI 15h.
11	Data Byte 7	00h	Value of SB-RMI 16h.
12	Data Byte 8	00h	Value of SB-RMI 17h.
13	PEC	XXh	Optional PEC byte.

4.3.2.3 SB-RMI Register Write Byte

The following example shows a write to SB-RMI 03 using the SMBus Write Byte protocol.

Table 10: SB-RMI register write byte protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address
2	Command	03h	Indicates SB-RMI register 03.
3	Data Byte	04h	Write a value of 04h.
4	PEC	XXh	Optional PEC byte.

5 SBI Physical Interface

5.1 SBI SMBus Address

The SMBus address is really 7 bits. Some vendors and the SMBus specification show the address as 8 bits, left justified with the R/W bit as a write (0) making bit 0. Some vendors use only the 7 bits to describe the address. The SB-TSI address is normally 98h (1001 100W) or 4Ch (100 1100). The address could vary with address select pins.

Table 11: SBI address encodings

F3x1E4[SbiAddr] or Address Select Pins	SB-TSI Address	SB-RMI Address
000b	98h	78h
001b	9Ah	7Ah
010b	9Ch	7Ch
011b	9Eh	7Eh
100b	90h	70h
101b	92h	72h
110b	94h	74h
111b	96h	76h

5.2 SBI Bus Timing

In a mobile or desktop environment, SBI supports 400 KHz fast-mode I^2C operation. In a server environment, SBI supports 3.4 MHz high-speed mode I^2C operation. Refer to the fast-mode and high-speed mode timing parameters in *The I^2C Specification*.

5.3 SBI Bus Electrical Parameters

SBI conforms to most of the I^2C fast-mode electrical parameters. Table 12 lists the electrical parameters of SBI that vary from I^2C or are specific to SBI.

Symbol	**Parameter**	**Unit**	**Minimum**	**Typical**	**Maximum**
$VDDIO_{DDR2}$ [1]	Nominal bus voltage	V	1.7	1.8	1.9
$VDDIO_{DDR3}$ [1]	Nominal bus voltage	V	1.425	1.5	1.575
VIH_DC (SCL, SDA)	Input high voltage	V	0.7(VDDIO)	-	VDDIO + 0.3
VIL_DC (SCL, SDA)	Input low voltage	V	-0.3	-	0.3(VDDIO)
Input Hysteresis of Schmitt Trigger Inputs	Vhyst	V	0.1(VDDIO)	-	-
VOL_DC (SDA, ALERT_L)	Low level output voltage at 3 mA sink current	V	0.2(VDDIO)	-	-

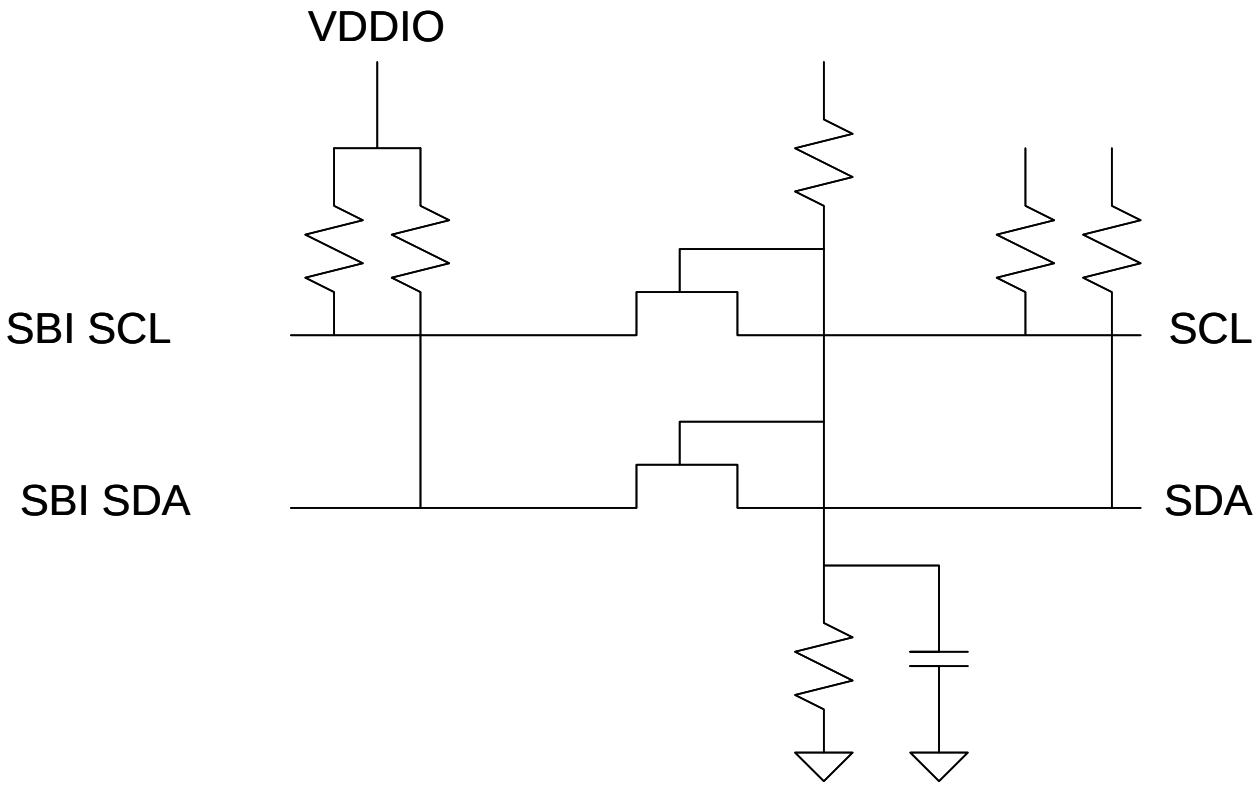
Table 12: SBI specific SMBus electrical parameters

Note:
1. Interface voltage is bounded by the AMD processor VDDIO voltage rail.
   For DDR2 VDDIO is nominally 1.8V.
   For DDR3 VDDIO is nominally 1.5V.

Table 12: SBI specific SMBus electrical parameters

5.4 Pass-FET Option

There is a possibility that a device with a standard SMBus interface will not be able to directly interface to SBI. Therefore, pass FETs must be used to create two SMBus segments, see Figure 2



Notes:
- SCL & SDA pull-up resistors are the normal pull-up resistors for an SMBus segment, and are not part of the translation circuit. They are shown for completeness.
- The gates of the FETs are tied to a voltage approximately Vgs above the lower rail voltage. A resistive divider is shown, but a convenient power rail will do nicely.
- care must be taken to install the FETs so that any body diode does not conduct.
- The key requirement is that the high side drive low enough to register as a low on the low side. (High side Vol < Vil on low side)

Figure 2: Pass FET implementation

6 Register List

The following is a list of all storage elements, context, and registers provided in this document. Page numbers, register mnemonics, and register names are provided.