

**Processor Programming
Reference (PPR)
for AMD Family 1Ah
Model 11h, Revision B0
Processors
Volume 4 of 7**

Legal Notices

© 2022-2024 Advanced Micro Devices, Inc. All rights reserved.

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of these materials, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of these materials, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by these materials. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks:

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

AGESA is a trademark of Advanced Micro Devices, Inc.

AMD Virtualization is a trademark of Advanced Micro Devices, Inc.

AMD-V is a trademark of Advanced Micro Devices, Inc.

Adobe is a registered trademark of Adobe.

Arm is a registered trademark of Arm Limited.

CXL is a trademark of Compute Express Link Consortium, Inc.

EPYC is a trademark of Advanced Micro Devices, Inc.

Infinity Fabric is a trademark of Advanced Micro Devices, Inc.

Linux is a registered trademark of Linus Torvalds.

MIPI I3C is a registered trademark of MIPI Alliance.

Microsoft is a registered trademark of Microsoft Corporation.

PCI Express is a registered trademark of PCI-SIG Corporation.

PCIe is a registered trademark of PCI-SIG Corporation.

SoundWire is a registered trademark of MIPI Alliance, Inc.

Windows is a registered trademark of Microsoft Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Reverse engineering or disassembly is prohibited.

USE OF THIS PRODUCT IN ANY MANNER THAT COMPLIES WITH THE MPEG ACTUAL OR DE FACTO VIDEO AND/OR AUDIO STANDARDS IS EXPRESSLY PROHIBITED WITHOUT ALL NECESSARY LICENSES UNDER APPLICABLE PATENTS. SUCH LICENSES MAY BE ACQUIRED FROM VARIOUS THIRD PARTIES INCLUDING, BUT NOT LIMITED TO, IN THE MPEG PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, L.L.C., 6312 S. FIDDLERS GREEN CIRCLE, SUITE 400E, GREENWOOD VILLAGE, COLORADO 80111.

List of Chapters

Volume 1:

- 1** [Overview](#)
- 2** [Core Complex \(CCX\)](#)

Volume 2:

- 3** [Reliability, Availability, and Serviceability \(RAS\) Features](#)

Volume 3:

- 4** [System Management Unit \(SMU\)](#)

Volume 4:

- 5** **Advanced Platform Management Link (APML)**
- 6** **SB Temperature Sensor Interface (SB-TSI)**
- 7** **Host System Management Port (HSMP)**
- 8** **Data Fabric (DF)**
- 9** **Unified Memory Controller (UMC)**

Volume 5:

- 10** [Northbridge IO \(NBIO\)](#)

Volume 6:

- 11** [Fusion Controller Hub \(FCH\)](#)

Volume 7:

- 12** [Reserved](#)

[List of Namespaces](#)

[List of Definitions](#)

Table of Contents

5	Advanced Platform Management Link (APML)
5.1	Overview
5.1.1	Definitions
5.2	SBI Bus Characteristics
5.2.1	I3C Protocol Support
5.3	SBI Processor Information
5.3.1	SBI Processor Pins
5.3.2	Processor States
5.4	SBI Remote Management Interface (SB-RMI)
5.4.1	SB-RMI Transmission Protocol
5.4.1.1	Register Address Offset Specification Change
5.4.2	SB-RMI Functions
5.4.2.1	SB-RMI Processor State Access
5.4.2.1.1	SB-RMI Read Processor Register Command Protocol
5.4.2.1.2	SB-RMI Read CPUID Command Protocol
5.4.2.1.3	SB-RMI Write Processor Register Command
5.4.2.1.4	SB-RMI Protocol Status Codes
5.4.2.2	SB-RMI Mailbox Service
5.4.2.2.1	APML Performance Modes
5.4.2.2.2	SB-RMI Mailbox Sequence
5.4.2.2.3	Reading RAPL Energy Counters
5.4.2.3	SB-RMI Boot code status
5.4.2.4	SB-RMI Register Access
5.4.2.4.1	SB-RMI Register Block Access
5.4.2.4.2	SB-RMI Register Byte Access
5.4.2.5	SB-RMI Alert
5.4.3	SBI Error Detection and Recovery
5.4.3.1	Error Detection
5.4.3.1.1	ACK/NAK Mechanism
5.4.3.1.2	Ninth Bit of SDR Controller Written Data as Parity
5.4.3.1.3	Bus Timeouts
5.4.3.2	Error Recovery
5.4.3.2.1	SB-RMI Soft Reset Recovery
5.5	SBI Physical Interface
5.5.1	SBI Static Address
5.5.2	SBI Bus Timing
5.6	SB-RMI Registers
6	SB Temperature Sensor Interface (SB-TSI)
6.1	Overview
6.1.1	Definitions
6.1.2	Tctl
6.2	SB-TSI Protocol
6.2.1	I3C Private Write and Read Protocol
6.2.2	SB-TSI Read/Write Byte Protocol
6.2.3	Alert Behavior
6.2.4	Atomic Read Mechanism
6.2.5	SB-TSI Temperature and Threshold Encodings
6.2.6	SB-TSI Temperature Offset Encoding
6.2.7	SB-TSI Error Detection and Recovery
6.3	SB-TSI Physical Interface

- 6.3.1 SB-TSI I3C Static Address
- 6.3.2 SB-TSI Bus Timing
- 6.3.3 SB-TSI Bus Electrical Parameters
- 6.3.4 Pass-FET Option
- 6.4 SB-TSI Registers
- 7 Host System Management Port (HSMP)**
 - 7.1 Overview
 - 7.2 SMN Mailbox Registers
 - 7.2.1 Message ID
 - 7.2.2 Message Arguments
 - 7.2.3 Message Response
 - 7.3 Mailbox Protocol
 - 7.3.1 Response Codes
 - 7.4 HSMP Functions
 - 7.4.1 HSMP Boost Limit
 - 7.4.2 ApicId Mapping
 - 7.4.3 HSMP Performance Modes
 - 7.5 Registers
 - 7.5.1 IOHC Registers
- 8 Data Fabric (DF)**
 - 8.1 Fabric Performance Monitor Counter (PMC) Events
 - 8.1.1 Background
 - 8.1.2 Interfaces and Instance IDs
 - 8.1.3 Event Definitions
 - 8.1.3.1 DATA_BW (Data Bandwidth)
 - 8.1.3.2 Unit of Measurement
 - 8.1.3.3 Usage Notes for SrcDst field
 - 8.1.4 Algorithm for Data Bandwidth measurement
 - 8.1.5 Examples for EventSelect and UnitMask
 - 8.1.5.1 Examples 1:
 - 8.1.5.2 Examples 2:
 - 8.1.5.3 Examples 3:
 - 8.2 DF Configuration Register Indirect Access
 - 8.2.1 Indirect Access Registers
 - 8.2.2 DF Indirect Access Procedure
 - 8.3 DRAM Address Maps
 - 8.4 Registers
 - 8.4.1 Function 4 Registers
 - 8.4.2 Function 7 Registers
 - 8.5 Data Fabric Instance and Fabric IDs
- 9 Unified Memory Controller (UMC)**
 - 9.1 DDR5 Overview
 - 9.1.1 UMC and DDR Phy Mapping
 - 9.1.1.1 UMC and DDR Phy Logical Mapping
 - 9.2 UMC Performance Monitors
 - 9.2.1 UMC Performance Monitor Events
 - 9.3 UMC Registers
 - 9.3.1 Controller Registers

List of Figures

Figure 28:	I3C Protocol for Private Write and Read
Figure 29:	SBI Transmission Protocol for Single Write and Read
Figure 30:	SBI Transmission Protocol for Block Write and Read
Figure 31:	SB-RMI Soft Reset Recovery
Figure 32:	RTS Thermal Management Example
Figure 33:	SB-TSI Thermal Management Example
Figure 34:	I3C Transmission Protocol for Private Write and Read
Figure 35:	SB-TSI Transmission Protocol for Single Write and Read
Figure 36:	Alert Assertion Diagram
Figure 37:	SB-TSI Soft Reset Recovery
Figure 38:	Pass FET Implementation
Figure 39:	Infinity Fabric™ Interconnect

List of Tables

Table 126:	APML Definitions
Table 127:	SB-RMI Functions
Table 128:	SB-RMI Read Processor Register (Write Phase)
Table 129:	SB-RMI Read Processor Register (Reload Read Address Phase)
Table 130:	SB-RMI Read Processor Register (Read Data/Status Phase)
Table 131:	SB-RMI Read CPUID (Write Phase)
Table 132:	SB-RMI Read CPUID (Reload Read Address Phase)
Table 133:	SB-RMI Read CPUID (Read Status/Data Phase)
Table 134:	SB-RMI Status Codes
Table 135:	SB-RMI Soft Mailbox Message
Table 136:	APML Frequency Limit Source
Table 137:	APML IO Bandwidth Encoding
Table 138:	APML XGMI Bandwidth Encoding
Table 139:	APML Link ID Encoding
Table 140:	APML LCLK Frequency Per DPM Level
Table 141:	APML DIMM ADDRESS Encodings
Table 142:	Runtime Error Info Encoding
Table 143:	Set OOB Config DataIn
Table 144:	Get OOB Config DataOut
Table 145:	SB-RMI Soft Mailbox Error Code
Table 146:	SB-RMI Register Block Write Protocol
Table 147:	SB-RMI Register Block Read Protocol
Table 148:	SB-RMI Register Write Byte Protocol
Table 149:	SB-RMI Register Read Byte Protocol
Table 150:	SB-TSI Definitions
Table 151:	SB-TSI CPU Temperature and Threshold Encoding Examples
Table 152:	SB-TSI Temperature Offset Encoding Examples
Table 153:	SB-TSI Static Address Encodings
Table 154:	Definitions
Table 155:	SMN Address for HSMP Mailbox Registers
Table 156:	HSMP Response Codes
Table 157:	HSMP Functions
Table 158:	HSMP Supported Functions Per Interface Version
Table 159:	HSMP LCLK Frequency Per DPM Level
Table 160:	HSMP Frequency Limit Source
Table 161:	HSMP IO Bandwidth Encoding
Table 162:	HSMP XGMI Bandwidth Encoding
Table 163:	HSMP Link ID Encoding
Table 164:	HSMP DIMM ADDRESS Encodings
Table 165:	BXXD00F0x0C4 (IOHC::NB_SMN_INDEX_3)
Table 166:	BXXD00F0x0C8 (IOHC::NB_SMN_DATA_3)
Table 167:	IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)
Table 168:	Interface Descriptions
Table 169:	Instance IDs for the Interface Blocks
Table 170:	DATA_BW PMC Event Unitmask for CCM, IOM and CS Interfaces
Table 171:	DATA_BW PMC Event Unitmask for LINK Controller Interface
Table 172:	DATA_BW PMC Data Beat Size per Interface and TxnType
Table 173:	DF DRAM Address Map Ranges
Table 174:	InstanceID and ComponentID assignment
Table 175:	Instance to Package Channel Mapping for SP5, 12 DDR5 Channels

5 Advanced Platform Management Link (APML)

5.1 Overview

The Advanced Platform Management Link I3C ([APML-I3C](#)) is an I3C based 2-wire processor target interface. APML-I3C is also referred as side band interface (SBI). At the application layer, APML has an expanded set of mailbox commands compared to the previous generation. At the data link layer, the protocol has been upgraded from [SMBus](#) 2.0 to I3C. As a fall back option the I3C target can operate in I2C mode.

APML-I3C is used to communicate with the Remote Management Interface (see SBI Remote Management Interface (SB-RMI) and SBI Temperature Sensor Interface (SB-TSI)). For related specifications, see 1.2 [[Reference Documents](#)].

5.1.1 Definitions

Table 126: APML Definitions

Term	Description
ARA	Alert response address.
ARP	Address Resolution Protocol
CCC	Common Command Code for I3C
Controller or I3C Controller	The device that initiates and terminates all communication and drives the clock, SCL.
EC	Embedded Controller.
ENTDAA	Enter Dynamic Address Assignment CCC for I3C
KBC	Keyboard Controller.
MDC	Mega Data Center
OOB	Out of Band interface, typically referring to APML as opposed to In Band interface which refers to HSMP
ODTS	On die temperature sensing. UMC could be configured to manage DIMM thermals via command throttling and appropriate refresh rate based on temperature range reported by inband MR4 reading from DRAM devices.
PEC	Packet Error Checking or Packet Error Code.
POR	Power on reset.
RTS	Remote temperature sensor, typical examples are ADM1032, LM99, MAX6657, EMC1002.
SBI	Sideband interface.
SB-RMI	Remote Management interface.
SDR	Single Data Rate
SDU	Scan Dump Utility
SETDASA	Set Dynamic Address from Static Address CCC for I3C
Target or I3C target	The target cannot initiate I3C communication and cannot drive the clock but can drive the data signal SDA and the alert signal ALERT_L.
TSI	Temperature sensor interface.
TSOD	Temperature sensor mounted on DIMM PCB.

5.2 SBI Bus Characteristics

The SBI largely follows MIPI I3C[®] Basic Specification 1.0; see [docI3C](#).

5.2.1 I3C Protocol Support

The SBI follows I3C protocol mainly for:

- The processor does not implement I3C controller functionality, but acts as target only.
- Max of 12.5Mhz SDR mode is supported.
- The SBI mainly uses the I3C private Write and Read transfers to support Single or Block Write and Read.
- Target can only drive SDA, but not SCL to controller.
- Dynamic address assigned based on Static address can be supported for I3C target.
- PEC is not supported in I2C or I3C modes by target.

5.3 SBI Processor Information

5.3.1 SBI Processor Pins

Up to five processor pins are used for SBI support: two for data transfer, two for address determination and one for an interrupt output. Of the two address pins, one bit is socket_id used to determine the static address for each socket. These pins do not have changeable pinstrap. The Serial Interface Clock (SIC) and Serial Interface Data (SID) pins function as the I3C clock and data pins respectively. The I3C alert output pin (ALERT_L) is used to signal interrupts to the I3C controller.

5.3.2 Processor States

SBI responds to I3C traffic except when PWROK is de-asserted (and for a brief period after it is de-asserted).

Access to internal processor state using SB-RMI is not supported under the following conditions:

- During cold and warm resets.
- During the APIC spin loop.

In order to allow [MP1](#) firmware to perform initialization of certain SB-RMI and SB-TSI registers, BMC should hold off conducting any transaction to either of these targets for at least 2 sec after cold or warm reset of the CPU.

5.4 SBI Remote Management Interface (SB-RMI)

SB-RMI interface uses the SB-RMI protocol to provide functions such as access to [CUID](#) registers, processor [MCA](#) registers and [APML](#) mailbox commands which are described in details later. The SB-RMI protocol described below matches the version number encoded in the SBRMI::Revision register.

5.4.1 SB-RMI Transmission Protocol

SBI uses a standard I3C protocol to provide Private Write and Private Read Transfers. Figure below shows the I3C spec defined Private Write and Private Read transaction protocol. Each byte in the protocol is sent with the most significant bit first (bit[7]). The ninth data bit of each SDR Data Word written by the I3C controller (also referred to as the T-Bit) is a Parity bit, calculated using odd parity. The ninth (T) Data bit of each SDR Data Word returned for Private Read indicates if it is the End-of-Data. The controller or the target may drive the ninth bit to 0 (SDA Low) to indicate the end of Message.

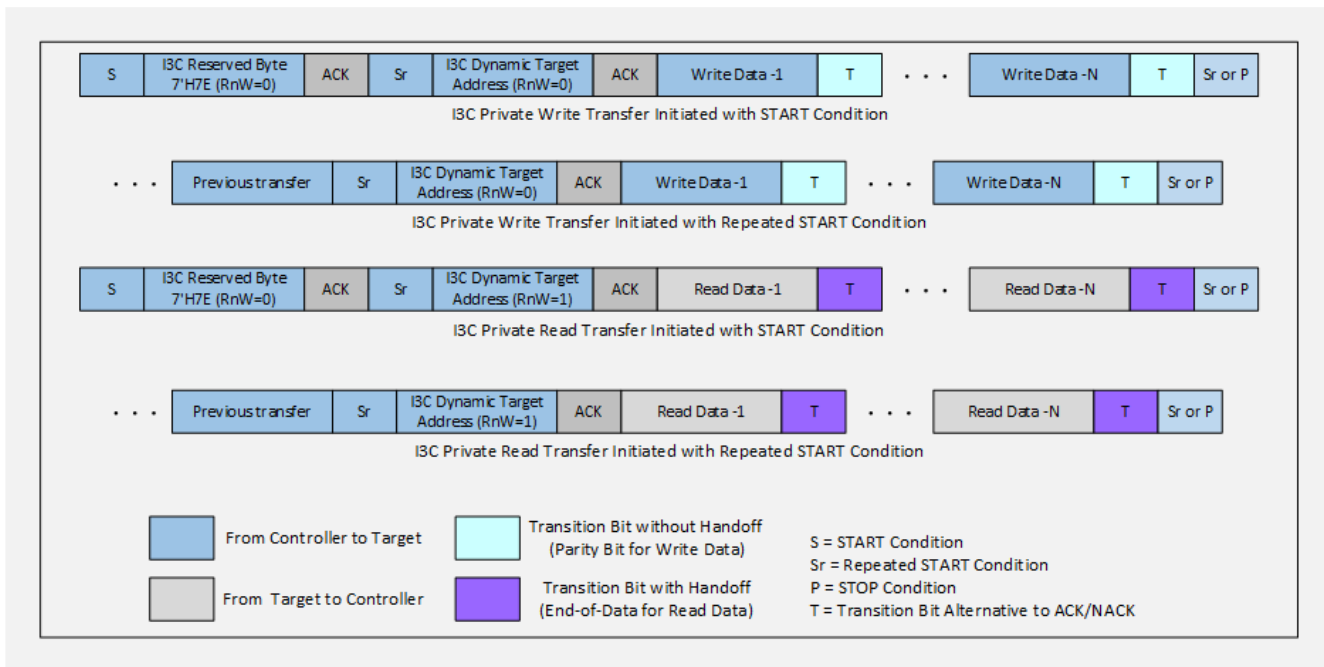


Figure 28: I3C Protocol for Private Write and Read

SB-RMI Write and Read transactions are conducted using I3C defined Private Write and Read protocol or standard I2C Write and Read. The "Command Code" provides the register offset to which the write or read is targeted. Command codes 71h, 72h and 73h are special because they are used for accessing [CUID](#) and [MCA](#) registers. All other values represent SBRMI registers. In SB-RMI revision 2.1 the command code is two bytes long.

Figure below shows the transaction protocol for Single Write and Read. Single Write consists of the "Command Code" followed by one byte of "Write Data". Single Read consists of two parts, a Private Write for providing the "Command Code", followed by a Private Read transaction to read one byte of data with Repeated START or START condition.

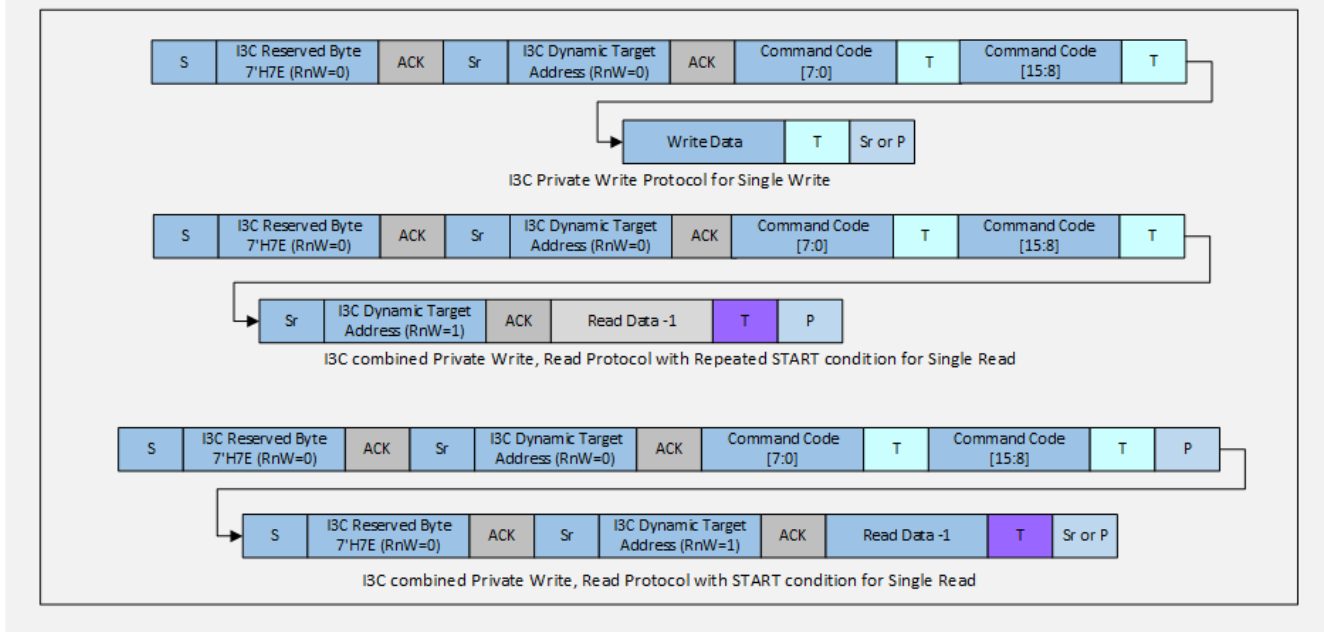


Figure 29: SBI Transmission Protocol for Single Write and Read

Figure below shows the transaction protocol for Block Write and Read. For Block Write, the "Command Code" is followed by multiple bytes of "Write Data". The interpretation of data bytes is based on the value of "Command Code". Block Read consists of two parts, a Private Write for providing the "Command Code", followed by multiple bytes of read data stream with Repeated START or START condition.

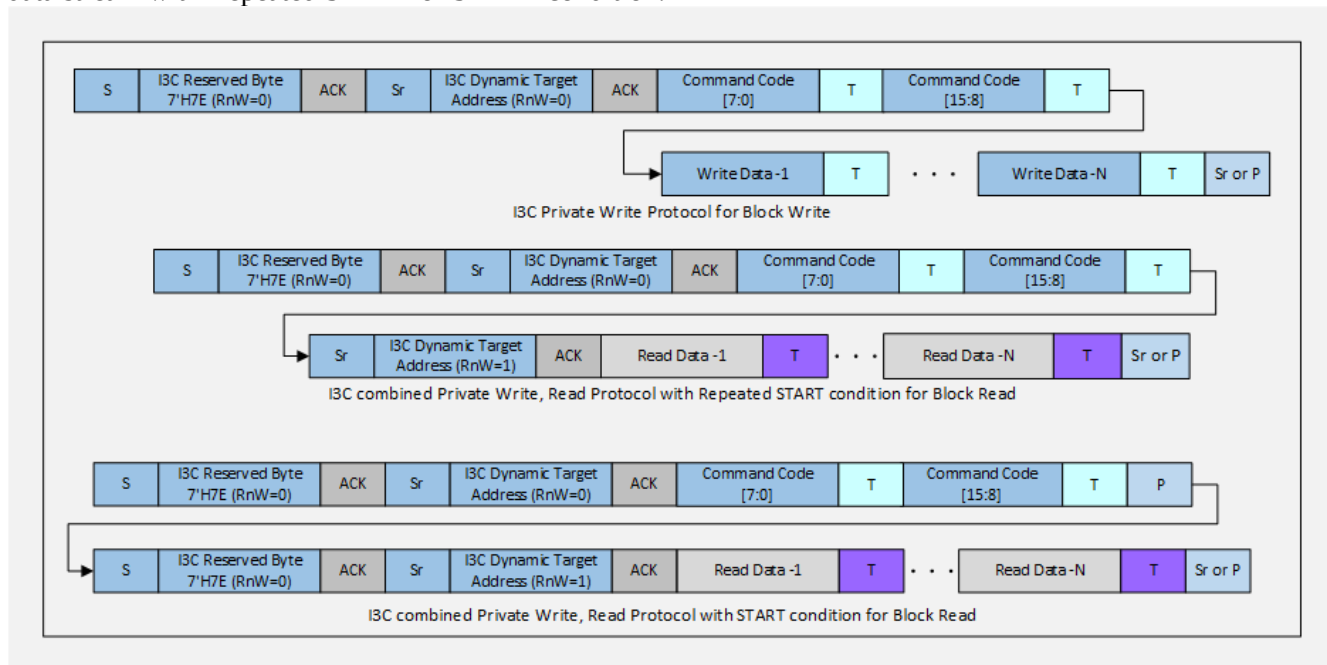


Figure 30: SBI Transmission Protocol for Block Write and Read

5.4.1.1 Register Address Offset Specification Change

The number of SBRMI registers in Family 1Ah Models 10-1Fh processors has increased above 256 to support larger number of thread count. Consequently, the specification of the register offset (i.e. command code) requires a 2-byte value instead of a 1-byte value that was prevalent for earlier processor families. The [APML](#) controller therefore needs to identify which protocol to use at the very beginning before any SBRMI transaction can be conducted. Note that there is no change in the SBTSI protocol, where the command code remains 1-byte long.

There are two methods available for identifying the SBRMI addressing mode.

Method 1:

1. Using the SB-TSI target, read the SBTSI::Config[SBRMIAddrMode] register bit.
2. If the SBRMIAddrMode bit is 0 then a 1-byte addressing protocol is required, else it is a 2-byte addressing protocol.

Method 2:

1. Using the SB-RMI target, read the SBRMI::Revision register using 2 byte length for command code.
2. If the command succeeds and the read value is 21h, the APML target is a Family 1Ah processor that uses 2-bytes for command code.
3. If the command fails (there will be a NACK when the Dynamic Target address with RnW=1 is transmitted), then retry reading SBRMI::Revision register using 1-byte for command code, which should succeed with a read value of 20h. It implies that the APML target is a Family 19h processor and the APML controller should use 1-byte command code.

Method 1 is the simpler of the two methods and recommended for use.

5.4.2 SB-RMI Functions

SB-RMI provides an interface for an external I3C controller that can be used to perform tasks such as monitoring the processor [MCA](#) registers, reading processor [CUID](#) registers and power management or [RAS](#) actions using mailbox interface. SB-RMI supports signaling ALERT_L when a MCE is received by any thread or when software sets SBRMI::Status[SwAlertSts]. Another situation to assert ALERT_L is when hardware sets SBRMI::Status[HwAlertSts] upon completion of 71h/72h/73h commands which is only valid for I3C interface. Each package has an independent I3C target device. See 5.5.1 [SBI Static Address].

Each package is required to contain the same number of logical threads. The I3C target port attached to each package may access only the logical threads within the package. Core::X86::CpuId::SizeId identifies the number of logical threads available in a package.

5.4.2.1 SB-RMI Processor State Access

The SB-RMI Functions table describes the functions for accessing processor state. See the [Processor](#) Programming Reference of the processor family for additional information about the processor registers. [MSR](#) not listed in below table is not accessible, will get "Unsupported Command" status.

Table 127: SB-RMI Functions

Function	Description	Thread Specific
CUID	Access to CUID registers. General purpose registers are not altered unlike a processor CUID instruction. Use Read CUID Command Protocol described in 5.4.2.1.2 [SB-RMI Read CUID Command Protocol]. Access is Read-only.	Y
MCA Registers	Register read command using the register address to access Core::X86::Msr::MCG_CAP determines the number of MCA banks. See 3.1.2.2.1 [Legacy MCA Registers] and 3.1.2.2.3 [MCAX Registers] for the MCA registers within this range. Use Read Processor Register Command Protocol described in 5.4.2.1.1 [SB-RMI Read Processor Register Command Protocol]. Access is Read-only.	Y
DRAM Throttle	Register read or write command access through legacy MSR C001_0079 SMN address to the DRAM Controller ThrottleCtrl register has been discontinued. DRAM throttling is available via "Write DRAM Throttle" command described in 5.4.2.2 [SB-RMI Mailbox Service] section. DRAM throttling can also occur due to Thermal Management Event based on UMC periodic polling of DDR5 MR4.	N
Mailbox Service	Soft mailbox service request to firmware for power management purposes. Past implementations allowed for mailbox operations to X86 software. No usage models for communication with x86 software exists and x86 software messaging is not supported. Access is Read-write. See 5.4.2.2 [SB-RMI Mailbox Service] for details.	N
Boot Status	Boot Status is placed in outbound message register SBRMI::MP0OutBndMsg. Access is Read-only.	N

5.4.2.1.1 SB-RMI Read Processor Register Command Protocol

SB-RMI read processor register command is performed by using the SBI Block Write to load register address first. After

the command is performed completely by the processor, hardware will set SBRMI::Status[HwAlertSts] to trigger ALERT_L pin asserted. After that, a read data/status can be issued from the controller side to check if the command is successfully completed or not as well as reading data by using a Block Read. The previous command must be complete before a new command can be issued.

Table 128: SB-RMI Read Processor Register (Write Phase)

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command	73h	Command code [7:0].
3	Command	00h	Command code [15:8]
4	WrDataLen	08h	8 Bytes.
5	WrData1	0Xh	Number of bytes to read from register (NumRdB). Valid values are 1 through 8.
6	WrData2	86h	Read Register command.
7	WrData3	XXXX_XXXXb	Bit [0] is reserved. Bits [7:1] represent thread[6:0].
8	WrData4	XXXX_XXXXb	Bits [7:0] represent thread[14:7]
9	WrData5	XXh	Register Address [7:0] from the SB-RMI Functions table.
10	WrData6	XXh	Register Address [15:8] from the SB-RMI Functions table.
11	WrData7	XXh	Register Address [23:16] from the SB-RMI Functions table.
12	WrData8	XXh	Register Address [31:24] from the SB-RMI Functions table.

After the above command is completed, hardware will set SBRMI::Status[HwAlertSts] to trigger ALERT_L pin asserted. Before ALERT_L pin is asserted, controller side can do other accesses for write or read. After assertion, a Block Read for data/status can be issued to read the data/status for the previous command. Please note, a new Private Write to reload the "Command" first must be issued first before the Block Read command.

Table 129: SB-RMI Read Processor Register (Reload Read Address Phase)

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command	73h	Command code [7:0].
3	Command	00h	Command code [15:8].

Table 130: SB-RMI Read Processor Register (Read Data/Status Phase)

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX1b	Target Address (Read).
2	RdDataLen	0Xh	Number of bytes returned = NumRdB+1.
3	Status	XXh	Status Code.
4	RdData1	XXh	Register Data [7:0].
5	RdData2	XXh	Register Data [15:8]. Optional.
6	RdData3	XXh	Register Data [23:16]. Optional.
7	RdData4	XXh	Register Data [31:24]. Optional.
8	RdData5	XXh	Register Data [39:32]. Optional.
9	RdData6	XXh	Register Data [47:40]. Optional.
10	RdData7	XXh	Register Data [55:48]. Optional.
11	RdData8	XXh	Register Data [63:56]. Optional.

5.4.2.1.2 SB-RMI Read CPUID Command Protocol

SB-RMI read [CPUID](#) command is performed by using the SBI Block Write based on Private Write Process Call to load CPUID request info first. After the command is completed by the processor, hardware will set SBRMI::Status[HwAlertSts] to trigger ALERT_L pin asserted. After that, a read data/status can be issued from the controller side to check if the command is successfully completed or not as well as reading data by using a Block Read based on Private Read Process Call command. The previous command must be complete before a new command can be issued.

Table 131: SB-RMI Read CPUID (Write Phase)

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command	73h	Command code [7:0].
3	Command	00h	Command code [15:8].
4	WrDataLen	09h	9 Bytes.
5	WrData1	08h	Number of CPUID bytes to read.
6	WrData2	91h	Read CPUID command.
7	WrData3	XXXX_XXXXb	Bit [0] is reserved. Bits [7:1] represent thread[6:0]
8	WrData4	XXXX_XXXXb	Bits [7:0] represent thread[14:7]
9	WrData5	XXh	CPUID Function [7:0].
10	WrData6	XXh	CPUID Function [15:8].
11	WrData7	XXh	CPUID Function [23:16].
12	WrData8	XXh	CPUID Function [31:24].
13	WrData9	ECX[3:0]_000Xb	ECX[3:0] is the initial ECX value for extended CPUID operations. Must be 0h for non-extended operations. X: 0b=Return ebx:eax; 1b=Return edx:ecx.

After the above command is completed, hardware will set SBRMI::Status[HwAlertSts] to trigger ALERT_L pin asserted. Before ALERT_L pin is asserted, controller side can do other accesses for write or read. After assertion, a Block Read for data/status can be issued to read the data/status for the previous command. Please note, a new Private Write to reload the "Command" first must be issued first before the Block Read command.

Table 132: SB-RMI Read CPUID (Reload Read Address Phase)

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command	73h	Command code [7:0].
3	Command	00h	Command code [15:8].

Table 133: SB-RMI Read CPUID (Read Status/Data Phase)

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX1b	Target Address (Read).
2	RdDataLen	09h	Number of CPUID bytes+1
3	Status	XXh	Status Code.
4	RdData1	XXh	eax or ecx [7:0].
5	RdData2	XXh	eax or ecx [15:8].
6	RdData3	XXh	eax or ecx [23:16].
7	RdData4	XXh	eax or ecx [31:24].
8	RdData5	XXh	ebx or edx [7:0].

9	RdData6	XXh	ebx or edx [15:8].
10	RdData7	XXh	ebx or edx [23:16].
11	RdData8	XXh	ebx or edx [31:24].

5.4.2.1.3 SB-RMI Write Processor Register Command

In prior processors, Write Register/Load Address command was only used for DRAM throttle register in UMC using legacy [MSR](#) address C001_0079. This feature is now discontinued. So SB-RMI Write [Processor](#) register command is no longer necessary.

5.4.2.1.4 SB-RMI Protocol Status Codes

The legal values for the Status byte of the SB-RMI processor state accesses are shown in the following table. For each of the status code which should be set after a command is processed, hardware will set SBRMI::Status[HwAlertSts] to trigger ALERT_L pin asserted first. After that, a read data/status can be issued to check the status code.

Table 134: SB-RMI Status Codes

Status Code	Name	Description
00h	Success	Command.
11h	Command Timeout	Command did not complete before a configured timeout event occurred, which indicates MP hasn't cleared the pending interrupt before timeout event occurs. This status code will never occur if (SBRMI_x01[TimeoutDis]==1). MP has not sent the request to CPU/NB.
22h	Warm reset	A warm reset occurred during the transaction.
40h	Unknown Command Format	The value in Command Format field is not recognized.
41h	Invalid Read Length	The value in RdDataLen is less than 1 or greater than 32.
44h	Invalid thread	Invalid thread selected, the data length 0x09 and dummy data will be returned on invalid thread.
45h	Unsupported Command	Command not supported by the processor.
81h	Command Aborted	The processor core targeted by the command could not start the command and was aborted by the processor.

5.4.2.2 SB-RMI Mailbox Service

SB-RMI supports soft mailbox service request to [MP1](#) (power management firmware) through SBRMI inbound/outbound message registers. The message type is defined in the following table.

Table 135: SB-RMI Soft Mailbox Message

Command	Message	Description	Command Data In	Command Data Out
01h	Read Package Power Consumption	Read the average package power consumption. This is estimated per part power capped to cTDP. This data	None	PkgPwr (mWatts), 32-bit integer.

		may be useful to assess relative variations in power under different operating conditions and workloads. This data should not be used for system power budgeting.		
02h	Write Package Power Limit	Set maximum power consumption for SOC package. Note: there is a limit on the minimum power that the processor can operate at; no further socket power reduction occurs if the socket power limit is set below that minimum.	PkgPwrLimit (mWatts), 32-bit integer.	None
03h	Read Package Power Limit	Read the maximum power consumption for SOC package	None	PkgPwrLimit (mWatts), 32-bit integer
04h	Read Max Package Power Limit	Read the maximum package power limit.	None	ReadMaxPackagePowerLimit (mWatts), 32-bit integer.
05h	Read TDP	Read current Thermal Design Power Limit	None	ReadTDP (mWatts), 32-bit integer.
06h	Read Max cTDP	Read Max configured Thermal Design Power	None	ReadMaxcTDP (mWatts), 32-bit integer.
07h	Read Min cTDP	Read Min configured Thermal Design Power	None	ReadMincTDP (mWatts), 32-bit integer.
08h	Read BIOS Boost Fmax	Read BIOS Boost Fmax	[31:16]=Logical Core ID. [15:0]=Reserved.	ReadBIOSBoostFmax (MHz), 32-bit integer.
09h	Read APML Boost Limit	Read APML Boost Limit	[31:16]=Logical Core ID. [15:0]=Reserved.	ReadAPMLBoostLimit (MHz), 32-bit integer.
0Ah	Write APML Boost Limit	Write APML Boost Limit Operates on logical cores and not threads.	[31:16]=Logical Core ID. [15:0]=Frequency in MHz.	None
0Bh	Write APML Boost Limit All Cores	Write APML Boost Limit operates on all logical cores and no logical core requires specifying.	[15:0]=Frequency in MHz.	None
0Ch	Read Dram Throttle	Read Dram Throttle will always report the highest percentage value imposed by PROCHOT induced throttle or Write Dram Throttle or ODTs thermal event. This command does not factor in throttling due to TSOD (if enabled and triggered). The	None	ReadDramThrottle (% 0-100)

		value returned is the average across all channels.		
0Dh	Write Dram Throttle	Write Dram Throttle command provides a software path to throttle the command rate on all channels immediately. Once this command is issued, the software based throttle is disabled by writing a throttle value of 0. MP1 FW caps the throttle percentage to 80 when the input value is > 80.	WriteDramThrottle (% ,0-80).	None
0Eh	Read Prochot Status	Read PROCHOT Status.	None	ReadProchotStatus: 0=Not PROCHOT. 1=PROCHOT.
0Fh	Read Prochot Residency	Read PROCHOT Residency (since the boot time or last read of Prochot Residency). Return value is expressed as 16 fractional bits. Value of FFFFh represents 100% and 0000h represents 0% of time the socket spends in PROCHOT.	None	ReadProchotResidency (Percentage of time), 16 fractional bits.
10h	Reserved	N/A	N/A	N/A
11h	Reserved	N/A	N/A	N/A
12h	Reserved	N/A	N/A	N/A
13h	Read IOD Bist Result	Read IOD BIST status.	None	ReadIODBistResult: 0=BIST pass. 1=BIST fail.
14h	Read CCD Bist Result	Read CCD BIST status. Results are read for each CCD present in the system. The number of logical CCD instances may be determined by use of CUID reported CoreId[7:5].	Logical CCD instance number, 32-bit integer.	ReadCCDBistResult: 0=BIST pass. 1=BIST fail.
15h	Read CCX Bist Result	Read the BIST results of Cpu Core Complex. Results represent a bit vector of the BIST status for each physical core in the complex(n:0) and L3. The number of CCX in the socket and number of cores per CCX can be determined using CPUID functions.	Logical CCX instance number (0..K-1) where K is the number of CCXs, 32-bit integer.	ReadCCXBist Result: [31:16] = Core[n:0]. [1] = L3 X3D if X3D variant else 1 [0] = L3. Each bit is BIST result where 0 = BIST fail 1 = BIST pass Default bit value is 0 when core count is < 16.

16h	Read CCLK Frequency Limit	Provides the socket's CPU core clock (CCLK) frequency limit from the most restrictive infrastructure limit at the time of the request.	None	Frequency (MHz).
17h	Read Socket C0 Residency	Provides the average C0 residency across all cores in the socket. 100% specifies that all enabled cores in the socket are running in C0.	None	Socket C0 residency [%].
18h	Get Max DDR Bandwidth and Utilization	Provides per socket: 1. Theoretical maximum DDR Bandwidth in GB/s. 2. Current utilized DDR Bandwidth (Read+Write) in GB/s. 3. Current utilized DDR Bandwidth as a percentage of theoretical maximum.	None	[31:20]=Max bandwidth in GBps. [19:8]=Utilized bandwidth (R+W) in GBps. [7:0]=Utilized bandwidth in %.
1Ch	Get MP1 Firmware Version	Read the MP1 firmware version	None	MP1 firmware version, a 32b hex value as documented in BIOS release notes.
1Fh	Read PPIN fuse	Return the 64b PPIN value. MP1 firmware may return an abort code if the data cannot be accessed. BMC firmware must enforce appropriate policy controls for allowing PPIN access through APLM.	[31:1]=Reserved [0]=Hi or Lo 32b requested	[31:0]=PPIN[31:0] if DataIn[0]==0 else PPIN[63:32]
20h	Read Postcode	Read most recent postcode at specified offset. MP1 caches the 8 most recent postcodes. When input parameter is 0 MP1 will refresh the cache before returning the latest postcode. Index 0 is the most recent postcode; higher index refers to correspondingly older postcode.	[31:3]=Reserved [2:0]=Postcode offset	[31:0]=postcode for given offset
21h	Read RTC	Read RTC timer value. RTC time represents the year, month, day, hour, minute and seconds value in a 64b encoding.	[31:4]=Reserved [3:0]=4-byte aligned offset. The lowest 2 bits will be forced to 0 by MP1 firmware. 0:RTC Lo Dword 4:RTC Hi Dword	[31:0]=RTC value If DataIn==0 RTC=DD_hh_mm_ss If DataIn==4 RTC=00_YYYY_MM All digits in BCD format.
31h ¹	WRITE_FAST_PT_LIMIT	Set APU power limit for system power supply peak	[31:0]=fPPT in mW.	None

		control.		
34h ¹	WRITE_THERM_CTL_LIMIT	Set the thermal throttling limit.	[31:0]=Therm limit in degree Celsius.	None
35h ¹	WRITE_VRM_VDD_CURRENT_LIMIT	Set VDDCR_VDD TDC .	[31:0]=VDD TDC in mA.	None
36h ¹	WRITE_VRM_VDD_MAXIMUM_CURRENT_LIMIT	Set VDDCR_VDD EDC.	[31:0]=VDD EDC in mA.	None
40h	BMC_REPORT_DIMM_POWER_CONSUMPTION	Per-DIMM Power Consumption reported periodically by BMC (at rate of 10ms or less) when BMC owns the SPD side-band bus.	[31:17] DIMM Power (mWatt), [16:8] Update Rate (ms), [7:0] DIMM_ADDRESS	None See notes under Table 141 [APML DIMM ADDRESS Encodings] for DataIn and DataOut field details
41h	BMC_REPORT_DIMM_THERMAL_SENSOR	Per-DIMM Thermal Sensor (2 Sensors per DIMM) reported periodically by BMC (at rate of 10ms or less) when BMC owns the SPD side-band bus.	[31:21] Temperature (Degrees C), [16:8] Update Rate (ms), [7:0] DIMM_ADDRESS	None See notes under Table 141 [APML DIMM ADDRESS Encodings] for DataIn and DataOut field details
42h	BMC_RAS_PCIE_CONFIG_ACCESS	Read 32-bit Data from Extended PCI Config Space. NOTE: This command allows access to PCIe® Config Space, except for Seg0:Bus0:Dev18 and Seg0:Bus0:Dev19.	PCI address: [31:28]=Segment [27:16]=4-Byte aligned Offset (0..4092). [15:8]=Bus. [7:3]=Device. [2:0]=Function.	[31:0]=Data from Offset in PCIe Config Space.
43h	BMC_RAS_MCA_VALIDITY_CHECK	Return number of MCA Banks with Valid (non-zero) Status.	None	[31:16]=Unsigned Integer for number of bytes per MCA bank. [15:0]=Unsigned Integer for number of MCA Banks with Valid Status.
44h	BMC_RAS_MCA_MSR_DUMP	Read 32-bit of Data from MCA Bank reported by BMC_RAS_MCA_VALIDITYt_CHECK.	[31:16]=Unsigned Integer for 0-based Index of MCA Bank. [15:0]=4-Byte aligned Offset within MCA Bank (0..124).	[31:0]=Data from Offset in MCA Bank.
45h	BMC_RAS_FCH_RESET_REASON	Get FCH Reason Code from previous RESET. This command is typically used to get source of previous reset from BSP (socket 0). BMC can optionally choose to read the	[31:0]=Unsigned Integer for ID of FCH Register: 0=FCH::PM:: S5_RESET_STATUS . 1=FCH::PM:: BREAK_EVENT .	[31:0]=Data from corresponding FCH Register.

		reason code from AP (socket 1).	Other values are Reserved.	
46h	GET_DIMM_TEMPERATURE_RANGE_AND_REFRESH_RATE	Get Per-DIMM Temperature Range and Refresh Rate for a given channel based on inband MR4 polling. The refresh rate is determined by the warmest DRAM device across all DIMMs in the channel (highest MR4.OP[2:0] value) and the refresh rate threshold programmed in the UMC. Refer to JEDEC DDR5 SDRAM specification (JESD79) for translation of MR4.OP[2:0] to temperature range.	[7:0] DIMM_ADDRESS See notes under Table 141 [APML DIMM ADDRESS Encodings] for DataIn and DataOut field details.	[31:4]=Reserved [3]=Refresh Rate 0=1x 1=2x. [2:0]=Temperature Range (MR4.OP[2:0]) of hottest DRAM device on given channel.
47h	GET_DIMM_POWER_CONSUMPTION	Get Per-DIMM Power Consumption. This is applicable when BMC does not own the SPD side-band bus.	[7:0] DIMM_ADDRESS See notes under Table 141 [APML DIMM ADDRESS Encodings] for DataIn and DataOut field details	[31:17] DIMM Power (mWatt), [16:8] Update Rate (ms), [7:0] DIMM_ADDRESS
48h	GET_DIMM_THERMAL_SENSOR	Get Per-DIMM Thermal Sensor (2 Sensors per DIMM). This is applicable when BMC does not own the SPD side-band bus.	[7:0] DIMM_ADDRESS See notes under Table 141 [APML DIMM ADDRESS Encodings] for DataIn and DataOut field details	[31:21] Temperature (Degrees C), [16:8] Update Rate (ms), [7:0] DIMM_ADDRESS
49h	PWR_CURRENT_ACTIVE_FREQUENCY_LIMIT_SOCKET	Get Current Active Frequency Limit per Socket.	None	[31:16]=Frequency (MHz). [15:0]=Source of Limit. As shown in Table 136 [APML Frequency Limit Source].
4Ah	PWR_CURRENT_ACTIVE_FREQUENCY_LIMIT_CORE	Get Current CCLK limit set per Core.	Logical Core ID, 32-bit integer	[31:16]=Reserved. [15:0]=BaseFreq (MHz).
4Bh	PWR_SVI_TELEMETRY_ALL_RAILS	Get SVI-based Telemetry for all rails.	None	[31:0]=Power (mW).
4Ch	GET_SOCKET_FREQUENCY_RANGE	Get Fmax and Fmin per Socket.	None	[31:16]=Fmax (MHz). [15:0]=Fmin (MHz).
4Dh	GET_CURRENT_IO_BANDWIDTH	Get Current Bandwidth on IO Link.	[15:8]=Link ID: As shown in Table 139	[31:0]=IO BandWidth (Mbps).

	TH		[APML Link ID Encoding]. [7:3]=Reserved. [2:0]=Bandwidth Type: As shown in Table 137 [APML IO Bandwidth Encoding].	
4Eh	GET_CURRENT_XGMI_BANDWIDTH	Get Current Bandwidth on xGMI Link.	[15:8]=Link ID: As shown in Table 139 [APML Link ID Encoding]. [7:3]=Reserved. [2:0]=Bandwidth Type: As shown in Table 138 [APML XGMI Bandwidth Encoding].	[31:0]=xGMI bandwidth (Mbps).
4Fh	GMI3_LINK_WIDTH_RANGE	Set or Get Max and Min width of GMI3 Link as follows: 0=Quater Width. 1=Half Width. 2=Full Width.	[31] = Set or Get 0 : set the range 1 : get the range If DataIn[31] = 0 [15:8]=Min link width. [7:0]=Max link width. Else [15:0] = Reserved. NOTE: Max value must be >= Min value. Invalid configurations result in a Failed response.	[31:16] = Reserved. [15:8] = Min link width. [7:0] = Max link width. These will be the updated values if DataIn[31]=0.
50h	XGMI_LINK_WIDTH_RANGE	Set or Get Max and Min width of xGMI Link as follows: 0=x4. 1=x8. 2=x16.	[31] = Set or Get 0 : set the range 1 : get the range If DataIn[31] = 0 [15:8]=Min link width. [7:0]=Max link width. Else [15:0] = Reserved NOTE1: Max value must be >= Min value. Invalid configurations result in a Failed response. NOTE2: If this function is called from both Primary and Secondary sockets, the largest Width values from either call are	[31:16] = Reserved. [15:8] = Min link width. [7:0] = Max link width. These will be the updated values if DataIn[31]=0.

			used.	
51h	APB_DISABLE	<p>Messages APBEnable and APBDisable specify DF (Infinity Fabric™) P-State behavior. DF P-States specify the frequency of clock domains from the CPU core boundary through to and including system memory, where DF P0 has highest performance and higher numbered P-states have lower performance.</p> <p>By default, an algorithm adjusts DF P-States automatically in order to optimize performance. However, this default may be changed to a fixed DF P-State through a CBS option at boot-time. APBDisable may also be used to disable this algorithm and force a fixed DF P-State.</p> <p>NOTE: While the socket is in PC6 or if PROCHOT_L is asserted, the lowest DF P-State (highest value) is enforced regardless of the APBEnable/APBDisable state.</p> <p>See Note 2 at the end of this table about arbitration of the active DF Pstate range.</p>	<p>[31] = Set or Get. 0 : set APB_DISABLE 1 : get APB state [30:8] = Reserved. If DataIn[31] = 1 [7:0] = Reserved. Else [7:0] DF P-state Valid values: 0, 1, 2</p>	<p>[31:9] = Reserved [8] = APB state 1 : APB is disabled. 0 : APB is enabled If DataOut[8] = 1 [7:0] = DF P-state locked value. Else [7:0] = Reserved.</p>
52h	APB_ENABLE	<p>This function enables the DF P-State Performance Boost algorithm. See the APBDisable function for more information.</p> <p>See Note 2 at the end of this table about arbitration of the active DF Pstate range.</p>	None	None
53h	GET_CURRENT_DFPSTATE_FREQUENCY	Provides Infinity Fabric Clock (FCLK), DRAM Memory Clock (MEMCLK), and UMC Clock Divider (UCLK) for the current socket DF P-state.	None	<p>[31:16]=MEMCLK (MHz). [15]=UCLK. (0=Divide by 1. 1=Divide by 2.) [14:12]=Reserved. [11:0]=FCLK (MHz).</p>

54h	SET_LCLK_DPM_LEVEL_RANGE	Sets the Max & Min LCLK DPM Level on a given NBIO per socket. The DPM Level is an encoding to represent the PCIe Link Frequency (LCLK) under a root complex (NBIO), as shown in Table 140 [APML LCLK Frequency Per DPM Level]. NOTE: These Levels can also be set from HSMP; the last value received from either APML or HSMP is enforced.	[23:16]=NBIO ID (0 to 3), [15:8]=Max DPM Level (0 to 3), [7:0]=Min DPM Level (0 to 3). NOTE: Max value must be \geq Min value.	None
55h	BMC_GET_RAPL_UNITS	Get RAPL (Running Average Power Limit) Units: Scaling factor for energy (in Joules) is $1/(2^{ESU})$, and for time (in seconds) is $1/(2^{TU})$. ESU: energy status units TU: time units	None	[31:20]=Reserved. [19:16]=TU Value. [15:13]=Reserved. [12:8]=ESU Value. [7:0]=Reserved.
56h	BMC_GET_RAPL_CORE_LO_COUNTER	Get RAPL (Running Average Power Limit) Lo-Word of Energy Counter per Core. Energy = Counter * ESU Multiplier (Joules). See 5.4.2.2.3 [Reading RAPL Energy Counters].	Logical Core ID, 32-bit integer.	[31:0]=Core Lo_Count Register Value.
57h	BMC_GET_RAPL_CORE_HI_COUNTER	Get RAPL (Running Average Power Limit) Hi-Word of Energy Counter per Core. Energy = Counter * ESU Multiplier (Joules). See 5.4.2.2.3 [Reading RAPL Energy Counters].	Logical Core ID, 32-bit integer.	[31:0]=Core Hi_Count Register Value.
58h	BMC_GET_RAPL_PKG_COUNTER	Get RAPL (Running Average Power Limit) Lo/Hi-Word of Energy Counter per Package. Energy = Counter * ESU Multiplier (Joules). See 5.4.2.2.3 [Reading RAPL Energy Counters].	0=Read Lo_Count Register. 1=Read Hi_Count Register.	[31:0]=Pkg Lo/Hi_Count Register Value.
59h	BMC_GET_CPU_BASE_FREQUENCY	Get Base Frequency per CPU socket.	None	[31:16]=Reserved. [15:0]=BaseFreq (MHz).
5Ah	BMC_CONTROL_PCIE_LINK_RATE	Set or Get the Link Rate on PCIe devices. Note that if CXL™ device operating at Gen5 rate is present on any link then PCIe rate is locked at Gen5 rate and request to	[31] = Set or Get 0 : set link rate 1 : get link rate [30:8] = Reserved If DataIn[31] = 1 [7:0] = Reserved	0=Auto-Detect. 1=Limit at Gen4. 2=Limit at Gen5. This will be the updated value if

		change to Gen4 rate will be ignored.	Else [7:0] = Link Rate encoding : 0=Auto-Detect bandwidth utilization, and set Link Rate accordingly. 1=Limit at Gen4 Rate. 2=Limit at Gen5 Rate.	DataIn[31] = 0.
5Bh	BMC_RAS_DBG_LOG_VALIDITY_CHECK	Return the number of Dbg Log instances of type Dbg Log Block ID that have an error log to report. Every instance of a given Block ID will have the same length of error log. The length of error log can vary between Block IDs.	[31:8]=Reserved [7:0]=Dbg Log Block ID	[31:25]=Reserved [24:16]=Unsigned integer, length of error log in bytes per instance (0, 4, 8, 12, ... 256) [15:9]=Reserved [8:0]=Unsigned integer, number of Dbg Log instances that have an error log to report (0..256). 0 means no instance has anything to report.
5Ch	BMC_RAS_DBG_LOG_DUMP	Read 32b of Data from Dbg Log instance if BMC_RAS_DBG_LOG_VALIDITY_CHECK reported non-zero instances.	[31:24]=Reserved [23:16]=Unsigned integer, 0-based index of Dbg Log instance (0..255) [15:8]=Dbg Log Block ID [7:0]=4-byte aligned offset in error log (0, 4, 8, 12, ...)	[31:0]=4 byte data from requested offset of Dbg Log Block instance
5Dh	PWR_EFFICIENCY_MODE_SELECTION	Select or get current power efficiency profile policy. See Note 2 at the end of this table about arbitration of the active DF Pstate range.	[31] = Set or Get 0 = Set policy 1 = Get policy [30:3] = Reserved. If DataIn[31] = 1 [2:0] = Reserved. Else [2:0] = Mode Selection: 000b = High Performance Mode. 001b = Power Efficiency Mode. 010b = IO Performance Mode. 011b = Balanced Memory Performance Mode. 100b = Balanced Core	[31:3] = Reserved [2:0] = Current power efficiency mode. This is the updated value if DataIn[31] = 0.

			Performance Mode. 101b = Balanced Core and Memory Performance Mode. Other = Reserved. See 5.4.2.2.1 [APML Performance Modes].	
5Eh	DF_PSTATE_RANGE	Set or Get the Max and Min DF P-State. Given the DF_PSTATE encodings, Max value must be \leq Min value. See Note 2 at the end of this table about arbitration of the active DF Pstate range.	[31] = Set or Get [30:16]=Reserved. If DataIn[31] = 1 [15:0] = Reserved. Else [15:8]=Min DF P-State. [7:0]=Max DF P-State. P_State values are: 0 – DFP0 1 – DFP1 2 – DFP2	[31:16] = Reserved [15:8] = Min DF P-State. [7:0] = Max DF P-State. This will be the updated values if DataIn[31] = 0.
5Fh	GET_LCLK_DPM_LEVEL_RANGE	Gets the Max & Min LCLK DPM Level on a given NBIO per socket. The DPM Level is an encoding to represent the PCIe Link Frequency (LCLK) under a root complex (NBIO), as shown in Table 140 [APML LCLK Frequency Per DPM Level]. NOTE: These Levels can also be set from HSMP; this message returns the current Levels which may have been set from either APML or HSMP.	[23:16]=NBIO ID (0 to 3).	[23:16]=NBIO ID (0 to 3) [15:8]=Max DPM Level (0 to 3), [7:0]=Min DPM Level (0 to 3).
60h	GET_UCODE_REVISION	Get microcode revision number. This is revision number reported from IA32_BIOS_SIGN_ID MSR using CPUID instruction. The command may return an Abort error code if the MSR is not accessible (eg. if core is not out of reset)	N/A	[31:0] = Microcode revision value from IA32_BIOS_SIGN_ID MSR
61h	BMC_RAS_RUNTIME_ERROR_VALIDITY_CHECK	Returns number of valid error instances per category: MCA, DRAM CECC, or PCIE. The number of instances with valid errors reported depends on the Interrupt/Polling bit specified in DataIn[31]. If	[31] = Request for Interrupt or Polling based data 0: polling mode 1: interrupt mode [1:0]=Runtime Error Category: 00 = MCA	MCA: [31:16]=Unsigned Integer for number of bytes per MCA bank. [15:0]=Unsigned Integer for number of MCA Banks with Valid Status.

		the bit is 0 (polling mode) then firmware will poll and report the number of instances of the given category that have valid error status. If the bit is 1 (interrupt mode), then firmware will report only those instances of the given category where the error count has met or exceeded the Error Count Threshold (see command 63h).	01 = DRAM CECC 10 = PCIe 11 = Reserved	<p>DRAM CECC: [31:16] = Unsigned integer for number of bytes of DRAM CECC state [15:0] = Unsigned integer for number of valid/non-zero DRAM CECC counters</p> <p>PCIe: [31:16] = Unsigned integer for number of bytes of state per OOB Root-port instance [15:0] = Unsigned Integer for number of Root-Port instances containing valid PCIe error</p>
62h	BMC_RAS_GET_RUNTIME_ERR_INFO	<p>MCA: Read 32-bit of Data from MCA MSR Bank</p> <p>DRAM CECC: Returns Error count, counter info and corresponding UMC_MCA information from the valid DRAM ECC Correctable Error counter instance</p> <p>PCIe: Returns 32 bit PCIe error data from given offset of given instance</p>	<p>[31:24] = Reserved [23:16] = 0-based index of valid instance returned by BMC_RAS_RUNTIME_ERR_VALIDITY_CHECK [15:8] = Runtime error category [7:0] = 4-byte aligned offset in instance</p> <p>See Table 142 [Runtime Error Info Encoding] for valid offset value for each category</p>	See Table 142 [Runtime Error Info Encoding] for return value for given offset for each category
63h	BMC_RAS_SET_ERR_THRESH	Requests MP1 firmware to configure thresholding counters for MCA, DRAM CECC or PCIe	<p>[0:1]=Error Type 00 = MCA 01 = DRAM CECC 10 = PCIe 11 = Reserved [17:2]=Error Count Threshold [21:18]=Max Interrupt Rate</p> <p>Note: Error Count Threshold is capped at 0xFFF for MCA 0xFFFF for DRAM CECC and</p>	N/A

			0x1 for PCIe. Setting Error Count Threshold to 0 disables the interrupt.	
64	BMC_RAS_SET_OOB_CONFIG	Configures OOB state infrastructure in the SoC	See Table 143 [Set OOB Config DataIn] for DataIn encoding	N/A
65h	BMC_RAS_GET_OOB_CONFIG	Returns current status of OOB state configuration in the SoC	N/A	See Table 144 [Get OOB Config DataOut] for DataOut encoding
66h-67h	Reserved	N/A	N/A	N/A
68h	BMC_PCIE_CONFIG_WRITE	<p>Write 32-bit DataIn to Extended PCI Config Space of target specified in previous BMC_RAS_PCI_CONFIG_ACCESS command.</p> <p>MP1 firmware will restrict the writes to SFI (System Firmware Intermediary) extended capability registers and return SB-RMI error code "Invalid input parameter" if offset does not target SFI extended capability.</p> <p>BMC is expected to know the offset of the SFI extended capability in the configuration space of the PCIe target.</p>	<p>[31:0] : Data value to be written.</p> <p>Note that byte enables are not supported. To modify a subset of bytes, BMC must read, modify and write at 4 byte granularity.</p>	Return code. 0 : Success any other value: Reserved
69h	Reserved	N/A	N/A	N/A
6Ah	BMC_RAS_DELAY_RESET_ON_SYNCFLOOD_OVERRIDE	Overrides the delay value set via BIOS for the current boot instance. Delay value reverts to BIOS config setting after reboot. Limited to 5 override requests per boot instance. Command will be rejected if sent more than once per minute.	<p>[31:10]= Reserved [9]=StopDelayCounter . If set, stops the active delay countdown, which extends the DelayResetCpuOnSyncFlood indefinitely; system will not reset. The CPU processes this bit only during the SyncFlood condition. This action is final; counter cannot be resumed. Use mailbox command BMC_RAS_RESET_</p>	<p>0 = NACK. MP1 firmware will not honor the override request 1 = ACK. MP1 firmware will honor the override request.</p>

			ON_SYNCFLOOD or toggle KBRESET_L pin if system reset is desired at any point after stopping ResetCpuOnSyncFlood counter. [8]=DisableDelayCounter. If set, ResetCpuOnSyncFlood response will not be delayed in the next SyncFlood, regardless of the delay value. Overriding this value takes effect at runtime; before a SyncFlood condition. [7:0]=Override delay value. Range: 5-255 mins. Overriding this value only takes effect at runtime; before a SyncFlood condition.	
6Bh	BMC_RAS_RESET_ON_SYNCFLOOD	Requests warm reset after syncflood. Reset only works during syncflood condition	N/A	0 = NACK. MP1 firmware will not proceed with reset 1 = ACK. MP1 firmware will proceed with reset
70h	SPD_SB_RD	Execute a four byte read transaction at a given register offset in a specified device on the target DIMM. The target DIMM is specified by the DIMM_ADDRESS described in Table 141 [APML DIMM ADDRESS Encodings]. The local identifier (LID) of the device on the DIMM is LID described in the JEDEC SidebandBus spec (JESD403)	[7:0] = DIMM_ADDRESS [11:8] : LID of device [22:12] : Register offset in given reg space [23] : Register space 0 = Volatile space 1 = NVM space [31:24] : Reserved Note: NVM register space is present only in some devices on SPD SB bus.	[31:0] : Read Data Byte[3:0]
73h	XGMI_PSTATE_RANGE	Set or Get the Max and Min XGMI P-State. P_State values are: 0 – DFP0 (high performance)	[31] = Set or Get 0 : Set 1 : Get [30:16]=Reserved. If DataIn[31] = 1 [15:0] = Reserved.	[31:16] = Reserved [15:8] = Min XGMI P-State. [7:0] = Max XGMI P-State.

		1 – DFP1 (low performance) Max value must be <= Min value. XGMI P-state can be set through APML or HSMP; the value that was last set is enforced.	Else [15:8]=Min XGMI P-State. [7:0]=Max XGMI P-State.	These will be the updated values if DataIn[31] = 0.
74h	CPU_RAIL_ISO_FREQ_POLICY	For core clock frequency limiters specific to CPU power rails (e.g. TDC), this policy allows or disables independent core clocks per rail (VDDCR_CPU0 or VDDCR_CPU1). If a socket wide limit (e.g. PPT) is setting the core clock frequency, then this setting has no effect.	[31]=Set or Get Policy 0: Set policy 1: Get policy [31:1]=Reserved [0]=Enable independent control 0: all cores on both rails have same frequency limit 1: each rail has independent frequency limit	[31:1] = Reserved [0] = Current policy 0 = both rails have same frequency limit. 1 = each rail has independent frequency limit. This will be the updated value if DataIn[31] = 0.
76h	DFC_ENABLE	Set or get DF C-state enabling control. DF C-state is a low power state for IOD.	[31] = Set or Get. 0: Set DFC control 1: Get DFC control [30:1] = Reserved [0] = Enable or Disable for set, Reserved for get. 0: Disable DFC 1: Enable DFC	[31:1] = Reserved [0] = current DFC control setting. This will be the updated value if DataIn[31] = 0.
78h	Read DRAM Throttle (Enhanced version)	Read the DRAM throttle percentage of the given channel. The returned value is the current throttle percentage occurring on the channel after arbitration among the PROCHOT induced, TSOD induced (if enabled), APML SW throttle or inband MR4 based throttle.	[31] = Average flag 1: Return average across all populated channels 0: Return throttle value for given channel [30:8] = Reserved [7:0] = Reserved if Average flag = 1, else DIMM_ADDRESS (mode 1 only) described in Table 141 [APML DIMM ADDRESS Encodings] .	[31:0] = DRAM Throttle (% , 0-100)

Notes:

1. Family 1Ah Models 10-1Fh processors do not support commands in the command code range 30h-3Fh.
2. APML commands APB_DISABLE, APB_ENABLE, PWR_EFFICIENCY_MODE_SELECTION, SET_DF_PSTATE_RANGE, as well as HSMP commands APBDisable, APBEnable,

PwrEfficiencyModeSelection, SetDfPstateRange all influence the active DF Pstate range. The last one of these HSMP or APML commands sent will override any previous setting of the active DF Pstate range, including overriding any active DF Pstate range set by BIOS options.

Table 136: APML Frequency Limit Source

Source Type Encoding (DataOut[15:0])	Description
DataOut[0]=1	cHTC-Active
DataOut[1]=1	PROCHOT
DataOut[2]=1	Thermal Designed Current (TDC) Limit
DataOut[3]=1	Package Power Tracking (PPT) Limit
DataOut[4]=1	OPN Max
DataOut[5]=1	Reliability Limit (Fused Max or Reliability Monitor Fmax@Vmax)
DataOut[6]=1	APML Agent
DataOut[7]=1	HSMP Agent
DataOut[15:8]	Reserved

Note: There may be more than one active limiter.

Table 137: APML IO Bandwidth Encoding

Encoding	Bandwidth Type
001b	Aggregate bandwidth
Other	Reserved

Table 138: APML XGMI Bandwidth Encoding

Encoding	Bandwidth Type
001b	Aggregate bandwidth (bi-directional bandwidth per link)
010b	Read bandwidth (uni-directional bandwidth per link)
100b	Write bandwidth (uni-directional bandwidth per link)
Other	Reserved

Note: max theoretical bandwidth per link = xGMI speed * link width * MULT, where MULT = 1 (uni-dir) or 2 (bi-dir). For a x16 link operating at 32Gbps, max theoretical aggregate bandwidth = 1024 Gbps (32*16*2)

Table 139: APML Link ID Encoding

Encoding	Link Type
00000001b	P0
00000010b	P1
00000100b	P2
00001000b	P3
00010000b	G0
00100000b	G1
01000000b	G2
10000000b	G3

Table 140: APML LCLK Frequency Per DPM Level

DPM Level	LCLK Frequency
0	Lowest Freq.
1..3	Highest Freq.

Table 141: APML DIMM ADDRESS Encodings

DIMM_ADDRESS[7:0] (Mode = 0)	DIMM_ADDRESS[7:0] (Mode = 1)
Bit[7]=0 (for Mode 0) Bit[6]=Temp Sensor#: TS0/1, else Reserved. Bit[5:4]=I2/I3C Port# Bit[3]=Reserved. Bit[2:0]=DIMM SPD Host ID. Refer to JEDEC SPD5118, SPD5108 Hub and Serial Presence Detect Device Standard (JESD300-5).	Bit[7]=1 (for Mode 1) Bit[6]=Temp Sensor#: TS0/1, else Reserved. Bit[5]=Reserved Bit[4]=DIMM#: DIMM0/1. Bit[3:0]=UMC/DDRPHY Instance ID. Refer to 9.1.1.1 [UMC and DDR Phy Logical Mapping].

Notes:

- 1: DIMM Power is a 15-bit unsigned value representing power consumed in mW (0-32767)
- 2: Refresh rate is the time between successive sampling from given DIMM (0-511ms). 0 means sampling time < 1ms, and 511 means sampling time >= 511ms.
- 3: DIMM_ADDRESS is encoded per above Table 141 [APML DIMM ADDRESS Encodings].
- 4: Temperature (Deg C) is an 11-bit signed value, with a scaling factor of 0.25.
Thus 3FFh=255.75 (1023*0.25), 400h= -256 (-1024*0.25), 1h=0.25 and 7FFh= -0.25 (-1*0.25).

Table 142: Runtime Error Info Encoding

Error Category	DataIn	Offset	DataOut
0=MCA Banks	Offset range = 0..124	0..124	32b data of corresponding MCA bank (0..31)
1=DRAM ECC	Offset range = 0..128	0	[15:0]=Error Count [19:16]=Channel number [20]=Sub channel [22:21]=Chip-select number [25:23]=Rank multiplier number (Plr) [31:26]=Reserved
		4,8..128	32b data of corresponding UMC MCA bank (0..31)
2=PCIE	Offset range = 0..60	0,4..60	32b PCIe error data from corresponding offset

Table 143: Set OOB Config DataIn

Bit Number	Description
0	MCA OOB MISC0 Error Counter Enable: 0 = disable, 1 = enable
2:1	DRAM CECC OOB Error Counter Mode: 00 = disable, 01 = enable in NoLeakMode, 10 = enable in LeakMode, 11 = Reserved
7:3	DRAM CECC OOB Leak Rate = valid values: 00h – 1Fh
8	PCIe OOB Error Reporting Enable: 0 = disable, 1 = enable
30:9	Reserved
31	MCA OOB Error Reporting Enable: 0 = disable, 1 = enable

Table 144: Get OOB Config DataOut

Bit Number	Description
0	MCA OOB MISC0 Error Counter Enable value
2:1	DRAM CECC OOB Error Counter Mode value
7:3	DRAM CECC OOB Leak Rate value
8	PCIe OOB Error Reporting Enable value
10:9	Reserved
11	MCA Thresholding Interrupt Enable value

12	DRAM CECC Thresholding Interrupt Enable value
13	PCIe error Thresholding Interrupt Enable value
14	Reserved
18:15	MCA MaxIntRate ¹ value
22:19	DRAM CECC MaxIntRate ¹ value
26:23	PCIe error MaxIntRate ¹ value
30:27	Reserved
31	MCA OOB Error Reporting Enable value

1. Refer to docRAS for MaxIntRate encoding

5.4.2.2.1 APML Performance Modes

Performance modes can be set through [APML](#) or HSMP; the value that was last set is enforced.

- High Performance Mode: this mode will choose the default [DF](#) P-State algorithm and will try to maximize application performance using the application characteristics. This mode favors core performance and the system boots into this mode. In this mode all DF P-States are available and the default DF P-State and DLWM algorithms are active.
- Power Efficiency Mode: this mode configures the system for power efficiency at the expense of some performance. This mode limits the boost frequency available to the cores and restricts the DF P-States available in the system. This mode also monitors the system load to dynamically adjust performance for maximum power efficiency.
- IO Performance Mode: this mode sets up the Infinity Fabric to maximize the IO sub-system performance. This mode can result in lower core performance in some cases, to maximize IO throughput.
- Balanced Memory Performance Mode: this mode biases the memory subsystem and Infinity Fabric performance towards efficiency, by lowering the frequency of the fabric and the width of the [xGMI](#) links under light traffic conditions. Core behavior is unaffected. There may be a performance impact under lightly loaded conditions for memory-bound applications compared to the default high performance mode. With higher memory and fabric load, the system becomes similar in performance to the default high performance mode.
- Balanced Core Performance Mode: this mode biases toward consistent core performance across varying core utilization levels, by preventing active cores from using the power budget of inactive cores. This mode allows core "boosting" as in the default high performance mode, but does not allow core boost to take advantage of the power budget of inactive cores, resulting in a more efficient operating point for the active cores. The memory subsystem and Infinity Fabric behavior is unaffected. There may be a performance impact under light core utilization conditions compared to the default high performance mode. With high core utilization levels, the performance is similar to the default high performance mode.
- Balanced Core and Memory Performance Mode: this mode combines the Balanced Memory Performance and the Balanced Core Performance mode and may result in lower performance under light loads compared to the default high performance mode, but with significant increase in efficiency under light loads. Performance in this mode will be similar to the default high performance mode as the system load increases.

5.4.2.2.2 SB-RMI Mailbox Sequence

The sequence is as follows:

1. The initiator (BMC) indicates that command is to be serviced by firmware by writing 80h to SBRMI::InBndMsg_inst7 (SBRMI_x3F). This register must be set to 80h after reset.
2. The initiator (BMC) writes the command to SBRMI::InBndMsg_inst0 (SBRMI_x38).
3. For Write operations or Read operations, which require additional addressing information as shown in Table 135 [SB-RMI Soft Mailbox Message] above, the initiator (BMC) writes Command Data In[31:0] to SBRMI::InBndMsg_inst[4:1] {SBRMI_x3C(MSB):SBRMI_x39(LSB)}.
4. The initiator (BMC) writes 01h to SBRMI::SoftwareInterrupt to notify firmware to perform the requested Read or Write command.

5. Firmware reads the message and performs the defined action.
6. Firmware writes the original command to outbound message register SBRMI::OutBndMsg_inst0 (SBRMI_x30).
7. Firmware writes SBRMI::Status[SwAlertSts] = 1 to generate an ALERT (if enabled) to initiator (BMC) to indicate completion of the requested command. Firmware must (if applicable) put the message data into the message registers SBRMI::OutBndMsg_inst[4:1] {SBRMI_x34(MSB):SBRMI_x31(LSB)}.
8. Firmware clears the interrupt on SBRMI::SoftwareInterrupt.
9. For a Read operation, the initiator (BMC) reads the firmware response Command Data Out[31:0] from SBRMI::OutBndMsg_inst[4:1] {SBRMI_x34(MSB):SBRMI_x31(LSB)}.
10. BMC must write 1'b1 to SBRMI::Status[SwAlertSts] to clear the ALERT to initiator (BMC). It is recommended to clear the ALERT upon completion of the current mailbox command.

Table 145: SB-RMI Soft Mailbox Error Code

Error Type	Description	Code
No error	Mailbox message command executed successfully without an error.	00h
Command Aborted	Mailbox message command was aborted due to internal error. DataOut must be ignored.	01h
Unknown Command	Unknown mailbox message.	02h
Invalid Core	Invalid core is specified in mailbox message parameters.	03h
Command Failed with Error	Mailbox message command failed. Additional error information available in DataOut.	05h
Invalid Input Arguments	Mailbox message command failed due to invalid input arguments.	09h
Invalid OOB RAS Config	Mailbox message command failed due to improper or invalid settings of OOB RAS Config fields.	0Ah
Data Not Ready	Data collection in progress. Command should be retried later.	0Bh

The mailbox error code is written by Firmware in SBRMI::OutBndMsg_inst7 (SBRMI_x37).

5.4.2.2.3 Reading RAPL Energy Counters

The following procedure is recommended to detect and correct a potential overflow when reading 64-bit RAPL counters with separate 32-bit Lo-Word and 32-bit Hi-Word reads via SB-RMI Mailbox messages:

1. Read the Hi-Word.
2. Read the Lo-Word.
3. Read the Hi-Word again and compare it to the previous Hi-Word value.
4. If the Hi-Word values are equal, keep the previous Lo-Word value.
5. Else, read the Lo-Word again and keep the last Hi-Word value.

5.4.2.3 SB-RMI Boot code status

Boot code will dump the dynamic boot status into SBRMI::MP0OutBndMsg. BMC can then just read this status through SBI interface to determine progress through the boot flow.

5.4.2.4 SB-RMI Register Access

The SB-RMI registers can be read or written from the I3C interface by using the I3C Private Write or Private Read protocols with the SB-RMI register address indicated in the command code, which follows the I3C dynamic target address. Both Block Read/Write and Single Read/Write can also be supported.

Block Read/Write protocol access for SB-RMI registers is controlled by SBRMI::Control[BlkRWEn]. The SB-RMI

interface supports Block Writes of up to 32 bytes, and Block Reads of up to 32 bytes as specified by `SBRMI::ReadSize[RdSize]`. Bytes are returned in ascending register order starting with the first SB-RMI register in the command byte. Some rules for write and read are listed as follows.

1. SB-RMI register offsets within `x00h-x0Fh`, can only support for Single Write if it is writable.
2. SB-RMI register offsets within `x10h-x5Fh` or `x80h-xDFh`, if it is writable it can support for Single Write and Block Write based on `SBRMI::Control[BlkRWEn]`.
3. SB-RMI register offsets within `x00h-x5Fh` or `x80h-xDFh`, if it is readable it can support for Single Read and Block Read.
 - If `SBRMI::Control[BlkRWEn]` is zero or `SBRMI::ReadSize[RdSize]` is zero or 1, only supports Single read by return 1 byte of read data.
 - If `SBRMI::Control[BlkRWEn]` is set and `SBRMI::ReadSize[RdSize]` is greater than 1, supports Block read by return `SBRMI::ReadSize[RdSize]` bytes of read data with extra 1 byte of "byte count".
4. SB-RMI register offsets within `x70h-x7Fh`, Single Read/Write access to these offsets is not supported:
 - Offsets `x71h`, `x72h` and `x73h` are used for processor state access; refer to 5.4.2.1 [SB-RMI Processor State Access] for the access sequence.
 - Offsets `x70h`, `x74h` - `x7fh` are reserved for future expansion; access to these offsets is undefined.

5.4.2.4.1 SB-RMI Register Block Access

The following example shows a write from `SBRMI_x18` to `SBRMI_x1F` using SB-RMI block write with `SBRMI_x01[BlkRWEn]` set to 1.

Table 146: SB-RMI Register Block Write Protocol

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command Code	18h	Indicates starting register <code>SBRMI_x0018</code> .
3	Command Code	00h	Indicates starting register <code>SBRMI_x0018</code> .
4	Byte Count	08h	Number of bytes to be written.
5	Data Byte 1	00h	Write a value to <code>SBRMI_x18h</code> .
6	Data Byte 2	00h	Write a value to <code>SBRMI_x19h</code> .
7	Data Byte 3	00h	Write a value to <code>SBRMI_x1Ah</code> .
8	Data Byte 4	00h	Write a value to <code>SBRMI_x1Bh</code> .
9	Data Byte 5	00h	Write a value to <code>SBRMI_x1Ch</code> .
11	Data Byte 6	00h	Write a value to <code>SBRMI_x1Dh</code> .
12	Data Byte 7	00h	Write a value to <code>SBRMI_x1Eh</code> .
13	Data Byte 8	00h	Write a value to <code>SBRMI_x1Fh</code> .

The following example shows a read from `SBRMI_x10` to `SBRMI_x17` using SBI block read with `SBRMI_x01[BlkRWEn]` set to 1 and `SBRMI_x03[RdSize]` set to 8.

Table 147: SB-RMI Register Block Read Protocol

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command	10h	Indicates starting register <code>SBRMI_x0010</code> .
2	Command	00h	Indicates starting register <code>SBRMI_x0010</code> .
3	Dynamic Address	XXXXXXXX1b	Target Address (Read).
4	Byte Count	08h	Number of bytes returned.
5	Data Byte 1	00h	Write a value to <code>SBRMI_x10h</code> .
6	Data Byte 2	00h	Write a value to <code>SBRMI_x11h</code> .
7	Data Byte 3	00h	Write a value to <code>SBRMI_x12h</code> .

8	Data Byte 4	00h	Write a value to SBRMI_x13h.
9	Data Byte 5	00h	Write a value to SBRMI_x14h.
10	Data Byte 6	00h	Write a value to SBRMI_x15h.
11	Data Byte 7	00h	Write a value to SBRMI_x16h.
12	Data Byte 8	00h	Write a value to SBRMI_x17h.

5.4.2.4.2 SB-RMI Register Byte Access

The following example shows a write to SBRMI_x03 using the SBI single write protocol.

Table 148: SB-RMI Register Write Byte Protocol

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command	03h	Indicates register SBRMI_0x0003.
3	Command	00h	Indicates register SBRMI_0x0003.
4	Data Byte	04h	Write a value of 04h.

The following example shows a read from SBRMI_x03 using the SBI single read with SBRMI_x01[BlkRWEn] set to 0.

Table 149: SB-RMI Register Read Byte Protocol

Byte	Byte Name	Value	Notes
1	Dynamic Address	XXXXXXXX0b	Target Address (Write).
2	Command	03h	Indicates register SBRMI_0x0003.
3	Command	00h	Indicates register SBRMI_0x0003.
3	Dynamic Address	XXXXXXXX1b	Target Address (Read).
4	Data Byte	04h	Value of SBRMI_x0003h.

5.4.2.5 SB-RMI Alert

The processor alerts the SBI when a Machine Check Exception occurs within the system. The Machine Check Exception status is reflected in registers SBRMI_x01[F:0] and SBRMI_x05[F:0]. To enable this functionality, SBRMI_x02[F:0] and SBRMI_x0C[F:0] must be clear.

The processor alerts the SBI on system fatal error event. This status is reflected in SBRMI_x02[SwAlertSts]. To enable this functionality, SBRMI_x01[SwAlertMask] must be clear. For this case, SBRMI::RasStatus[fatal_err] is set. controller can issue a Read command to dump this register after Alert detection, clears it by writing 1'b1 to the corresponding bit. For the SBRMI::RasStatus register definition, please see SB-RMI Registers section for the details.

The processor alerts the SBI on _x71/ _x72/ _x73 command completion. This status is reflected in SBRMI_x02[HwAlertSts]. To enable this functionality, SBRMI_x01[HwAlertMask] must be clear.

5.4.3 SBI Error Detection and Recovery

This section describes the various error detection and recovery methods that can be used on the SBI bus. The important item in providing a high reliability SBI connection is the ability to detect when an error occurs and to gracefully recover from that error. When the SBI connections are noisy, messages can become garbled which, in turn, may cause undefined behavior on the SBI bus. The most common noise sources are cross-talk and clock skew. Cross-talk results when the SBI connections are routed too close to other signal carrying lines. Clock skew is usually a result of higher than expected capacitance, between the SBI signals (clock and / or data) and ground, which causes the controller and target devices to disagree on when data should be stable and when it is allowed to be changing.

5.4.3.1 Error Detection

SBI provides several methods of error detection: Transition from address ACK/NACK to SDR controller write data, ninth bit of SDR controller written data as parity checking. For `_x71/ _x72/ _x73` command, timeout is also protected. The ACK/NAK mechanism and parity checking is always active in SBI, but timeout is optional.

5.4.3.1.1 ACK/NAK Mechanism

When performing an SDR Write, the hand off from the target's Address Header ACK to the controller's first Data bit is different in I3C. The end of any Address Header is an ACK or NACK by the one or more addressed targets, using Open Drain on SDA. If `7'h7E` is received, then it is the ACK of all I3C targets on the bus. If a single target address is received, then it is ACK or NACK of the addressed target. When the Address Header results in a NACK, the controller may choose to continue the transaction by generating a Repeated START or relinquish the bus by generating a STOP.

5.4.3.1.2 Ninth Bit of SDR Controller Written Data as Parity

The ninth data bit of each SDR Data Word written by the I3C controller (also referred to as the T-Bit) is a Parity bit, calculated using odd parity. Parity can help in detecting noise-caused errors on the line.

5.4.3.1.3 Bus Timeouts

The I3C protocol itself has no timeout mechanism. For commands such as `_x71/ _x72/ _x73`, hardware will wait for MP to complete the operation by clearing pending interrupt, and then assert `ALERT_L` to controller to trigger following status/data read. If MP has no response within a configured timeout period, then "Status Code" as "Command Timeout" is asserted as well as `ALERT_L` pin, to prevent a device waiting indefinitely on a message that may not be coming.

5.4.3.2 Error Recovery

The simplest form of error recovery is a retry. When the bus controller detects an unexpected NACK, it should abort the current transfer and retry the message sequence. In some cases, however, a message can be so garbled that a simple retry is insufficient. This can occur, if there are multiple devices on the bus and a garbled address byte has caused the wrong target device to be selected. That target device may even continue to transmit during the retry. In those cases, it will be necessary to force a reset of all devices on the SBI bus, before retrying the message transfer.

Unexpected reset of BMC in the middle of SBI transactions can potentially leave the target device controllers in a hung state or RX, TX FIFOs in non-empty state that could skew the data bytes associated with SBI requests. Therefore it is recommended that both SB-RMI and SB-TSI reset recovery flows should be executed by BMC in its boot flow to clear off stale state in the SBI targets.

5.4.3.2.1 SB-RMI Soft Reset Recovery

This section describes the method to recover the SB-RMI peripheral using a soft reset and avoid an expensive CPU socket reset. The soft reset recovery flow should be triggered by the SBI controller such as a BMC when the SB-RMI peripheral gets into a state such as a hang or perpetual NACK for all attempted transactions or read data that is out of sync with read requests (e.g. sequence of reads for registers `<R1, R2, R3>` results in data sequence of `<Indeterminate, R1value, R2value>`).

To trigger a soft reset recovery flow of the SB-RMI, assuming that the SB-TSI peripheral is functional, the following sequence should be followed:

1. SBI controller writes 1 to `SBTSI::ConfigWr[RMI SoftReset]`.
2. [MP1](#) firmware polls on that bit and when set, executes the SB-RMI peripheral recovery flow through a soft reset

of the I3C hardware and clears it upon completion.

3. BMC polls for SBTSt::Config[RMISoftReset] to clear. Once satisfied, BMC can start sending I2C/I3C commands to SB-RMI again.

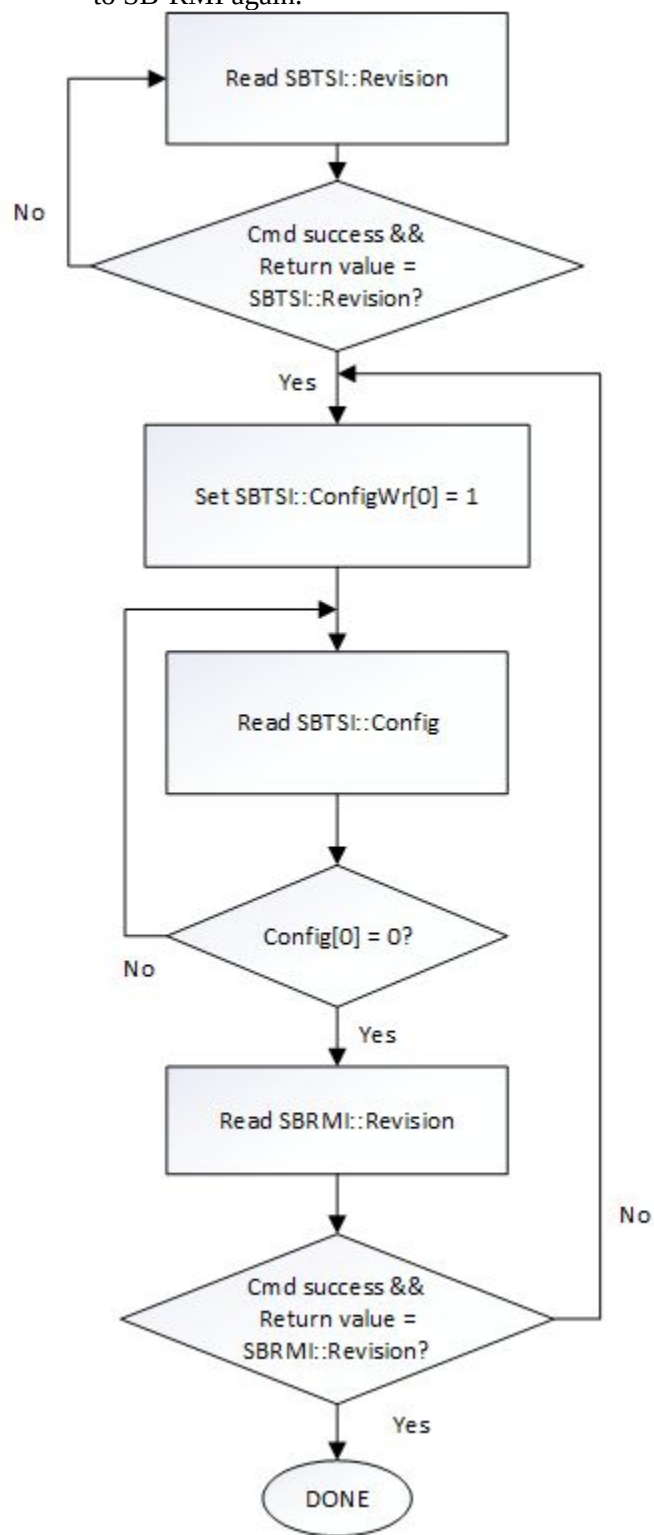


Figure 31: SB-RMI Soft Reset Recovery

5.5 SBI Physical Interface

5.5.1 SBI Static Address

The I3C controller should assign dynamic addresses to any I3C target devices with or without a known static address based on I3C protocol. If it is assigned based on static address, then SETDASA CCC is sent from controller to target, otherwise ENTDAAC CCC is sent from controller to target. Currently static address is supported. The I3C dynamic or static address are really 7 bits: bits[7:1] as the left-justified address, and bit[0] as the Read/Write flag, where 0 indicates a Write and 1 indicates a Read.

5.5.2 SBI Bus Timing

SBI supports 12.5MHz SDR I3C operation. Refer to the timing parameters in the I3C specification.

5.6 SB-RMI Registers

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

SBRMIx00 [Revision] (SBRMI::Revision)	
Read-only. Reset: 21h.	
Bits	Description
7:0	Revision: SB-RMI revision. Read-only. Reset: 21h. This field specifies the APML -I3C specification revision that the product is compliant to. 0x21 = 2.1x Revision.

SBRMIx01 [Control] (SBRMI::Control)	
Read-write. Reset: 01h.	
Bits	Description
7	HwAlertMask: hardware alert mask. Read-write. Reset: 0. 0=ALERT_L signaling is enabled when SBRMI_x02[HwAlertSts] is set. 1=ALERT_L signaling is disabled when SBRMI_x02[HwAlertSts] is set.
6:5	Reserved.
4	SwAlertMask: software alert mask. Read-write. Reset: 0. 0=ALERT_L signaling is enabled when SBRMI_x02[SwAlertSts] is set. 1=ALERT_L signaling is disabled when SBRMI_x02[SwAlertSts] is set.
3	BlkRWEn: block read/write enable. Read-write. Reset: 0. 0=I3C accesses can only use the byte read/write protocol. 1=I3C accesses can only use the block read/write protocol. Description: Controls block read/write access to register ranges SBRMI_x[5F:10] and SBRMI_x[DF:80]. NOTE: For register ranges SBRMI_x[0F:00], it only support Single Write independent of the state of the BlkRWEn control bit, but its read still depends on BlkRWEn control bit.
2	TimeoutDis: SB-RMI timeout disable. Read-write. Reset: 0. 1=I3C defined timeouts are disabled. It is only used for SB-RMI to process _x71, _x72, _x73 commands to wait for Firmware to complete the commands by clearing pending interrupt.
1	TSISoftReset: trigger TSI soft reset flow. Read-write. Reset: 0. 1=SBI controller triggers MP1 firmware to soft reset SB-TSI peripheral. 0=soft reset of SB-TSI I3C hardware completed and BMC may proceed communication with the SB-TSI peripheral. See 6.2.7 [SB-TSI Error Detection and Recovery] for details.
0	AlertMask: SB-RMI alert mask. Read-write. Reset: 1. Read-write. 1=ALERT_L signaling disabled. 0=ALERT_L is asserted if any unmasked event is present in the SBRMI::AlertStatus registers SBRMI_x01[F:0] and SBRMI_x05[F:0], or if SBRMI_x02[SwAlertSts] = 1 and SBRMI_x01[SwAlertMask] = 0, or if SBRMI_x02[HwAlertSts] = 1 and SBRMI_x01[HwAlertMask] = 0.

SBRMIx02 [Status] (SBRMI::Status)

Reset: 00h.

Bits	Description
7	HwAlertSts: SB-RMI hardware alert status. Read-write, Volatile . Reset: 0. Write-one-to-clear from the I3C interface; read-write from the processor. Set by hardware as a result of a _x71/ _x72/ _x73 command completion and ready to assert ALERT_L for this completion.
6	MP0AlertSts: Alert from MP0 for MP0 outbound message. Read-write, Volatile . Reset: 0. Write-one-to-clear from the I3C interface; set by hardware when MP0 firmware writes to an internal register after setting up Mp0OutBndMsg registers. It indicates availability of MP0 data to BMC. This bit was formerly named PSPAlertStatus.
5:2	Reserved.
1	SwAlertSts: SB-RMI software alert status. Read-write, Volatile . Reset: 0. Write-one-to-clear from the I3C interface; read-write from the processor. Set by firmware to indicate the completion of a mailbox operation or due to an asynchronous event, such as those reported by the SBRMI::RasStatus register.
0	AlertSts: SB-RMI alert status. Read-only, Volatile . Reset: 0. This bit is set to 1 by hardware when an alert event is present in at least one bit of SBRMI_x01[F:0], SBRMI_x05[F:0] or SBRMI_x2[7:2][F:0]. It is cleared by hardware when all those registers are clear.

SBRMIx03 [Read Size] (SBRMI::ReadSize)

Read-write. Reset: 01h.

This register specifies the number of bytes to return when using the block read protocol.

Bits	Description								
7:6	Reserved.								
5:0	RdSize: read size. Read-write. Reset: 01h. Specifies the number of bytes to return when using the block read protocol. ValidValues:								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>20h-01h</td><td><Value> bytes.</td></tr> <tr> <td>3Fh-21h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Reserved.	20h-01h	<Value> bytes.	3Fh-21h	Reserved.
Value	Description								
00h	Reserved.								
20h-01h	<Value> bytes.								
3Fh-21h	Reserved.								

SBRMIx[04...DF] [Thread Enable Status] (SBRMI::ThreadEnableStatus)

Read-only.

_inst[31:8,47:32,7:0]; SBRMIx[D[F:8],9[8:1],4[A:3],10[F:0],0[D:8],05,04]

Bits	Description
7:0	threadEnStat: thread Enable Status. Read-only. Description: 1=thread is enabled. See: SBRMI::ThreadEnableStatus[threadEnStat] Description.

SBRMI::ThreadEnableStatus[threadEnStat] Description

Offset[7:0]	inst	Description
04h	0	threads [7:0]
05h	1	threads [15:8]
08h	2	threads [23:16]
09h	3	threads [31:24]
0Ah	4	threads [39:32]
0Bh	5	threads [47:40]
0Ch	6	threads [55:48]

0Dh	7	threads [63:56]
43h	8	threads [71:64]
44h	9	threads [79:72]
45h	10	threads [87:80]
46h	11	threads [95:88]
47h	12	threads [103:96]
48h	13	threads [111:104]
49h	14	threads [119:112]
4Ah	15	threads [127:120]
91h	16	threads [135:128]
92h	17	threads [143:136]
93h	18	threads [151:144]
94h	19	threads [159:152]
95h	20	threads [167:160]
96h	21	threads [175:168]
97h	22	threads [183:176]
98h	23	threads [191:184]
D8h	24	threads [199:192]
D9h	25	threads [207:200]
DAh	26	threads [215:208]
DBh	27	threads [223:216]
DCh	28	threads [231:224]
DDh	29	threads [239:232]
DEh	30	threads [247:240]
DFh	31	threads [255:248]
100h	32	threads [263:256]
101h	33	threads [271:264]
102h	34	threads [279:272]
103h	35	threads [287:280]
104h	36	threads [295:288]
105h	37	threads [303:296]
106h	38	threads [311:304]
107h	39	threads [319:312]
108h	40	threads [327:320]
109h	41	threads [335:328]
10Ah	42	threads [343:336]
10Bh	43	threads [351:344]
10Ch	44	threads [359:352]
10Dh	45	threads [367:360]
10Eh	46	threads [375:368]
10Fh	47	threads [383:376]

SBRMIx[10...5F] [Alert Status] (SBRMI::AlertStatus)

Read,Write-1-to-clear, Volatile.

_inst[31:16,47:32,15:0]; SBRMIx[5[F:0],22[F:0],1[F:0]]

Bits	Description
7:0	MceStat: MCE status. Read, Write-1-to-clear , Volatile . Description: Bit vector for threads. 1=MCE occurred for thread. Set by hardware. See: SBRMI::AlertStatus[MceStat] Description.

SBRMI::AlertStatus[MceStat] Description

Offset[7:0]	inst	Description
10h	0	threads [176,160,144,128,48,32,16,0]
11h	1	threads [177,161,145,129,49,33,17,1]
12h	2	threads [178,162,146,130,50,34,18,2]
13h	3	threads [179,163,147,131,51,35,19,3]
14h	4	threads [180,164,148,132,52,36,20,4]
15h	5	threads [181,165,149,133,53,37,21,5]
16h	6	threads [182,166,150,134,54,38,22,6]
17h	7	threads [183,167,151,135,55,39,23,7]
18h	8	threads [184,168,152,136,56,40,24,8]
19h	9	threads [185,169,153,137,57,41,25,9]
1Ah	10	threads [186,170,154,138,58,42,26,10]
1Bh	11	threads [187,171,155,139,59,43,27,11]
1Ch	12	threads [188,172,156,140,60,44,28,12]
1Dh	13	threads [189,173,157,141,61,45,29,13]
1Eh	14	threads [190,174,158,142,62,46,30,14]
1Fh	15	threads [191,175,159,143,63,47,31,15]
50h	16	threads [240,224,208,192,112,96,80,64]
51h	17	threads [241,225,209,193,113,97,81,65]
52h	18	threads [242,226,210,194,114,98,82,66]
53h	19	threads [243,227,211,195,115,99,83,67]
54h	20	threads [244,228,212,196,116,100,84,68]
55h	21	threads [245,229,213,197,117,101,85,69]
56h	22	threads [246,230,214,198,118,102,86,70]
57h	23	threads [247,231,215,199,119,103,87,71]
58h	24	threads [248,232,216,200,120,104,88,72]
59h	25	threads [249,233,217,201,121,105,89,73]
5Ah	26	threads [250,234,218,202,122,106,90,74]
5Bh	27	threads [251,235,219,203,123,107,91,75]
5Ch	28	threads [252,236,220,204,124,108,92,76]
5Dh	29	threads [253,237,221,205,125,109,93,77]
5Eh	30	threads [254,238,222,206,126,110,94,78]
5Fh	31	threads [255,239,223,207,127,111,95,79]
220h	32	threads [368,352,336,320,304,288,272,256]
221h	33	threads [369,353,337,321,305,289,273,257]
222h	34	threads [370,354,338,322,306,290,274,258]
223h	35	threads [371,355,339,323,307,291,275,259]
224h	36	threads [372,356,340,324,308,292,276,260]
225h	37	threads [373,357,341,325,309,293,277,261]
226h	38	threads [374,358,342,326,310,294,278,262]
227h	39	threads [375,359,343,327,311,295,279,263]
228h	40	threads [376,360,344,328,312,296,280,264]
229h	41	threads [377,361,345,329,313,297,281,265]
22Ah	42	threads [378,362,346,330,314,298,282,266]
22Bh	43	threads [379,363,347,331,315,299,283,267]
22Ch	44	threads [380,364,348,332,316,300,284,268]
22Dh	45	threads [381,365,349,333,317,301,285,269]

22Eh	46	threads [382,366,350,334,318,302,286,270]
22Fh	47	threads [383,367,351,335,319,303,287,271]

SBRMIx1C0...SBRMIxCF [Alert Mask] (SBRMI::AlertMask)

Read-write.

_inst[31:0,47:32]; SBRMIx[C[F:0],2[F:0],1C[F:0]]

Bits	Description
7:0	MceAlertMsk: MCE alert mask. Read-write. Description: Bit vector for threads. 1=Alert signaling disabled for corresponding SBRMI::AlertStatus[MceStat] for thread. See: SBRMI::AlertMask[MceAlertMsk] Description.

SBRMI::AlertMask[MceAlertMsk] Description

Offset[7:0]	inst	Description
20h	0	threads [176,160,144,128,48,32,16,0]
21h	1	threads [177,161,145,129,49,33,17,1]
22h	2	threads [178,162,146,130,50,34,18,2]
23h	3	threads [179,163,147,131,51,35,19,3]
24h	4	threads [180,164,148,132,52,36,20,4]
25h	5	threads [181,165,149,133,53,37,21,5]
26h	6	threads [182,166,150,134,54,38,22,6]
27h	7	threads [183,167,151,135,55,39,23,7]
28h	8	threads [184,168,152,136,56,40,24,8]
29h	9	threads [185,169,153,137,57,41,25,9]
2Ah	10	threads [186,170,154,138,58,42,26,10]
2Bh	11	threads [187,171,155,139,59,43,27,11]
2Ch	12	threads [188,172,156,140,60,44,28,12]
2Dh	13	threads [189,173,157,141,61,45,29,13]
2Eh	14	threads [190,174,158,142,62,46,30,14]
2Fh	15	threads [191,175,159,143,63,47,31,15]
C0h	16	threads [240,224,208,192,112,96,80,64]
C1h	17	threads [241,225,209,193,113,97,81,65]
C2h	18	threads [242,226,210,194,114,98,82,66]
C3h	19	threads [243,227,211,195,115,99,83,67]
C4h	20	threads [244,228,212,196,116,100,84,68]
C5h	21	threads [245,229,213,197,117,101,85,69]
C6h	22	threads [246,230,214,198,118,102,86,70]
C7h	23	threads [247,231,215,199,119,103,87,71]
C8h	24	threads [248,232,216,200,120,104,88,72]
C9h	25	threads [249,233,217,201,121,105,89,73]
CAh	26	threads [250,234,218,202,122,106,90,74]
CBh	27	threads [251,235,219,203,123,107,91,75]
CCh	28	threads [252,236,220,204,124,108,92,76]
CDh	29	threads [253,237,221,205,125,109,93,77]
CEh	30	threads [254,238,222,206,126,110,94,78]
CFh	31	threads [255,239,223,207,127,111,95,79]
1C0h	32	threads [368,352,336,320,304,288,272,256]
1C1h	33	threads [369,353,337,321,305,289,273,257]

1C2h	34	threads [370,354,338,322,306,290,274,258]
1C3h	35	threads [371,355,339,323,307,291,275,259]
1C4h	36	threads [372,356,340,324,308,292,276,260]
1C5h	37	threads [373,357,341,325,309,293,277,261]
1C6h	38	threads [374,358,342,326,310,294,278,262]
1C7h	39	threads [375,359,343,327,311,295,279,263]
1C8h	40	threads [376,360,344,328,312,296,280,264]
1C9h	41	threads [377,361,345,329,313,297,281,265]
1CAh	42	threads [378,362,346,330,314,298,282,266]
1CBh	43	threads [379,363,347,331,315,299,283,267]
1CCh	44	threads [380,364,348,332,316,300,284,268]
1CDh	45	threads [381,365,349,333,317,301,285,269]
1CEh	46	threads [382,366,350,334,318,302,286,270]
1CFh	47	threads [383,367,351,335,319,303,287,271]

SBRMIx3[0...7] [Out-Bound Message] (SBRMI::OutBndMsg)

Read-write. Reset: 00h.

_inst[7:0]; SBRMIx3[7:0]

Bits	Description
7:0	OutBndMsg: outbound message data. Read-write. Reset: 00h.
	Description: Read-write from the processor; read-only from the I3C interface.
	Usage convention is:
	<ul style="list-style-type: none">SBRMI::OutBndMsg_inst0 is command copied by firmware from SBRMI::InBndMsg_inst0.SBRMI::OutBndMsg_inst[4:1] are 32 bit data.SBRMI::OutBndMsg_inst[5] reserved.SBRMI::OutBndMsg_inst[6] is count of RMI soft reset flows executed by MP1 firmware since CPU resetSBRMI::OutBndMsg_inst[7] contains Mailbox Error Code, per Table 145 [SB-RMI Soft Mailbox Error Code].

SBRMIx3[8...F] [In-Bound Message] (SBRMI::InBndMsg)

Read-write. Reset: 00h.

_inst[7:0]; SBRMIx3[F:8]

Bits	Description																											
7:0	InBndMsg: inbound message data. Read-write. Reset: 00h.																											
	Description: Read-write from the I3C interface; read-only from the processor. These registers are used for communicating 32 bit messages from BMC to firmware.																											
	Usage convention is:																											
	<ul style="list-style-type: none">• SBRMI::InBndMsg_inst0 is command• SBRMI::InBndMsg_inst[4:1] are 32 bit data• SBRMI::InBndMsg_inst[6:5] are reserved.• SBRMI::InBndMsg_inst7: [7] Must be 1'b1 to send message to firmware																											
	<table><tr><th>Offset[7:0]</th><th>inst</th><th>Description</th></tr><tr><td>38h</td><td>0</td><td>Inbound message 0</td></tr><tr><td>39h</td><td>1</td><td>Inbound message 1</td></tr><tr><td>3Ah</td><td>2</td><td>Inbound message 2</td></tr><tr><td>3Bh</td><td>3</td><td>Inbound message 3</td></tr><tr><td>3Ch</td><td>4</td><td>Inbound message 4</td></tr><tr><td>3Dh</td><td>5</td><td>Inbound message 5</td></tr><tr><td>3Eh</td><td>6</td><td>Inbound message 6</td></tr><tr><td>3Fh</td><td>7</td><td>Inbound message 7</td></tr></table>	Offset[7:0]	inst	Description	38h	0	Inbound message 0	39h	1	Inbound message 1	3Ah	2	Inbound message 2	3Bh	3	Inbound message 3	3Ch	4	Inbound message 4	3Dh	5	Inbound message 5	3Eh	6	Inbound message 6	3Fh	7	Inbound message 7
	Offset[7:0]	inst	Description																									
	38h	0	Inbound message 0																									
	39h	1	Inbound message 1																									
	3Ah	2	Inbound message 2																									
	3Bh	3	Inbound message 3																									
3Ch	4	Inbound message 4																										
3Dh	5	Inbound message 5																										
3Eh	6	Inbound message 6																										
3Fh	7	Inbound message 7																										

SBRMIx40 [Software Interrupt] (SBRMI::SoftwareInterrupt)

Read,Write-1-only. Reset: 00h.

This register is used by the I3C controller to generate an interrupt to the processor to indicate that a message is available.

Bits	Description
7:1	Reserved.
0	SwInt: firmware interrupt. Read, Write-1-only . Reset: 0. Read, write-1-only from the I3C interface; read, write-1-to-clear from firmware. 1=Indicates a firmware mailbox service request.

SBRMIx4B [Thread128 CS register] (SBRMI::Thread128CS)

Read-write. Reset: 00h.

This register is used to extend the threads from 128 to 256 for _x71, _x72, _x73 commands. Setting as 1'b1 to select high 128 threads while setting as 1'b0 to select low 128 threads. It is set by I3C interface.

Bits	Description
7:1	Reserved.
0	thread128cs: high or low thread 128 selection register. Read-write. Reset: 0. Read-Write from the I3C interface; specifies the high or low thread 128 indicated in <u>_x71</u> , <u>_x72</u> , <u>_x73</u> commands. Setting as 1'b0 to indicate the low 128 threads; setting as 1'b1 to indicate the high 128 threads.

SBRMIx4C [RAS Status register] (SBRMI::RasStatus)

Read-write. Reset: 00h.

This register is used to indicate the [APML](#) ALERT source. It is set by processor, and write-1-to-clear by I3C interface.

Bits	Description
7:6	Reserved.
5	pcie_err: PCIE error. Read-write. Reset: 0. Read-Write from the I3C interface; specifies the APML ALERT source is due to a PCIE error or PCIe error counter threshold overflow. It is set by software, and write-1-to-clear by I3C interface.
4	dram_cecc_err: DRAM Correctable ECC error counter overflow. Read-write. Reset: 0. Read-Write from the I3C interface; specifies the APML ALERT source is due to a DRAM Correctable ECC error counter threshold overflow. It is set by software, and write-1-to-clear by I3C interface.
3	mca_err: MCA error. Read-write. Reset: 0. Read-Write from the I3C interface; specifies the APML ALERT source is due to a non-fatal MCA error or MCA error counter threshold overflow. It is set by software, and write-1-to-clear by I3C interface.
2	reset_ctrl_err: reset control error. Read-write. Reset: 0. Read-Write from the I3C interface; specifies the APML ALERT source is due to a reset controller error; warm-reset requests may not complete. It is set by software, and write-1-to-clear by I3C interface.
1	fch_err: FCH error. Read-write. Reset: 0. Read-Write from the I3C interface; specifies the APML ALERT source is due to a FCH error; warm- or cold-reset requests may not complete. It is set by software, and write-1-to-clear by I3C interface.
0	fatal_err: fatal error. Read-write. Reset: 0. Read-Write from the I3C interface; specifies the APML ALERT source is due to a fatal error. It is set by software, and write-1-to-clear by I3C interface.

SBRMIx4E [Thread Number Low] (SBRMI::ThreadNumberLow)

Read-write. Reset: 00h.

This register indicates the low part of maximum number of threads present, starting from 1.

Bits	Description
7:0	threadNumlow: thread number low. Read-write. Reset: 00h. Read-only from the I3C interface; specifies the low part of the maximum number of threads present. Firmware loads the initial value based on the maximum number of threads available after any fused off or soft-down-coring is complete. 'd64 means max of 64 threads, 'd128 means max of 128 threads.

SBRMIx4F [Thread Number High] (SBRMI::ThreadNumberHigh)

Read-write. Reset: 00h.

This register indicates the high part of maximum number of threads present. It is only valid when 256 threads needs to be supported.

Bits	Description
7:0	threadNumhigh: thread number high. Read-write. Reset: 00h. Read-only from the I3C interface; specifies the high part of the maximum number of threads present. This register is only needed when thread number is equal or larger than 256. Firmware loads the initial value based on the maximum number of threads available after any fused off or soft-down-coring is complete.

SBRMIx8[0...7] [ASP Out-Bound Message] (SBRMI::MP0OutBndMsg)

Read-write. Reset: 00h.

_inst[7:0]; SBRMIx8[7:0]

Bits	Description	
7:0	MP0OutBndMsg: outbound message data. Read-write. Reset: 00h. Description: Read-write from the processor; read-only from the I3C interface. These registers are used for sending messages from ASP firmware running on the ASP to the I3C controller. ASP boot status is dynamically written to this register during the boot process.	
	Offset[7:0]	Description
	80h	ASP Outbound message 0
	81h	ASP Outbound message 1
	82h	ASP Outbound message 2
	83h	ASP Outbound message 3
	84h	ASP Outbound message 4
	85h	ASP Outbound message 5
	86h	ASP Outbound message 6
	87h	ASP Outbound message 7

SBRMIx90 [ASP Alert Mask] (SBRMI::MP0AlertMask)

Read-write. Reset: 00h.

Bits	Description
7:2	Reserved.
1	Mp0AlertMask. Read-write. Reset: 0. 0=ALERT_L signaling is enabled when MP0AlertSts is set. 1=ALERT_L signaling is disabled when MP0AlertSts is set.
0	Reserved.

6 SB Temperature Sensor Interface (SB-TSI)

6.1 Overview

The SBI temperature sensor interface (SB-TSI) integrates the functionality of a typical 8-pin remote temperature sensor (RTS) shown in Figure 32 [RTS Thermal Management Example], into the CPU socket. The goal is to resemble a typical RTS so that KBC or BMC firmware requires minimal changes for future AMD products as shown in Figure 33 [SB-TSI Thermal Management Example]. SB-TSI supports the I3C protocols that typical RTS supports.

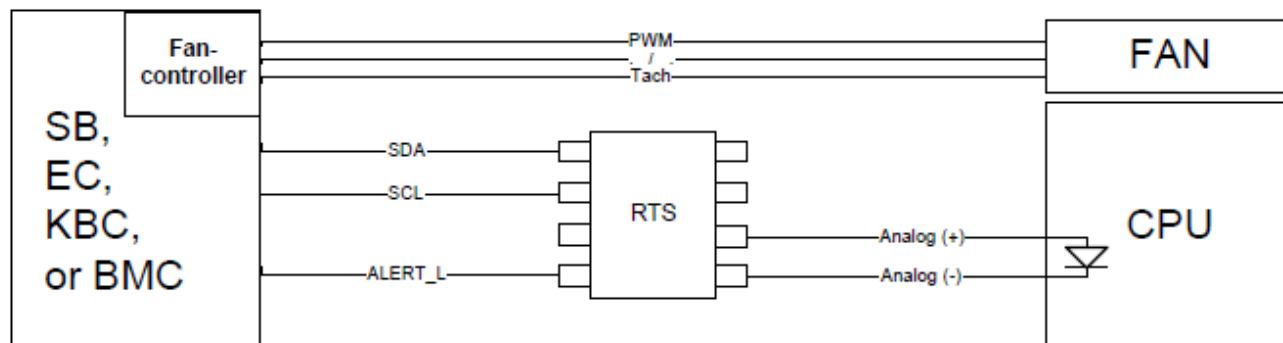


Figure 32: RTS Thermal Management Example

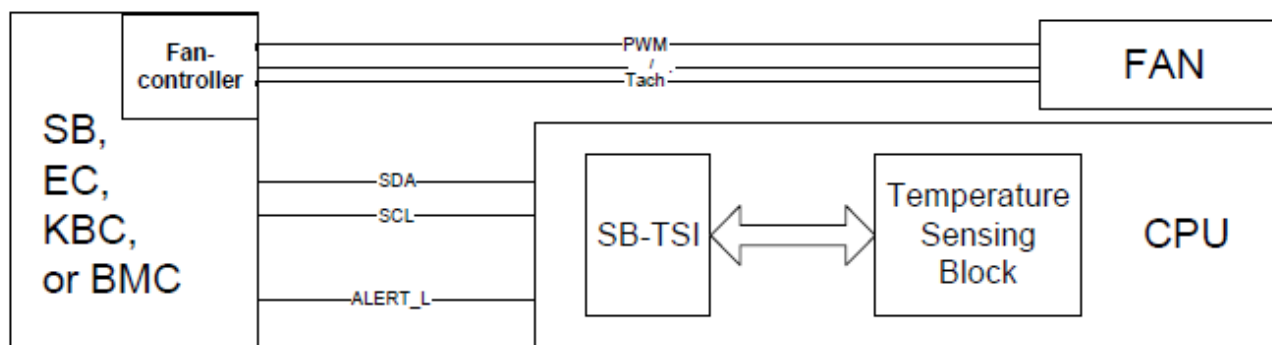


Figure 33: SB-TSI Thermal Management Example

Refer to the following external sources for additional information.

- MIPI I3C[®] specification Version 1.0, see [docI3C](#).

6.1.1 Definitions

Table 150: SB-TSI Definitions

Term	Description
BMC	Base management controller.
TCC	Temperature calculation circuit.
Tctl	Processor temperature control value.
TSM	Temperature sensor macro.
SB-TSI	Sideband Internal Temperature Sensor Interface. See APML .

6.1.2 Tctl

Tctl is a processor temperature control value used for processor thermal management. In mission mode it reports the maximum temperature amongst multiple on-die temperature sensors located across all the constituent chiplets in a socket. Tctl is accessible externally on the platform through SBTSI interface. Tctl does not represent the real measured temperature but instead an offset adjusted value such that a value of 100 represents the max sustainable operating die temperature depending on the processor family and model. This enables the abstraction of the real temperature to a standard scale where fan speed profile can be specified against standardized Tctl values.

6.2 SB-TSI Protocol

The SB-TSI largely follows I3C protocol with the following characteristics:

- The processor does not implement I3C controller functionality. It only functions as a target.
- Max of 12.5Mhz SDR mode is supported.
- The SBI mainly uses the I3C private Write and Read transfers to support Single Write and Read.
- As a target device, it can only drive SDA, but not SCL to controller.
- Dynamic address assigned based on Static address can be supported for I3C target.
- Packet Error Correction (PEC) is not supported in I2C or I3C modes by target.

In order to allow [MP1](#) firmware to perform initialization of certain SB-RMI and SB-TSI registers, BMC should hold off conducting any transaction to either of these targets for at least 2 secs after cold or warm reset of the CPU.

6.2.1 I3C Private Write and Read Protocol

SB-TSI uses a standard I3C protocol to provide Private Write and Private Read Transfers. Figure below shows the I3C spec defined Private Write and Private Read transmission protocol. Each byte in the protocol is sent with the most significant bit first (bit[7]). The ninth data bit of each SDR Data Word written by the I3C controller (also referred to as the T-Bit) is a Parity bit, calculated using odd parity. The ninth (T) Data bit of each SDR Data Word returned by the I3C target indicates if it is the End-of-Data, it returns the ninth bit as 0 (SDA Low) to end the Message.

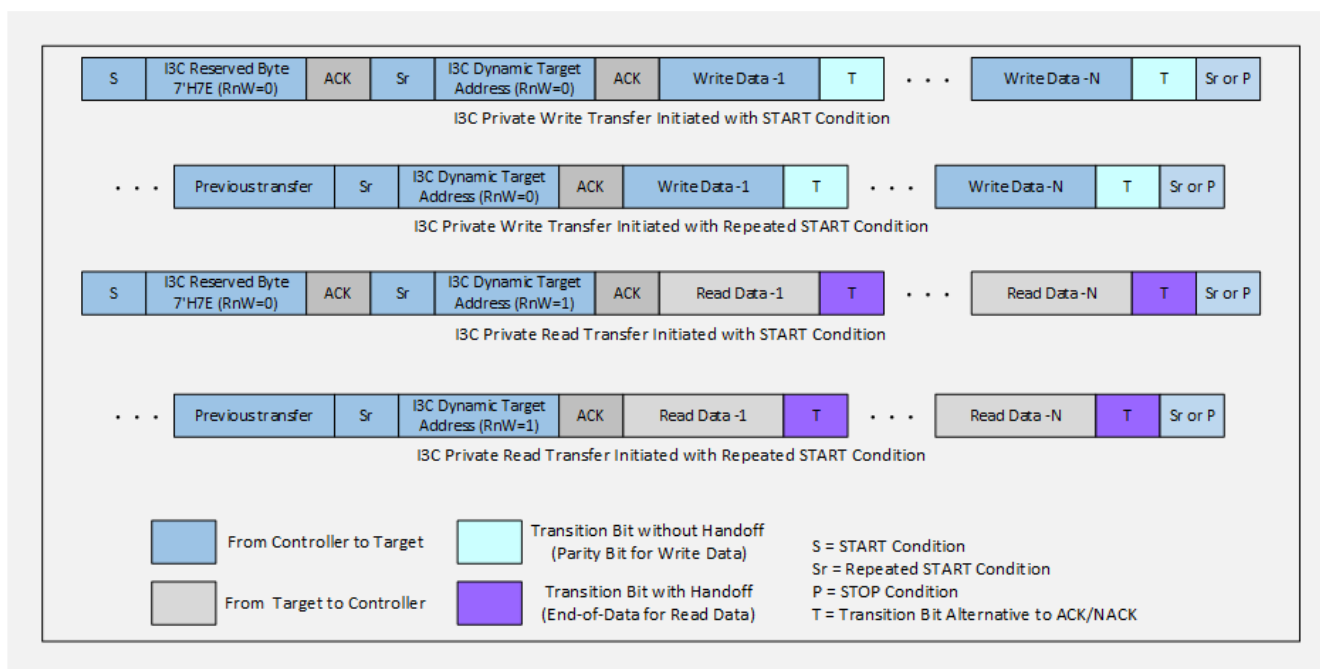


Figure 34: I3C Transmission Protocol for Private Write and Read

6.2.2 SB-TSI Read/Write Byte Protocol

An I3C controller can Read or Write SB-TSI internal registers by issuing a Private Read or a Write command with register address, written data filled in the transmission. SB-TSI only supports Single Read or Write. Figure below shows the transmission protocol for Single Write and Read based on I3C defined Private Write and Read protocol. For Single Write, one byte of "Command Code" is provided for register offset and one byte of "Write Data" is provided for written data. For Single Read, one byte of "Command Code" is provided for register offset by using a Private Write first, and followed by a Private Read to read one byte of data using Repeated START or START condition.

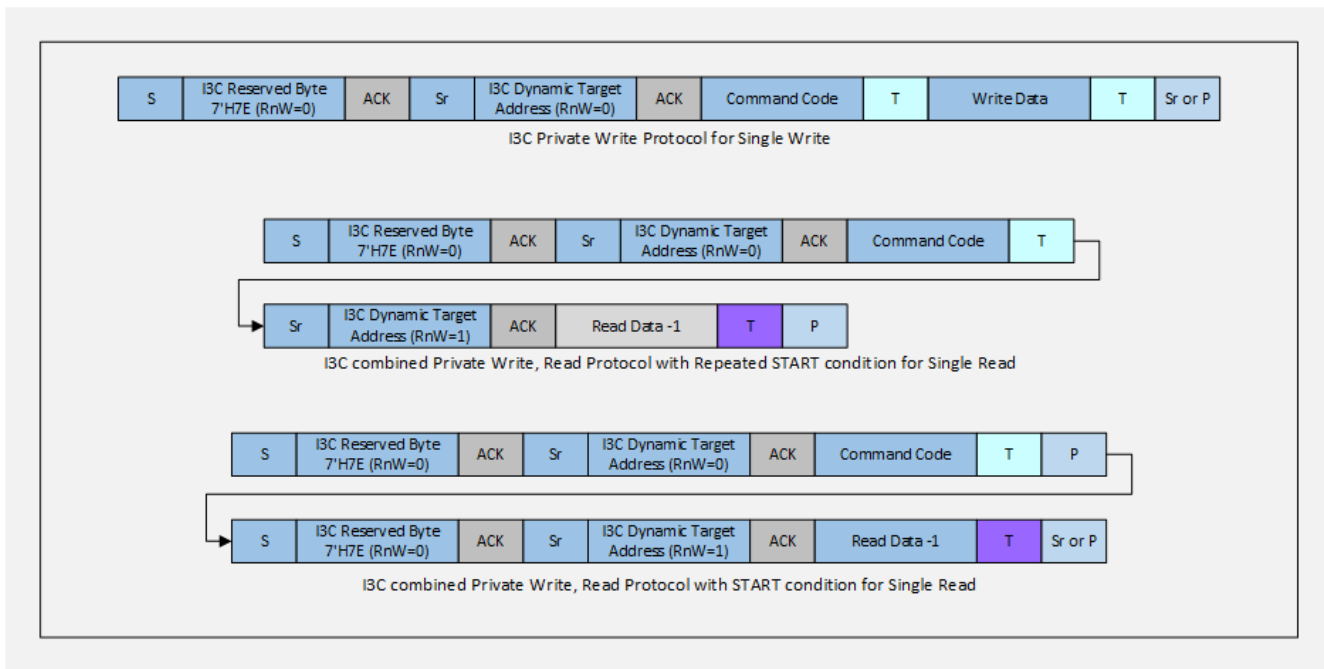


Figure 35: SB-TSI Transmission Protocol for Single Write and Read

6.2.3 Alert Behavior

The ALERT_L pin is asserted if (SBTSI::Status[TempHighAlert] || SBTSI::Status[TempLowAlert]) && ~SBTSI::Config[AlertMask] as shown in Figure 3. The following registers also affect temperature alert behavior.

- SBTSI::UpdateRate[UpRate]: Specifies rate at which temperature thresholds are checked.
- {SBTSI::HiTempInt[HiTempInt], SBTSI::HiTempDec[HiTempDec]}: Sets high temperature threshold.
- {SBTSI::LoTempInt[LoTempInt], SBTSI::LoTempDec[LoTempDec]}: Sets low temperature threshold.
- SBTSI::AlertThreshold[AlertThr]: Specifies number of consecutive temperature samples to assert an alert.
- SBTSI::AlertConfig[AlertCompEn]: Specifies ALERT_L pin to be in latched or comparator mode.

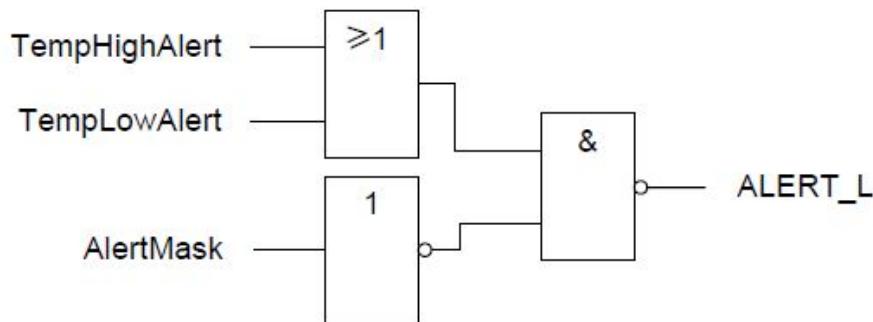


Figure 36: Alert Assertion Diagram

6.2.4 Atomic Read Mechanism

Atomic reading procedures are required to ensure the two required Reads (integer and decimal) parts of CPU temperature always originate from one temperature value. SB-TSI offers functions to maintain atomicity between the temperature

integer and decimal bytes.

SBTSI::Config[ReadOrder] specifies the order for reading the following registers:

- Integer and decimal part of SBTSI::CpuTempInt and SBTSI::CpuTempDec

If SBTSI::Config[ReadOrder] is 0, then a read of the integer part triggers a latch of the decimal part until the next read of the integer part. This latching syncs the decimal part with the integer part. The integer part is updated continuously.

If SBTSI::Config[ReadOrder] is 1, the read order to ensure atomicity is reversed, i.e., the read of the decimal part triggers a latch of the integer part until the next read of the decimal part. This latching syncs the integer part with the decimal part. The decimal part is updated continuously.

If it is not possible to specify a read order as described above, the Run/Stop bit (SBTSI::Config[RunStop]) may be used to provide atomicity of reading the CPU temperature. If this bit is 0, the CPU temperature registers are updated continuously. If this bit is 1, the CPU temperature registers are no longer updated and any subsequent read requests will deliver the last latched value.

- Set SBTSI::Config[RunStop].
- Read the integer (SBTSI::CpuTempInt) or the decimal (SBTSI::CpuTempDec) part of the CPU temperature.
- Read the remaining part of the CPU temperature.
- Clear SBTSI::Config[RunStop].

6.2.5 SB-TSI Temperature and Threshold Encodings

SB-TSI CPU temperature readings and limit registers encode the temperature in increments of 0.125 from 0 to 255.875. The high byte represents the integer portion of the temperature from 0 to 255. One increment in the high byte is equivalent to a step of one. The upper three bits of the low byte represent the decimal portion of the temperature. One increment of these bits is equivalent to a step of 0.125.

Table 151: SB-TSI CPU Temperature and Threshold Encoding Examples

Temperature	Temperature High Byte SBTSI::CpuTempInt[CpuTempInt] SBTSI::HiTempInt[HiTempInt] SBTSI::LoTempInt[LoTempInt]	Temperature Low Byte SBTSI::CpuTempDec[CpuTempDec] SBTSI::HiTempDec[HiTempDec] SBTSI::LoTempDec[LoTempDec]
0.000 °C	0000_0000b	0000_0000b
1.000 °C	0000_0001b	0000_0000b
25.125 °C	0001_1001b	0010_0000b
50.875 °C	0011_0010b	1110_0000b
90.000 °C	0101_1010b	0000_0000b

6.2.6 SB-TSI Temperature Offset Encoding

By default, SBTSI::CpuTempInt and SBTSI::CpuTempDec provide Tctl from the processor. The temperature offset registers allow the system to adjust the SB-TSI temperature from Tctl.

The SB-TSI temperature offset registers use a different encoding in order to provide negative temperature values. SBTSI::CpuTempOffInt[CpuTempOffInt] and SBTSI::CpuTempOffDec[CpuTempOffDec] form an 11-bit, 2's complement value representing the temperature offset. The high byte encodes the integer portion of the temperature and the upper three bits of the low byte represent the fractional portion of the temperature offset. One increment of these bits is equivalent to a step of 0.125 °C. After reset the offset is always set to 0 °C. Software needs to adjust the offset to the

appropriate level.

Table 152: SB-TSI Temperature Offset Encoding Examples

Temperature	Temperature High Byte SBTSI::CpuTempOffInt[CpuTempOffInt]	Temperature Low Byte SBTSI::CpuTempOffDec[CpuTempOffDec]
-10.375 °C	1111_0101b	1010_0000b
-0.250 °C	1111_1111b	1100_0000b
0.000 °C	0000_0000b	0000_0000b
0.875 °C	0000_0000b	1110_0000b
10.000 °C	0000_1010b	0000_0000b

6.2.7 SB-TSI Error Detection and Recovery

SB-TSI is one of the two peripheral SBI devices on the SBI bus. The section 5.4.3 [SBI Error Detection and Recovery] describes the overall error detection and recovery methods available for SBI bus. This section describes the method to recover the SB-TSI peripheral using a soft reset and avoid an expensive CPU socket reset. The soft reset recovery flow should be triggered by the SBI controller such as a BMC when the SB-TSI peripheral gets into a state such as a hang or perpetual NACK for all attempted transactions or read data that is out of sync with read requests (e.g. sequence of reads for registers <R1, R2, R3> results in data sequence of <Indeterminate, R1value, R2value>).

To trigger a soft reset recovery flow of the SB-TSI, assuming that the SB-RMI peripheral is functional, the following sequence should be followed:

1. SBI controller writes 1 to SBRMI::Control[TSISoftReset].
2. [MP1](#) firmware polls on that bit and when set, executes the SB-TSI peripheral recovery flow through a soft reset of the I3C hardware and clears it upon completion.
3. BMC polls for TSISoftReset to clear. Once satisfied, BMC can start sending I2C/I3C commands to TSI again.

Unexpected reset of BMC in the middle of SBI transactions can potentially leave the target device in a hung state or RX, TX FIFOs in non-empty state that could skew the data bytes associated with SBI requests. Therefore it is recommended that both SB-RMI and SB-TSI reset recovery flows should be executed by BMC in its boot flow to clear off stale state in the SBI targets.

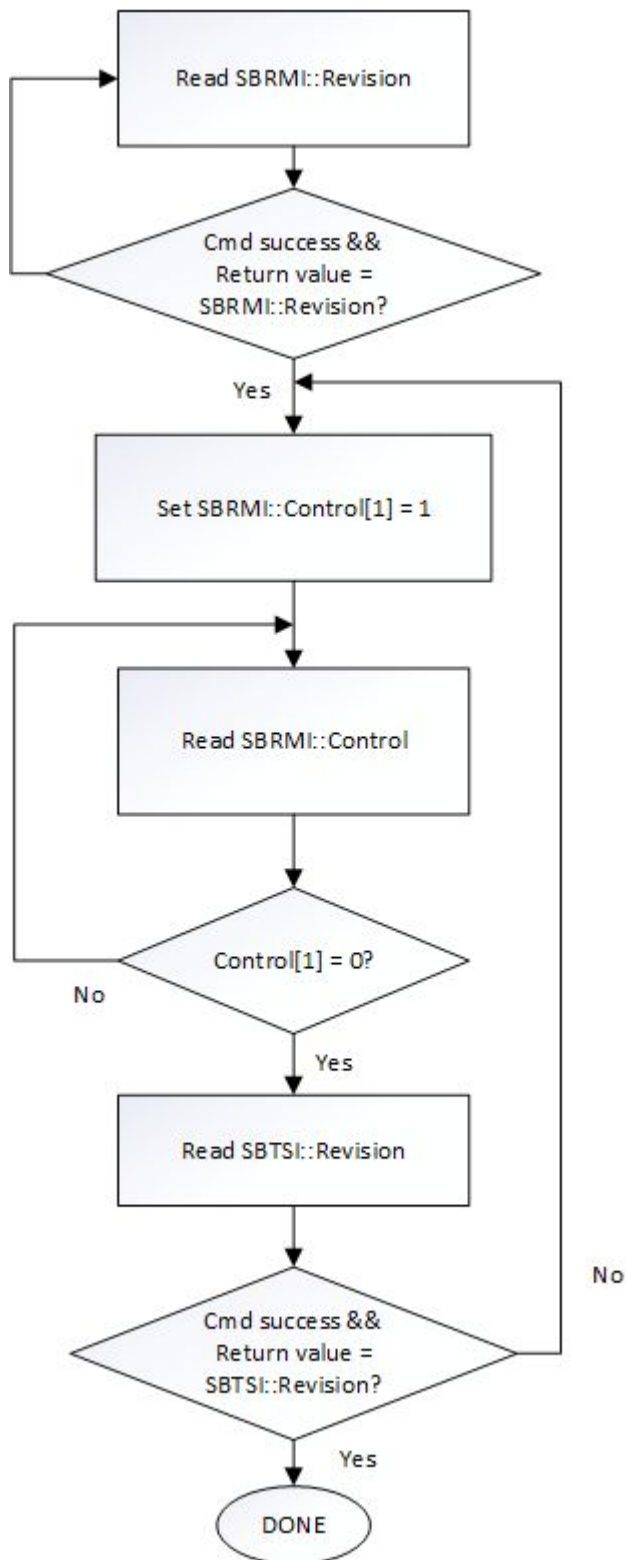


Figure 37: SB-TSI Soft Reset Recovery

6.3 SB-TSI Physical Interface

This chapter describes the physical interface of the SB-TSI.

6.3.1 SB-TSI I3C Static Address

The I3C controller should assign dynamic addresses to any I3C target devices with or without a known static address based on I3C protocol. If it is assigned based on static address, then SETDASA CCC is sent from controller to target, otherwise ENTDAAC CCC is sent from controller to target. Currently static address is supported.

The I3C dynamic or static address are really 7 bits, and left with the 1 bit of Read/Write indication as a WRITE (0) making bit[0].

Table 153: SB-TSI Static Address Encodings

Socket ID	SB-TSI Address
0b	4Ch for 7 bit.
1b	48h for 7 bit.

6.3.2 SB-TSI Bus Timing

SB-TSI supports 12.5MHz SDR mode according to the I3C-bus Specification and User Manual.

6.3.3 SB-TSI Bus Electrical Parameters

SB-TSI conforms to most of the I3C fast-mode electrical parameters. See the Electrical Data Sheet for the processor family for electrical parameters.

6.3.4 Pass-FET Option

The KBC may not have the capability to directly interface to SB-TSI. Pass FETs may be used to create two [SMBus](#) segments, see Figure 4.

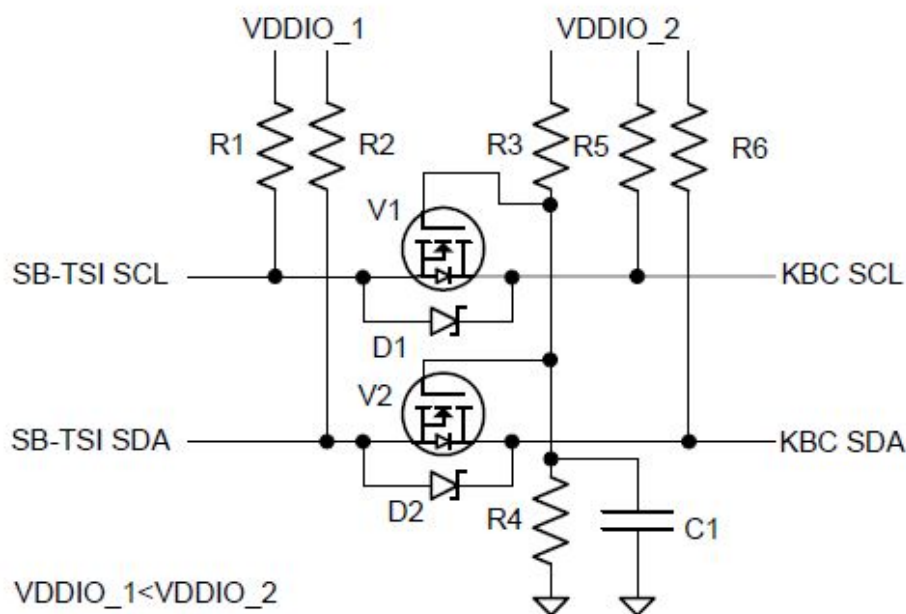


Figure 38: Pass FET Implementation

Notes:

- SCL and SDA pull-up resistors (R5 and R6, respectively) 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 V_{gs} above the lower rail voltage. A resistive divider is shown, but a convenient power rail would 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 $V_{ol} < V_{il}$ on low side)

6.4 SB-TSI Registers

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

SBTSIx01 [CPU Integer Temperature] (SBTSI::CpuTempInt)	
Read-only.	
The CPU temperature is calculated by adding the CPU temperature offset (SBTSI::CpuTempOffInt, SBTSI::CpuTempOffDec) to the processor control temperature (Tctl). SBTSI::CpuTempInt and SBTSI::CpuTempDec combine to return the CPU temperature. For the temperature encoding, see 6.2.5 [SB-TSI Temperature and Threshold Encodings]	
Bits	Description
7:0	CpuTempInt: integer CPU temperature value. Read-only. Reset: Cold,XXh. This field returns the integer portion of the CPU temperature.

SBTSIx02 [SB-TSI Status] (SBTSI::Status)	
Read-only, Volatile.	
If SBTSI::AlertConfig[AlertCompEn] == 0, the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. See 6.2.3 [Alert Behavior].	
Bits	Description
7:5	Reserved.
4	TempHighAlert: temperature high alert. Read-only, Volatile . Reset: Cold,X. 1=Indicates that the CPU temperature is greater than or equal to the high temperature threshold (SBTSI::HiTempInt, SBTSI::HiTempDec) for SBTSI::AlertThreshold[AlertThr] consecutive samples. 0=Indicates that the CPU temperature is less than the high temperature threshold (SBTSI::HiTempInt, SBTSI::HiTempDec) for SBTSI::AlertThreshold[AlertThr] samples and SBTSI::AlertConfig[AlertCompEn] == 1. Hardware will clear this bit when read if SBTSI::AlertConfig[AlertCompEn] == 0.
3	TempLowAlert: temperature low alert. Read-only, Volatile . Reset: Cold,X. 1=Indicates that the CPU temperature is less than or equal to the low temperature threshold (SBTSI::LoTempInt, SBTSI::LoTempDec) for SBTSI::AlertThreshold[AlertThr] consecutive samples. 0=Indicates the CPU temperature is greater than the low temperature threshold (SBTSI::LoTempInt, SBTSI::LoTempDec) for SBTSI::AlertThreshold[AlertThr] samples and SBTSI::AlertConfig[AlertCompEn] == 1. Hardware will clear this bit when read if SBTSI::AlertConfig[AlertCompEn] == 0.
2:0	Reserved.

SBTSIx03 [SB-TSI Configuration] (SBTSI::Config)

Reset: Cold,00h.

The bits in this register are Read-only and can be written by Writing to the corresponding bits in SBTSI::ConfigWr. See 6.2.3 [Alert Behavior] and 6.2.4 [Atomic Read Mechanism].

Bits	Description
7	AlertMask: alert mask. Read-only, Volatile . Reset: Cold,0. 0=ALERT_L pin enabled. 1=ALERT_L pin disabled and does not assert. Read-only; set-by-hardware. Either SBTSI::Status[TempHighAlert] == 1 or SBTSI::Status[TempLowAlert] == 1.
6	RunStop: run stop. Read-only. Reset: Cold,0. 0=Updates to SBTSI::CpuTempInt and SBTSI::CpuTempDec and the alert comparisons are enabled; alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]) continue to update. 1=Updates to SBTSI::CpuTempInt and SBTSI::CpuTempDec and the alert comparisons are disabled; alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]) are stopped. See 6.2.4 [Atomic Read Mechanism] for further details.
5	ReadOrder: atomic read order. Read-only. Reset: Cold,0. 0=Reading SBTSI::CpuTempInt causes the state of SBTSI::CpuTempDec to be latched. 1=Reading SBTSI::CpuTempDec causes the state of SBTSI::CpuTempInt to be latched. See 6.2.4 [Atomic Read Mechanism] for further details.
4	Reserved.
3	SBRMIAddrMode: Addressing mode for SBRMI register space. Read-only. Reset: Cold,0. 1=2 bytes required for SBRMI register address. This bit is set to 1 by MP1 firmware during initialization. This bit should not be modified from I3C side through SBTSI::ConfigWr.
2:1	Reserved.
0	RMISoftReset: trigger RMI soft reset flow. Read-only. Reset: Cold,0. 0=soft reset of SB-RMI I3C hardware completed and BMC may proceed communication with the SB-RMI peripheral. 1=SBI controller triggers MP1 firmware to soft reset the SB-RMI peripheral. See 5.4.3.2.1 [SB-RMI Soft Reset Recovery] for further details.

SBTSIx04 [Update Rate] (SBTSI::UpdateRate)

Read-write. Reset: Cold,08h.

Bits	Description																										
7:0	<p>UpRate: update rate. Read-write. Reset: Cold,08h. This field specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Write access causes a reset of the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]).</p> <p>ValidValues:</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>0.0625 Hz</td></tr> <tr> <td>01h</td><td>0.125 Hz</td></tr> <tr> <td>02h</td><td>0.25 Hz</td></tr> <tr> <td>03h</td><td>0.5 Hz</td></tr> <tr> <td>04h</td><td>1 Hz</td></tr> <tr> <td>05h</td><td>2 Hz</td></tr> <tr> <td>06h</td><td>4 Hz</td></tr> <tr> <td>07h</td><td>8 Hz</td></tr> <tr> <td>08h</td><td>16 Hz</td></tr> <tr> <td>09h</td><td>32 Hz</td></tr> <tr> <td>0Ah</td><td>64 Hz</td></tr> <tr> <td>FFh-0Bh</td><td>Reserved.</td></tr> </table>	Value	Description	00h	0.0625 Hz	01h	0.125 Hz	02h	0.25 Hz	03h	0.5 Hz	04h	1 Hz	05h	2 Hz	06h	4 Hz	07h	8 Hz	08h	16 Hz	09h	32 Hz	0Ah	64 Hz	FFh-0Bh	Reserved.
Value	Description																										
00h	0.0625 Hz																										
01h	0.125 Hz																										
02h	0.25 Hz																										
03h	0.5 Hz																										
04h	1 Hz																										
05h	2 Hz																										
06h	4 Hz																										
07h	8 Hz																										
08h	16 Hz																										
09h	32 Hz																										
0Ah	64 Hz																										
FFh-0Bh	Reserved.																										

SBTSIx07 [High Temperature Integer Threshold] (SBTSI::HiTempInt)

Read-write. Reset: Cold,46h.

The high temperature threshold specifies the CPU temperature that causes ALERT_L to assert if the CPU temperature is greater than or equal to the threshold. SBTSI::HiTempInt and SBTSI::HiTempDec combine to specify the high temperature threshold. See 6.2.5 [SB-TSI Temperature and Threshold Encodings]. Reset value equals 70 °C. Write access causes a reset of the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]). See 6.2.3 [Alert Behavior].

Bits	Description
7:0	HiTempInt: high temperature integer threshold. Read-write. Reset: Cold,46h. This field specifies the integer portion of the high temperature threshold.

SBTSIx08 [Low Temperature Integer Threshold] (SBTSI::LoTempInt)

Read-write. Reset: Cold,00h.

The low temperature threshold specifies the CPU temperature that causes ALERT_L to assert if the CPU temperature is less than or equal to the threshold. SBTSI::LoTempInt and SBTSI::LoTempDec combine to specify the low temperature threshold. See 6.2.5 [SB-TSI Temperature and Threshold Encodings]. Write access causes a reset of the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]). See 6.2.3 [Alert Behavior].

Bits	Description
7:0	LoTempInt: low temperature integer threshold. Read-write. Reset: Cold,00h. This field specifies the integer portion of the low temperature threshold.

SBTSIx09 [SB-TSI Configuration Write] (SBTSI::ConfigWr)

Read-write. Reset: Cold,00h.

This register provides write access to SBTSI::Config.

Bits	Description
7	AlertMask: alert mask. Read-write. Reset: Cold,0. See SBTSI::Config[AlertMask].
6	RunStop: run stop. Read-write. Reset: Cold,0. See SBTSI::Config[RunStop].
5	ReadOrder: atomic read order. Read-write. Reset: Cold,0. See SBTSI::Config[ReadOrder].
4	Reserved.
3	SBRMIAddrMode: Addressing mode for SBRMI register space. Read-write. Reset: Cold,0. Addressing mode needed for SBRMI register address space. MP1 firmware sets this to 1 during initialization to indicate 2-byte mode. Writes from I3C should not modify this bit.
2:1	Reserved.
0	RMISoftReset: trigger RMI soft reset flow. Read-write. Reset: Cold,0. See SBTSI::Config[RMISoftReset].

SBTSIx10 [CPU Decimal Temperature] (SBTSI::CpuTempDec)

Read-only.

See SBTSI::CpuTempInt.

Bits	Description
7:5	CpuTempDec: decimal CPU temperature value. Read-only. Reset: Cold,XXXb. Read-only. This field returns the decimal portion of the CPU temperature.
4:0	Reserved.

SBTSIx11 [CPU Temperature Offset High Byte] (SBTSI::CpuTempOffInt)

Read-write. Reset: Cold,00h.

SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to specify the CPU temperature offset. See 6.2.6 [SB-TSI Temperature Offset Encoding] for encoding details.

Bits	Description
7:0	CpuTempOffInt: CPU temperature integer offset. Read-write. Reset: Cold,00h. This field specifies the integer portion of the CPU temperature offset added to Tctl to calculate the CPU temperature. Write access causes a reset of the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]).

SBTSIx12 [CPU Temperature Decimal Offset] (SBTSI::CpuTempOffDec)

Read-write. Reset: Cold,00h.

See SBTSI::CpuTempOffInt.

Bits	Description
7:5	CpuTempOffDec: CPU temperature decimal offset. Read-write. Reset: Cold,0h. This field specifies the decimal/fractional portion of the CPU temperature offset added to Tctl to calculate the CPU temperature. Write access causes a reset of the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]).
4:0	Reserved.

SBTSIx13 [High Temperature Decimal Threshold] (SBTSI::HiTempDec)

Read-write. Reset: Cold,00h.

See SBTSI::HiTempInt.

Bits	Description
7:5	HiTempDec: high temperature decimal threshold. Read-write. Reset: Cold,0h. This field specifies the decimal portion of the high temperature threshold.
4:0	Reserved.

SBTSIx14 [Low Temperature Decimal Threshold] (SBTSI::LoTempDec)

Read-write. Reset: Cold,00h.

See SBTSI::LoTempInt.

Bits	Description
7:5	LoTempDec: low temperature decimal threshold. Read-write. Reset: Cold,0h. This field specifies the decimal portion of the low temperature threshold.
4:0	Reserved.

SBTSIx22 [Reserved Register] (SBTSI::Reserved)

Read-write. Reset: 00h.

Bits	Description
7:1	Reserved.
0	Reserve: Reserved register. Read-write. Reset: 0. This is reserved register bit.

SBTSIx32 [Alert Threshold Register] (SBTSI::AlertThreshold)

Read-write. Reset: Cold,00h.

See 6.2.3 [Alert Behavior].

Bits	Description
7:3	Reserved.
2:0	AlertThr: alert threshold. Read-write. Reset: Cold,0h. Specifies the number of consecutive CPU temperature samples for which a temperature alert condition needs to remain valid before the corresponding alert bit is set. For SBTSI::AlertConfig[AlertCompEn] == 1, it specifies the number of consecutive CPU temperature samples for which a temperature alert condition need to remain not valid before the corresponding alert bit gets cleared. Write access resets the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]). Details in SBTSI::Status.
ValidValues:	
Value	Description
0h	1 Sample
6h-1h	<Value+1> Samples
7h	8 Samples

SBTSIxBF [Alert Configuration] (SBTSI::AlertConfig)

Read-write.

Bits	Description
7:1	Reserved.
0	AlertCompEn: alert comparator mode enable. Read-write. Reset: Cold,X. 0=SBTSI::Status[TempHighAlert] and SBTSI::Status[TempLowAlert] are Read to clear. 1=SBTSI::Status[TempHighAlert] and SBTSI::Status[TempLowAlert] are Read-only. Write access does not change the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) or the corresponding timer (specified by SBTSI::UpdateRate[UpRate]). See SBTSI::Status.

SBTSIxFE [Manufacture ID] (SBTSI::ManId)

Read-only. Reset: Cold,00h.

Bits	Description
7:1	Reserved.
0	ManId: Manufacture ID. Read-only. Reset: Cold,0. Returns the AMD manufacture ID.

SBTSIxFF [Revision] (SBTSI::Revision)

Read-only. Reset: Cold,04h.

Bits	Description
7:0	Revision: SB-TSI revision. Read-only. Reset: Cold,04h. Specifies the SBI temperature sensor interface revision.

7 Host System Management Port (HSMP)

Table 154: Definitions

Term	Description
HSMP	Host System Management Port
SMN	System Management Network

7.1 Overview

The Host System Management Port (HSMP) is an interface to provide OS-level software with access to system management functions via a set of mailbox registers.

7.2 SMN Mailbox Registers

The set of HSMP mailbox registers can be accessed in SMN space via a pair of SMN_INDEX and SMN_DATA registers:

- SMN_INDEX = Table 165 [BXXD00F0x0C4 (IOHC::NB_SMN_INDEX_3)].
- SMN_DATA = Table 166 [BXXD00F0x0C8 (IOHC::NB_SMN_DATA_3)].

To access a specific HSMP mailbox register, software writes the SMN Address of the HSMP mailbox register into the SMN_INDEX register, and then reads from or writes to the SMN_DATA register.

Table 155: SMN Address for HSMP Mailbox Registers

HSMP Mailbox Register	SMN Address (SMN_INDEX)
Message ID	3B10934h
Message Response	3B10980h
Message Argument_0	3B109E0h
Message Argument_1	3B109E4h
Message Argument_2	3B109E8h
Message Argument_3	3B109ECh
Message Argument_4	3B109F0h
Message Argument_5	3B109F4h
Message Argument_6	3B109F8h
Message Argument_7	3B109FCh

7.2.1 Message ID

A 32-bit read-write register to specify the value for the requested system management function. Writes to this register initiate the command message sequence.

7.2.2 Message Arguments

A set of eight 32-bit read-write registers to specify the input data and to capture the output data for each command message. Software writes input values to these register prior to writing to the Message ID register.

7.2.3 Message Response

An 8-bit read-write register that provides the response when the command message has completed. Software writes zero to this register to clear the Response value prior to writing to the Message ID register.

7.3 Mailbox Protocol

To request a given HSMP function, software executes the following sequence:

- Write zero (0) to the Message Response register.
- Write function arguments into the Message Argument registers.
- Write the function ID into the Message ID register.
- Wait (poll) until the Message Response register returns a non-zero value.

7.3.1 Response Codes

Upon completion of the command sequence, if the Message Response contains a RESULT_OK (01h) value, the command completed successfully and results may be read from the Message Argument registers. Otherwise an error occurred and results from the Message Argument registers are invalid.

Table 156: HSMP Response Codes

Response Value	Description
01h	RESULT_OK. Command completed successfully.
FFh	Command failed – Invalid input Arguments.
FEh	Command failed – Invalid or unsupported message ID.
FDh	Command rejected – Prerequisite to execute the command not satisfied
FCh	Command rejected – MP1 is busy
02h..FBh	Command failed – Other reasons.

7.4 HSMP Functions

Table 157 [HSMP Functions] below describes the supported HSMP functions. Unless otherwise noted, Input and Output Arguments are specified via the Message Argument_0 (Arg0) register. When needed, other message registers such as Argument_1 (Arg1) etc. may be used.

Table 157: HSMP Functions

Function ID	Function Name	Description	Input Arguments	Output Arguments
01h	TestMessage	Increments the input argument value by 1. This is used to check if the HSMP interface is functioning correctly.	[31:0] Any value	[Arg0] + 1
02h	GetSmuVersion or GetMP1FwVersion	Provides the MP1 firmware version.	None	MP1 firmware version
03h	GetInterfaceVersion	Provides the HSMP interface version. Each version supports varying Function IDs as specified in the Table 158 [HSMP Supported Functions Per Interface Version].	None	Interface Version
04h	ReadSocketPower	Provides the average package	None	Socket power

		power consumption. This data may be useful to assess relative variations in power under different operating conditions and workloads. This data should not be used for system power budgeting.		(mWatts)
05h	WriteSocketPowerLimit	<p>Sets the socket power limit. The value written is clipped to the maximum cTDP range for the processor. Note: there is a limit on the minimum power that the processor can operate at; no further socket power reduction occurs if the socket power limit is set below that minimum.</p> <p>NOTE: There are independent Power Limit registers through HSMP and APML; whichever is the most constraining between the HSMP and APML limit at any given time is enforced.</p>	[31:0] Socket Power Limit (mWatts)	None
06h	ReadSocketPowerLimit	Provides the current socket power limit.	None	Socket power limit (mWatts)
07h	ReadMaxSocketPowerLimit	Provides the maximum socket power limit. This specifies the clip value for the WriteSocketPowerLimit function.	None	Max Socket power limit (mWatts)
08h	WriteBoostLimit	<p>Sets a Maximum Frequency Limit on a CPU core defined by the specified ApicId.</p> <ol style="list-style-type: none"> 1. For processors with SMT enabled, writes to different ApicIds that map to the same physical core overwrite the previous write to that core. 2. Values written are constrained to the supported frequency range of the processor. 3. Writes that contain an invalid ApicId result in a Failed response. <p>NOTE: There are independent Boost Limit registers through HSMP and APML; whichever is the most constraining between the HSMP and APML limit at any given time is enforced. See 7.4.1 [HSMP Boost Limit].</p>	[31:16] ApicId, 15:0] Max Frequency (MHz)	None

09h	WriteBoostLimitAllCores	<p>Similar to WriteBoostLimit, however the Frequency Limit applies to all cores in the socket.</p> <p>NOTE: There are independent Boost Limit registers through HSMP and APML; whichever is the most constraining between the HSMP and APML limit at any given time is enforced. See 7.4.1 [HSMP Boost Limit].</p>	[15:0] Max Frequency (MHz)	None
0Ah	ReadBoostLimit	<p>Provides the frequency limit currently enforced through WriteBoostLimit and WriteBoostLimitAllCores. If no boost limits have been specified, Fmax is returned. Writes that contain an invalid ApicId result in a Failed response.</p> <p>NOTE: There are independent Boost Limit registers through HSMP and APML; this message provides only the boost limit associated with HSMP.</p>	[15:0] ApicId	[15:0] Frequency (MHz)
0Bh	ReadProchotStatus	<p>Provides the current PROCHOT status:</p> <p>0 = PROCHOT not asserted. 1 = PROCHOT asserted.</p>	None	PROCHOT Status
0Ch	SetXgmiLinkWidthRange	<p>Sets or Get Max and Min width of <u>xGMI</u> Link as follows:</p> <p>0 = x4 1 = x8 2 = x16</p>	<p>[31] = Set or Get 0 : set the range 1 : get the range If DataIn[31] = 0 [15:8] MinLinkWidth [7:0] MaxLinkWidth Else [15:0] = Reserved</p> <p>NOTE1: Max value must be >= Min value. Invalid configurations result in Failed response.</p> <p>NOTE2: If this function is called from both Primary and Secondary sockets, the largest Width values from either call are used.</p>	<p>[31:16] = Reserved. [15:8] = Min link width. [7:0] = Max link width. These will be the updated values if DataIn[31]=0.</p>
0Dh	APBDisable	Messages APBEnable and	[31] = Set or Get.	[31:9] = Reserved

		<p>APBDisable specify DF (Infinity Fabric™) P-State behavior. DF P-States specify the frequency of clock domains from the CPU core boundary through to and including system memory, where DF P0 has highest performance and higher numbered P-states have lower performance.</p> <p>By default, an algorithm adjusts DF P-States automatically in order to optimize performance. However, this default may be changed to a fixed DF P-State through a CBS option at boot-time. APBDisable may also be used to disable this algorithm and force a fixed DF P-State.</p> <p>NOTE: While the socket is in PC6 or if PROCHOT_L is asserted, the lowest DF P-State (highest numerically) is enforced regardless of the APBEnable/APBDisable state.</p>	<p>0 : set APB_DISABLE 1 : get APB state [30:8] = Reserved. If DataIn[31] = 1 [7:0] = Reserved. Else [7:0] DF P-state (0 to 2)</p>	<p>[8] = APB state 1 : APB is disabled. 0 : APB is enabled If DataOut[8] = 1 [7:0] = DF P-state locked value. Else [7:0] = Reserved.</p>
0Eh	APBEnable	This function enables the DF P-State Performance Boost algorithm. See the APBDisable function for more information.	None	None
0Fh	ReadCurrentFclkMemclk	Provides Infinity Fabric Clock (FCLK) and DRAM Memory Clock (MEMCLK) for the current socket DF P-state.	None	Arg0 = FCLK (MHz), Arg1 = MEMCLK (MHz)
10h	ReadCclkFrequencyLimit	Provides the CPU core clock (CCLK) frequency limit for the socket, from the most restrictive infrastructure limit at the time of the message.	None	Frequency (MHz)
11h	ReadSocketC0Residency	Provides the average C0 residency across all cores in the socket. 100% specifies that all enabled cores in the socket are running in C0.	None	Socket C0 Residency (%)
12h	SetLclkDpmLevelRange	Sets the Max & Min LCLK DPM Level on a given NBIO per socket. The DPM Level is an encoding to represent the PCIe® Link Frequency (LCLK) under a root complex (NBIO), as shown in Table 159 [HSMP LCLK Frequency Per DPM Level].	<p>[23:16] NBIO ID (0 to 3), [15:8] Max DPM Level (0 to 3), [7:0] Min DPM Level (0 to 3).</p> <p>NOTE: Max value</p>	None

		NOTE: These Levels can also be set from APML; the last value received from either HSMP or APML is enforced.	must be \geq Min value.	
13h	GetLclkDpmLevelRange	Gets the Max & Min LCLK DPM Level on a given NBIO per socket. The DPM Level is an encoding to represent the PCIe Link Frequency (LCLK) under a root complex (NBIO), as shown in Table 159 [HSMP LCLK Frequency Per DPM Level]. NOTE: These Levels can also be set from APML; this message returns the current Levels which may have been set from either HSMP or APML.	[23:16] NBIO ID (0 to 3).	[15:8] Max DPM Level (0 to 3), [7:0] Min DPM Level (0 to 3).
14h	GetMaxDDRBandwidthAndUtilization	Provides per socket: 1. Theoretical maximum DDR Bandwidth in GB/s. 2. Current utilized DDR Bandwidth (Read+Write) in GB/s. 3. Current utilized DDR Bandwidth as a percentage of theoretical maximum.	None	[31:20] Max bandwidth in Gbps [19:8] Utilized (R+W) bandwidth in Gbps [7:0] Utilized bandwidth in %
15h	Reserved	N/A	N/A	N/A
16h	GetDIMMTempRangeAndRefreshRate	Get Per-DIMM Temperature Range and Refresh Rate for a given channel based on inband MR4 polling. The refresh rate is determined by the warmest DRAM device across all DIMMs in the channel (highest MR4.OP[2:0] value) and the refresh rate threshold programmed in the UMC. Refer to JEDEC DDR5 SDRAM specification (JESD79) for translation of MR4.OP[2:0] to temperature range.	[7:0] DIMM_ADDRESS See notes under Table 164 [HSMP DIMM ADDRESS Encodings] for DataIn and DataOut field details	[31:4]=Reserved [3]=Refresh Rate 0=1x 1=2x. [2:0]=Temperature Range (MR4.OP[2:0]) of hottest DRAM device on given channel.
17h	GetDIMMPowerConsumption	Get Per-DIMM Power Consumption.	[7:0] DIMM_ADDRESS See notes under Table 164 [HSMP DIMM ADDRESS Encodings] for DataIn and DataOut field details	[31:17] DIMM Power ¹ (mWatt), [16:8] Update Rate (ms), [7:0] DIMM_ADDRESS
18h	GetDIMMThermal	Get Per-DIMM Thermal Sensor	[7:0]	[31:21] Temperature

	Sensor	(2 Sensors per DIMM).	DIMM_ADDRESS See notes under Table 164 [HSMP DIMM ADDRESS Encodings] for DataIn and DataOut field details	(Degrees C), [16:8] Update Rate (ms), [7:0] DIMM_ADDRESS
19h	PwrCurrentActiveFreqLimitSocket	Get Current Active Frequency Limit per Socket.	None	[31:16] Frequency (MHz) [15:0] Source of Limit: As shown in Table 160 [HSMP Frequency Limit Source]
1Ah	PwrCurrentActiveFreqLimitCore	Get Current CCLK limit set per Core.	[15:0] ApicId	[31:0] Frequency (MHz)
1Bh	PwrSviTelemetryAllRails	Get SVI-based Telemetry for all rails.	None	[31:0] Power (mW)
1Ch	GetSocketFreqRange	Get Fmax and Fmin per Socket.	None	[31:16] Fmax (MHz) [15:0] Fmin (MHz)
1Dh	GetCurrentIoBandwidth	Get Current Bandwidth on IO Link.	[15:8] Link ID: As shown in Table 163 [HSMP Link ID Encoding]. [7:3] Reserved [2:0] Bandwidth Type: As shown in Table 161 [HSMP IO Bandwidth Encoding].	[31:0] IO BandWidth (Mbps)
1Eh	GetCurrentXgmiBandwidth	Get Current Bandwidth on xGMI Link.	[15:8] Link ID: As shown in Table 163 [HSMP Link ID Encoding]. [7:3] Reserved [2:0] Bandwidth Type: As shown in Table 162 [HSMP XGMI Bandwidth Encoding].	[31:0] xGMI BandWidth (Mbps)
1Fh	GMI3LinkWidthRange	Set or Get Max and Min GMI3 Link width as follows: 0 = Quarter Width 1 = Half Width 2 = Full Width	[31] = Set or Get 0 : set the range 1 : get the range If DataIn[31] = 0 [15:8]	[31:16] = Reserved. [15:8] = Min link width. [7:0] = Max link width.

			MinLinkWidth [7:0] MaxLinkWidth Else [15:0] = Reserved NOTE: Max value must be >= Min value. Invalid configurations result in a Failed response.	These will be the updated values if DataIn[31]=0.
20h	ControlPcieLinkRate	Set or Get the Link Rate on PCIe devices. Note that if CXL™ device operating at Gen5 rate is present on any link then PCIe rate is locked at Gen5 rate and request to change to Gen4 rate will be ignored.	[31] = Set or Get 0 : set link rate 1 : get link rate [30:8] = Reserved If DataIn[31] = 1 [7:0] = Reserved Else [7:0] = Link Rate encoding : 0=Auto-Detect bandwidth utilization, and set Link Rate accordingly. 1=Limit at Gen4 Rate. 2=Limit at Gen5 Rate	0=Auto-Detect. 1=Limit at Gen4. 2=Limit at Gen5. This will be the updated values if DataIn[31] = 0.
21h	PwrEfficiencyModeSelection	Select or get power efficiency mode. Both set or get return the arbitrated result of the current efficiency mode.	[31]= Set or Get policy 0 = Set policy 1 = Get policy [30:3]=Reserved. [2:0]=Mode Selection: 000b = High Performance Mode. 001b = Power Efficiency Mode. 010b = IO Performance Mode. 011b = Balanced Memory Performance Mode. 100b = Balanced Core Performance Mode. 101b = Balanced Core-Memory Performance Mode. Other = Reserved. See 7.4.3 [HSMP Performance Modes]	[31:3] = Reserved [2:0] = Current efficiency mode.

			.	
22h	DfPstateRange	<p>Set or Get the Max and Min DF P-State.</p> <p>P_State values are: 0 – DFP0 (high performance) 1 – DFP1 2 – DFP2 (low performance)</p> <p>Max value must be <= Min value. DF P-States can be set through APML or HSMP; the value that was last set is enforced.</p>	<p>[31] = Set or Get [30:16]=Reserved. If DataIn[31] = 1 [15:0] = Reserved. Else[15:8]=Min DF P-State. [7:0]=Max DF P-State.</p>	<p>[31:16] = Reserved [15:8] = Min DF P-State. [7:0] = Max DF P-State.</p> <p>This will be the updated values if DataIn[31] = 0.</p>
23h..25h	Reserved			
26h	XgmiPstateRange	<p>Set or Get the Max and Min XGMI P-State.</p> <p>P_State values are: 0 – DFP0 (high performance) 1 – DFP1 (low performance)</p> <p>Max value must be <= Min value. XGMI P-States can be set through APML or HSMP; the value that was last set is enforced.</p>	<p>[31] = Set or Get 0 : Set 1 : Get [30:16]=Reserved. If DataIn[31] = 1 [15:0] = Reserved. Else [15:8]=Min XGMI P-State. [7:0]=Max XGMI P-State.</p>	<p>[31:16] = Reserved [15:8] = Min XGMI P-State. [7:0] = Max XGMI P-State.</p> <p>These will be the updated values if DataIn[31] = 0.</p>
27h	CpuRailIsoFreqPolicy	<p>If a socket wide limit (e.g. PPT) is setting the core clock frequency, then this setting has no effect.</p> <p>For other limiters specific to CPU power rails (e.g. TDC), this policy allows or disables independent core clocks per rail (VDDCR_CPU0 or VDDCR_CPU1).</p>	<p>[31]=Set or Get Policy 0: Set policy 1: Get policy [31:1]=Reserved [0]=Disable independent control 1: all cores on both rails have same frequency limit 0: each rail has independent frequency limit</p>	<p>[31:1] = Reserved [0] = Current policy 1 = both rails have same frequency limit. 0 = each rail has independent frequency limit.</p>
28h	DfcEnable	<p>Set or get DF C-state enabling control.</p> <p>DF C-state is a low power state for IOD.</p>	<p>[31] = Set or Get. 0: Set DFC control 1: Get DFC control [30:1] = Reserved [0] = Enable or Disable for set, Reserved for get. 0: Disable DFC 1: Enable DFC</p>	<p>[31:1] = Reserved [0] = current DFC control setting for get, Reserved for set.</p>
29h-2Fh	Reserved			
30h	GetRaplUnits	Get RAPL (Running Average	None	[31:20]=Reserved.

		Power Limit) Units: Scaling factor for energy (in Joules) is $1/(2^{\text{ESU}})$, and for time (in seconds) is $1/(2^{\text{TU}})$. ESU: energy status units TU: time units		[19:16]=TU Value. [15:13]=Reserved. [12:8]=ESU Value. [7:0]=Reserved.
31h	GetRaplCoreCounter	Get RAPL (Running Average Power Limit) for given core. Energy = Counter * ESU Multiplier (Joules).	[15:0] ApicId	Counter = {Arg1, Arg0} Arg1 constitutes the upper 32 bits and Arg0 constitutes the lower 32 bits.
32h	GetRaplPackageCounter	Get RAPL (Running Average Power Limit) package level Energy Counter. Energy = Counter * ESU Multiplier (Joules).	None	Counter = {Arg1, Arg0} Arg1 constitutes the upper 32 bits and Arg0 constitutes the lower 32 bits.

Table 158: HSMP Supported Functions Per Interface Version

Interface Version	Supported Function IDs
0001h	01h through 11h
0002h	01h through 12h
0003h	01h through 14h
0004h	01h through 15h
0005h	01h through 2Fh
007h	01h through 3Fh

Table 159: HSMP LCLK Frequency Per DPM Level

DPM Level	LCLK Frequency
0	Lowest Freq.
1..3	Highest Freq.

Table 160: HSMP Frequency Limit Source

Source Type Encoding (Arg0[15:0])	Description
Arg0[0]	cHTC-Active
Arg0[1]	PROCHOT
Arg0[2]	Thermal Designed Current (TDC) Limit
Arg0[3]	Package Power Tracking (PPT) Limit
Arg0[4]	OPN Max
Arg0[5]	Reliability Limit (Fused Max or Reliability Monitor Fmax@Vmax)
Arg0[6]	APML Agent
Arg0[7]	HSMP Agent
Arg0[15..8]	Reserved

Note: There may be more than one active limiter.

Table 161: HSMP IO Bandwidth Encoding

Encoding	Bandwidth Type
001b	Aggregate bandwidth
Other	Reserved

Table 162: HSMP XGMI Bandwidth Encoding

Encoding	Bandwidth Type
001b	Aggregate bandwidth
010b	Read bandwidth
100b	Write bandwidth
Other	Reserved

Table 163: HSMP Link ID Encoding

Encoding	Link Type
00000001b	P0
00000010b	P1
00000100b	P2
00001000b	P3
00010000b	G0
00100000b	G1
01000000b	G2
10000000b	G3

Table 164: HSMP DIMM ADDRESS Encodings

DIMM_ADDRESS[7:0] (Mode = 0)	DIMM_ADDRESS[7:0] (Mode = 1)
Bit[7]=0 (for Mode 0) Bit[6]=Temp Sensor#: TS0/1, else Reserved. Bit[5:4]=I2/I3C Port# Bit[3]=Reserved. Bit[2:0]=DIMM SPD Host ID. Refer to JEDEC SPD5118, SPD5108 Hub and Serial Presence Detect Device Standard (JESD300-5).	Bit[7]=1 (for Mode 1) Bit[6]=Temp Sensor#: TS0/1, else Reserved. Bit[5]=Reserved Bit[4]=DIMM#: DIMM0/1. Bit[3:0]=UMC/DDRPHY Instance ID. Refer to 9.1.1.1 [UMC and DDR Phy Logical Mapping].

Notes:

- 1: DIMM Power is a 15-bit unsigned value representing power consumed in mW (0-32767)
- 2: Refresh rate is time since last update (0-511ms). 0 means last update was < 1ms, and 511 means update was >= 511ms.
- 3: DIMM_ADDRESS is encoded per above Table 164 [HSMP DIMM ADDRESS Encodings].
- 4: Temperature (Deg C) is an 11-bit signed value, with a scaling factor of 0.25.
Thus 3FFh=255.75 (1023*0.25), 400h= -256 (-1024*0.25), 1h=0.25 and 7FFh= -0.25 (-1*0.25).

7.4.1 HSMP Boost Limit

The WriteBoostLimit function (ID 08h), provides the ability to limit frequency on a per core basis. However, processors are constrained to frequency resolution per the following requirements:

- Cores are grouped into Core Complexes (CCXes), each of which contains 1 or more cores, depending on the OPN.
- Each [CCX](#) includes a clock generator that supports a resolution of 25 MHz.
- If all cores of a CCX are not programmed for the same boost limit frequency, then the core with the lowest requested frequency will get assigned a frequency that is up to 15-20% lower than the cores with the highest

requested frequency.

- If the specified boost limit frequency of a core is not supported, then the processor selects the next lower supported frequency.

Example 1: If all cores in a CCX are programmed for a boost limit of 3.024 GHz, then all cores of the CCX are limited to 3.000 GHz (next lower supported frequency, with a 25MHz resolution).

Example 2: If three out of four cores of a CCX are programmed for a boost limit of 3.000 GHz, and the last core for 2.700 GHz, then the three high-frequency cores remain set to 3.000 GHz, while the low-frequency core is limited to 2.550 GHz! (~15% lower than highest requested frequency).

7.4.2 ApicId Mapping

The WriteBoostLimit (ID 08h) and ReadBoostLimit (ID 0Ah) functions are directed at logical cores, as specified by their ApicId. Hence, this section describes how to determine which cores are grouped into the same Core Complex ([CCX](#)), and therefore abide by the boost limit frequency resolution rules in section 7.4.1 [HSMP Boost Limit] above.

All cores grouped into a common CCX share a Last-Level (L3) cache, hence Core::X86::CpuId::[CachePropEax3\[NumSharingCache\]](#) can be used to first determine all logical processors (threads) in the cores that share an L3 cache, as follows:

Calculate L3ShareId from the ApicId of each thread as follows:

$$\text{L3ShareId} = \text{ApicId} \gg \log_2 (\text{NumSharingCache} + 1)$$

If (NumSharingCache + 1) is not a power of two, round it up to the next power of two. Then, threads with the same L3ShareId share the L3 cache.

Once all threads grouped into a given CCX have been determined, a representative ApicId per core can be derived by filtering out redundant ApicIds when there are multiple threads enabled per core ([SMT](#)). The number of threads per core can be determined from Core::X86::CpuId::[CoreId\[ThreadsPerCore\]](#), as follows:

Calculate CoreSharedId from the ApicId of each thread as follows:

$$\text{CoreSharedId} = \text{ApicId} \gg \log_2 (\text{ThreadsPerCore} + 1)$$

If (ThreadsPerCore + 1) is not a power of two, round it up to the next power of two. Then, threads with the same CoreSharedId belong to the same core.

7.4.3 HSMP Performance Modes

Performance modes can be set through HSMP or [APML](#); the value that was last set is enforced.

- High Performance Mode: this mode will choose the default [DF](#) P-State algorithm and will try to maximize application performance using the application characteristics. This mode favors core performance and the system boots into this mode. In this mode all DF P-States are available and the default DF P-State and DLWM algorithms are active.
- Power Efficiency Mode: this mode configures the system for power efficiency at the expense of some performance. This mode limits the boost frequency available to the cores and restricts the DF P-States available in the system. This mode also monitors the system load to dynamically adjust performance for maximum power efficiency.
- IO Performance Mode: this mode sets up the Infinity Fabric to maximize the IO sub-system performance. This mode can result in lower core performance in some cases, to maximize IO throughput.
- Balanced Memory Performance Mode: this mode biases the memory subsystem and Infinity Fabric performance

towards efficiency, by lowering the frequency of the fabric and the width of the [xGMI](#) links under light traffic conditions. Core behavior is unaffected. There may be a performance impact under lightly loaded conditions for memory-bound applications compared to the default high performance mode. With higher memory and fabric load, the system becomes similar in performance to the default high performance mode.

- **Balanced Core Performance Mode:** this mode biases toward consistent core performance across varying core utilization levels, by preventing active cores from using the power budget of inactive cores. This mode allows core "boosting" as in the default high performance mode, but does not allow core boost to take advantage of the power budget of inactive cores, resulting in a more efficient operating point for the active cores. The memory subsystem and Infinity Fabric behavior is unaffected. There may be a performance impact under light core utilization conditions compared to the default high performance mode. With high core utilization levels, the performance is similar to the default high performance mode.
- **Balanced Core and Memory Performance Mode:** this mode combines the Balanced Memory Performance and the Balanced Core Performance mode and may result in lower performance under light loads compared to the default high performance mode, but with significant increase in efficiency under light loads. Performance in this mode will be similar to the default high performance mode as the system load increases.

7.5 Registers

7.5.1 IOHC Registers

Table 165: BXXD00F0x0C4 (IOHC::NB_SMN_INDEX_3)

Read-write. Reset: 0000_0000h.	
_iohub0_nbio0_aliasHOST; BXXD00F0x0C4; BXX=_iohub0_nbio0 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub0_nbio1_aliasHOST; BXXD00F0x0C4; BXX=_iohub0_nbio1 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub1_nbio0_aliasHOST; BXXD00F0x0C4; BXX=_iohub1_nbio0 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub1_nbio1_aliasHOST; BXXD00F0x0C4; BXX=_iohub1_nbio1 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub[1:0]_nbio0_aliasSMN; NBCFG[1:0]x000000C4; NBCFG[1:0]=13[C:B]0_0000h	
_iohub[1:0]_nbio1_aliasSMN; NBCFG[3:2]x000000C4; NBCFG[3:2]=13[E:D]0_0000h	
Bits	Description
31:0	NB_SMN_INDEX_3. Read-write. Reset: 0000_0000h. Index value SMN Index/Data pair access. Register access using these reg pairs will have security level 7.

Table 166: BXXD00F0x0C8 (IOHC::NB_SMN_DATA_3)

Read-write. Reset: 0000_0000h.	
_iohub0_nbio0_aliasHOST; BXXD00F0x0C8; BXX=_iohub0_nbio0 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub0_nbio1_aliasHOST; BXXD00F0x0C8; BXX=_iohub0_nbio1 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub1_nbio0_aliasHOST; BXXD00F0x0C8; BXX=_iohub1_nbio0 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub1_nbio1_aliasHOST; BXXD00F0x0C8; BXX=_iohub1_nbio1 instance of NB_BUS_NUM from Table 167 [IOHCMISC[0...3]x00000044 (IOHC::NB_BUS_NUM_CNTL)]	
_iohub[1:0]_nbio0_aliasSMN; NBCFG[1:0]x000000C8; NBCFG[1:0]=13[C:B]0_0000h	

_iohub[1:0]_nbio1_aliasSMN; NBCFG[3:2]x000000C8; NBCFG[3:2]=13[E:D]0_0000h	
Bits	Description
31:0	NB_SMN_DATA_3. Read-write. Reset: 0000_0000h. Index value SMN Index/Data pair access. Register access using these reg pairs will have security level 7.

Table 167: IOHCMISC[0...3]x000000044 (IOHC::NB_BUS_NUM_CNTL)

Read-write. Reset: 0000_0000h.	
GNB Bus Number Control.	
_iohub[1:0]_nbio0_aliasSMN; IOHCMISC[1:0]x000000044; IOHCMISC[1:0]=13[C:B]1_0000h	
_iohub[1:0]_nbio1_aliasSMN; IOHCMISC[3:2]x000000044; IOHCMISC[3:2]=13[E:D]1_0000h	
Bits	Description
31:24	Reserved
23:16	NB_SEGMENT. Read-write. Reset: 00h. Specifies the number of the NBIO segment in a multi-segmented system.
15:9	Reserved
8	NB_BUS_LAT_Mode. Read-write. Reset: 0. Description: NBIO bus number is specified by NB_BUS_NUM. 0 = Local bus number of NBIO is capture from any type 0 configuration request. 1= Use the NB_BUS_NUM to decode for configuration cycles targeting the NBIO bus.
7:0	NB_BUS_NUM. Read-write. Reset: 00h. Specifies the number of the NBIO local bus when NB_BUS_LAT_mode is set.

8 Data Fabric (DF)

8.1 Fabric Performance Monitor Counter (PMC) Events

8.1.1 Background

The Infinity Fabric™ (also known as "Fabric", or as "[DF](#)") is a scalable and modular coherent interconnect for the EPYC™ Server SOC (System On Chip).

The following abstract block diagram of an [IOD](#) shows the some of the key interfaces of the Infinity Fabric. In this conceptual rendering, there are a certain number of CPU Moderator blocks (CCMs), IO Moderator blocks (IOMs), Coherent Stations (CSs), and Infinity Link Controller blocks (LINKs). While the Infinity Fabric stretches across IODs over the [xGMI](#) Infinity Links, the Fabric interface blocks are associated with a given IOD.

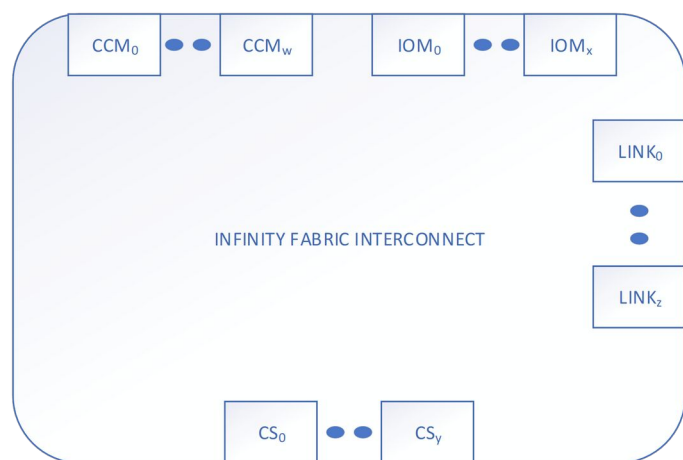


Figure 39: Infinity Fabric™ Interconnect

Table 168: Interface Descriptions

Interface	Descriptions
CS	Coherent Station (Interface to one UMC instance)
CCM	CPU Moderator (interface to one or two CCD instances)
IOM	IO Moderator (interface to one NBIO instance)
LINK	Interface to inter-socket Infinity Fabric xGMI link

Hardware counter events are implemented to provide visibility into the flow of transaction traffic. There are up to 16 counters available that can be programmed to count events concurrently. The number of available Fabric PMCs can be determined by accessing CPUID_Fn80000022_EBX [Extended Performance Monitoring and Debug EBX]. The Fabric [PMC](#) counters can be programmed to select the desired instance and events via [MSR](#) Core::X86::Msr::[DF_PERF_CTL](#). The count values captured in the Fabric PMCs can subsequently be accessed via [MSR](#) Core::X86::Msr::[DF_PERF_CTR](#). The counts are also accessible by software via the RDPMC instruction.

8.1.2 Interfaces and Instance IDs

Each CCM can interface to either one or two Core/Cache Complex Dies (CCDs). Each IOM interfaces to one Northbridge IO (NBIO) instance. Each CS interfaces with either one Unified Memory Controller (UMC) instance that controls a conventional DDR5 memory channel, or one CXL™ link controller instance that controls one or more CXL.mem devices. The LINK blocks are applicable only to Dual Socket (2P) systems, and interface with the respective [xGMI](#) links.

Each [PMC](#) event is selected by means of a 14-bit Event Selection field (EventSel[13:0]).

- Bits [13:6] are used to specify an Instance ID, which corresponds to a unique identifier of one of the interface blocks (CS, CCM, IOM, LINK).
- Bits [5:0] are used to specify the Event Encoding (the specific counter event within the block)

The available Instance IDs for the interface blocks on Family 1Ah Models 00h-0Fh are shown in the table below. On a given processor and platform configuration, not all interfaces may be connected or active.

Table 169: Instance IDs for the Interface Blocks

Interface Block	Instance ID (hexadecimal)
CS0 through CS11 (UMC interface)	0x0 through 0xB
CS12 through CS15 (CXL interface)	0xC through 0xF
CCM0 through CCM7	0x10 through 0x17
IOM0 through IOM7	0x20 through 0x27
LINK0 through LINK5	0x35 through 0x3A

8.1.3 Event Definitions

This section describes the available Fabric PMCs.

8.1.3.1 DATA_BW (Data Bandwidth)

The DATA_BW (data bandwidth) event is intended to help software measure data movement at the key interfaces of the Fabric. The event is designed to count the number of beats of data at the interface being measured. In some cases, there is a degree of selectivity on what beats to measure (for example, beats associated with read transactions versus those associated with write transactions).

The Event Encoding for the DATA_BW [PMC](#) event is 0x1F. A subset of the CCM blocks have two interfaces active. Depending on the processor and platform configuration, these can be used either to connect to two [CCD](#) instances (one interface per CCD), or to connect to a single CCD instance (both interfaces to the same CCD). In such cases, the Event Encoding for the DATA_BW PMC event for the first interface on that CCM is 0x1E, and that for the second interface on that CCM is 0x1F.

Table 170: DATA_BW PMC Event Unitmask for CCM, IOM and CS Interfaces

UnitMask Bit Field	UnitMask Field Name	Description
11:10	SrcDstDieProx	Select proximity of the source or destination of the data beat selected by TxnType 0x0: Reserved 0x1: Count only beats with SrcDst on the same processor die 0x2: Count only beats with SrcDst on a remote processor die 0x3: Count all beats regardless of SrcDst die proximity
9:1	Reserved	Required to be programmed to 0x1FF

0	TxnType	Select type of data beat 0: Data beat in response to a Read request 1: Data beat accompanying a Write request
---	---------	---

Table 171: DATA_BW PMC Event Unitmask for LINK Controller Interface

UnitMask Bit Field	UnitMask Field Name	Description
11:1	Reserved	Required to be programmed to 0x79F
0	Dirn	Select Direction of Data Transfer 0: Outbound Data beat dispatched from this controller to the link 1: Inbound Data beat accepted by this controller from the link

8.1.3.2 Unit of Measurement

The count represents the number of data beats that were monitored at the selected interface over the measurement time window. The number of bytes transferred are computed by the product of the number of beats and the size of the data beat. Dividing the number of bytes transferred by the measurement time window will allow the user to compute the data bandwidth at the selected interface. The following table specifies the size (unit) of the data beats at each interface in bytes for Family 1Ah Models 00h-0Fh.

Table 172: DATA_BW PMC Data Beat Size per Interface and TxnType

Interface	TxnType	Size of data beat counted by DATA_BW
CS	READ	64B
CS	WRITE	64B
CCM	READ	32B
CCM	WRITE	64B
IOM	READ	64B
IOM	WRITE	64B
LINK	READ	64B
LINK	WRITE	64B

8.1.3.3 Usage Notes for SrcDst field

SrcDst captures the notion of either the source of a beat of data, or the destination. The interface being monitored determines whether the field represents the source or the destination.

The general flow for a transaction involves the movement of requests and data between originators (source) and completers (destination).

The Coherent Station (CS) interface is a natural sink point or destination for write memory transactions originating from the [CCD](#) or the NBIO. At the same time, it is the natural originator or source for read data being returned to the CCD or NBIO from the memory controller. Therefore, when monitoring at the CS interface, SrcDst refers to the Source of write data beats (when TxnType == 1), and to the destination of read response data beats (when TxnType == 0).

This perspective is inverted when monitoring the CCM or IOM interfaces. The CCM or IOM are natural originators (or sources) for write transactions (to either memory or [MMIO](#)) originating from the CPU or NBIO respectively. They are also natural sink points or destinations for read data being returned to the CPU or NBIO respectively. Therefore, when monitoring at the CCM or IOM interfaces, SrcDst refers to the eventual destination of write data beats (when TxnType ==

1), and to the source of read response data beats (when TxnType == 0).

Thus, SrcDstDieProx allows the user to select the source of write data beats, or the destination of read response data beats, when being monitored at a natural destination like the CS. Alternatively, it allows the user to select the destination of write data beats, or the source of read response data beats, when being monitored at a natural originator like the CCM or IOM. The SrcDstDieProx field allows selecting based on whether the source or destination is on the local processor node or a remote processor node. This field is only applicable for systems with more than one socket.

8.1.4 Algorithm for Data Bandwidth measurement

- Write to the appropriate instance of Core::X86::Msrr::[DF_PERF_CTL](#) with desired value of EventSelect[13:0] and UnitMask[11:0]
- Write zero to the appropriate instance of Core::X86::Msrr::[DF_PERF_CTR](#) (to initialize the counter)
- Have software wait a reasonable/appropriate amount of time.
- Read the Core::X86::Msrr::[DF_PERF_CTR](#) count value with the RDPMC instruction.
- Convert the count to the number of bytes using the Beat Size table above.
- Divide the number of bytes by the amount of time that occurred between the write of zero to Core::X86::Msrr::[DF_PERF_CTR](#) and the read of Core::X86::Msrr::[DF_PERF_CTR](#), to obtain a throughput value in bytes/second.

8.1.5 Examples for EventSelect and UnitMask

8.1.5.1 Examples 1:

At the Coherent Station 0, count the number of write data beats received from all sources on the Local processor node.

EventSelect[13:0] = {InstanceId[13:6], Event[5:0]} = {8'h0, 6'h1F} = 0x1F

UnitMask[11:0] = {SrcDstDieProx[11:10], Reserved[9:1], TxnType[0]} = {2'h1, 9'h1FF, 1'h1} = 0x7FF

8.1.5.2 Examples 2:

At CPU Moderator 3 Interface 0, count the number of read response data beats received from all destinations on all processor nodes

EventSelect[13:0] = {InstanceId[13:6], Event[5:0]} = {8'h13, 6'h1E} = 0x4DE

UnitMask[11:0] = {SrcDstDieProx[11:10], Reserved[9:1], TxnType[0]} = {2'h3, 9'h1FF, 1'h0} = 0xFFE

8.1.5.3 Examples 3:

At LINK Controller 2, count the number of outbound data beats (from the controller towards the link)

EventSelect[13:0] = {InstanceId[13:6], Event[5:0]} = {8'h2F, 6'h1F} = 0xBDF

UnitMask[11:0] = {Reserved[11:1], Dirn[0]} = {11'h79F, 1'h0} = 0xF3E

8.2 DF Configuration Register Indirect Access

8.2.1 Indirect Access Registers

Indirect access to [DF](#) configuration registers requires setup of a specific registers:

DF::FabricIndirectConfigAccessAddress register (FICAA), and the use of associated

DF::FabricIndirectConfigAccessDataLo, DF::FabricIndirectConfigAccessDataHi registers (FICADL and FICADH;

FICAD for both).

There are six sets of indirect access registers, with three registers in each set: one for address and two for data (to support 64-bit accesses). DF provides six sets of FICAA/FICAD (FICAA[5:0], and paired FICADL[5:0], FICADH[5:0]) and associates them by index. For example, a write through FICADL[2] is routed to the register specified in FICAA[2]. Using one FICAA/FICAD to access a different instance of FICAA/FICAD has undefined results.

The sets are assigned to specific sources:

- set 0: [PSP](#)
- set 1: SMU
- set 2: Microcode
- set 3: BIOS, operating-system, and drivers
- set 4: LW-[SMI](#)
- set 5: Reserved

Accesses using single-instance and 64-bit registers requires use of indirect access.

8.2.2 DF Indirect Access Procedure

Indirect access procedure is:

1. Select the appropriate FICAA/FICAD according to the source.
2. Write FICAA, specifying the following fields. Note that the fully concatenated value can be found within the register description of the targeted register.
 1. CfgRegInstID: The instance ID.
 2. IndCfgAccFuncNum: [DF](#) Function number.
 3. IndCfgAccRegNum : DF Offset number.
 4. CfgRegInstAccEn:
 1. Set to 1: Single-instance access.
 2. Set to 0: broadcast
 5. SixtyFourBitRegEn: set to 1 when accessing a 64-bit register.
3. For reads:
 1. Read FICADL. If reading 64-bit registers, the request must be made with a 64-bit register access to DF::FabricIndirectConfigAccessDataLo.
4. For writes:
 1. Write FICADL. If writing 64-bit registers, the request must be made with a 64-bit access to DF::FabricIndirectConfigAccessDataLo.

8.3 DRAM Address Maps

The processor routes DRAM accesses using DRAM address mapping registers, which must be accessed per-instance as each specifies a different address range. Software programs DRAM Base/Limit pairs for each component as summarized below:

Table 173: DF DRAM Address Map Ranges

Moderator	Coherent Station (CS)	PIE
20	4	20

List of applicable registers:

- DF::DramBaseAddress
- DF::DramLimitAddress
- DF::DramHoleControl

- DF::DramOffset

8.4 Registers

8.4.1 Function 4 Registers

D18F4x040 [Fabric Configuration Access Control (FCAC)] (DF::FabricConfigAccessControl)	
Read-write. Reset: 0000_0000h.	
Configuration register visible for documentation purposes only.	
_n[5:0]_aliasHOST; D18F4x040	
_aliasSMN; D18F4x000000040; D18F4=4900_4000h	
Bits	Description
31:2	Reserved.
1	CfgRegInstAccRegLock . Read-write. Reset: 0. Init: 1. This field is fixed to 1.
0	Reserved.

D18F4x0[80...94] [Fabric Indirect Configuration Access Address (FICAA)] (DF::FabricIndirectConfigAccessAddress)	
Read-write. Reset: 0000_0000h.	
For more information, see 8.2 [DF Configuration Register Indirect Access].	
_n[5:0]_aliasHOST; D18F4x0[94,90,8[C,8,4,0]]	
_n[5:0]_aliasSMN; D18F4x0000_00[94,90,8C,88,84,80]; D18F4=4900_4000h	
Bits	Description
31:25	Reserved.
24:16	CfgRegInstID . Read-write. Reset: 000h. Identifies the instanceID of the register to access.
15	Reserved.
14	SixtyFourBitRegEn . Read-write. Reset: 0. Specifies 64-bit register access using both the upper and lower 32 bit data registers.
13:11	IndCfgAccFuncNum . Read-write. Reset: 0h. Function number of the local DF configuration register to be accessed through DF::FabricIndirectConfigAccessDataLo and DF::FabricIndirectConfigAccessDataHi.
10:1	IndCfgAccRegNum . Read-write. Reset: 000h. Description: Register number of the local DF Configuration register to be accessed through DF::FabricIndirectConfigAccessDataLo and DF::FabricIndirectConfigAccessDataHi. The register offset address is {IndCfgAccRegNum[11:2], 00b}.
0	CfgRegInstAccEn . Read-write. Reset: 0. 0=Target all instances. 1=Target the instance specified by CfgRegInstID. Configuration Register Instance Access Enable.

D18F4x0[A0...C8] [Fabric Indirect Configuration Access Data Lo (FICAD)] (DF::FabricIndirectConfigAccessDataLo)	
Read-write, Volatile. Reset: 0000_0000h.	
Accesses through this register affect the lower 32 bits of the register specified in DF::FabricIndirectConfigAccessAddress. Fabric Indirect Configuration Access Data Registers at offsets xC and x4 are to the upper 32 bits of a 64 bit access. Registers at offsets x8 and x0 are to the lower 32 bits. For more information, see 8.2 [DF Configuration Register Indirect Access].	
_n[5:0]_aliasHOST; D18F4x0[[C:A][8,0]]	
_n[5:0]_aliasSMN; D18F4x0000_00[C8,C0,B8,B0,A8,A0]; D18F4=4900_4000h	
Bits	Description
31:0	Data . Read-write, Volatile . Reset: 0000_0000h. Indirect Data Lo

D18F4x0[A4...CC] [Fabric Indirect Configuration Access Data Hi (FICAD)] (DF::FabricIndirectConfigAccessDataHi)

Read-write, Volatile. Reset: 0000_0000h.

Accesses through this register affect the upper 32 bits of register specified in DF::FabricIndirectConfigAccessAddress. Fabric Indirect Configuration Access Data Registers at offsets xC and x4 are to the upper 32 bits for 64 bit access. Registers at offsets x8 and x0 registers are to the lower 32 bits. For more information, see 8.2 [DF Configuration Register Indirect Access].

_inst[5:0]_aliasHOST; D18F4x0[[C:A][C,4]]

_inst[5:0]_aliasSMN; D18F4x0000_00[CC,C4,BC,B4,AC,A4]; D18F4=4900_4000h

Bits	Description
31:0	Data. Read-write, <u>Volatile</u> . Reset: 0000_0000h. Indirect Data Hi

D18F4x1B0 [System Fabric ID Mask 0] (DF::SystemFabricIdMask0)

Read-write. Reset: 0080_007Fh.

Describes how to identify the socket / die / component portions of a FabricId for the entire system. See 8.5 [Data Fabric Instance and Fabric IDs].

_instBCST_aliasHOST; D18F4x1B0

_inst[SPF[15:0],CNLI[3:0],PIE0,ICNG[3:0],IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasHOST; D18F4x1B0_x[0[56:47],03[E:B],0[34:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],03[E:B],0[34:00]], IndCfgAccFuncNum=4, IndCfgAccRegNum=0x06C]

_instBCST_aliasSMN; D18F4x000001B0; D18F4=4900_4000h

_inst[SPF[15:0],CNLI[3:0],PIE0,ICNG[3:0],IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasSMN; D18F4x000001B0_x[0[56:47],03[E:B],0[34:00]]; D18F4=4900_4000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],03[E:B],0[34:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x06C]

Bits	Description
31:16	NodeIdMask. Read-write. Reset: 0080h. NodeId Mask for the system. If a bit in this mask is set, then the corresponding bit in the FabricId is used to identify a node. NodeId bits must be contiguous.
15:0	CompIdMask. Read-write. Reset: 007Fh. ComponentId Mask for the system. If a bit in this mask is set, then the corresponding bit in the FabricId is used to identify a component. ComponentId bits must be contiguous.

D18F4x1B4 [System Fabric ID Mask 1] (DF::SystemFabricIdMask1)

Read-write. Reset: 0000_0007h.

Describes how to identify the socket / die / component portions of a FabricId for the entire system. See 8.5 [Data Fabric Instance and Fabric IDs].

_instBCST_aliasHOST; D18F4x1B4

_inst[SPF[15:0],CNLI[3:0],CAKEXGMI[5:0],PIE0,ICNG[3:0],IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasHOST; D18F4x1B4_x[0[56:47],0[3E:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],0[3E:00]], IndCfgAccFuncNum=4, IndCfgAccRegNum=0x06D]

_instBCST_aliasSMN; D18F4x000001B4; D18F4=4900_4000h

_inst[SPF[15:0],CNLI[3:0],CAKEXGMI[5:0],PIE0,ICNG[3:0],IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasSMN; D18F4x000001B4_x[0[56:47],0[3E:00]]; D18F4=4900_4000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],0[3E:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x06D]

Bits	Description
31:12	Reserved.
11:8	SocketIdShft. Read-write. Reset: 0h. SocketId shift amount for the system. Tells how many bits to shift the SocketId by to align it to bit 0. In effect, this field identifies the LSB of the SocketId.
7:4	Reserved.
3:0	NodeIdShft. Read-write. Reset: 7h. NodeId shift amount for the system. Tells how many bits to shift the NodeId by to align it to bit 0. In effect, this field identifies the LSB of the NodeId.

D18F4x1B8 [System Fabric ID Mask 2] (DF::SystemFabricIdMask2)

Read-write. Reset: 0001_0000h.

Describes how to identify the socket / die / component portions of a FabricId for the entire system. See 8.5 [Data Fabric Instance and Fabric IDs].

`_instBCST_aliasHOST; D18F4x1B8``_inst[SPF[15:0],CNLI[3:0],CAKEXGMI[5:0],PIE0,ICNG[3:0],IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasHOST; D18F4x1B8_x[0[56:47],0[3E:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],0[3E:00]], IndCfgAccFuncNum=4, IndCfgAccRegNum=0x06E]``_instBCST_aliasSMN; D18F4x000001B8; D18F4=4900_4000h``_inst[SPF[15:0],CNLI[3:0],CAKEXGMI[5:0],PIE0,ICNG[3:0],IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasSMN; D18F4x000001B8_x[0[56:47],0[3E:00]]; D18F4=4900_4000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],0[3E:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x06E]`

Bits	Description
31:16	SocketIdMask. Read-write. Reset: 0001h. SocketId Mask for the system. If a bit in this mask is set, then the corresponding bit in the NodeId is used to identify a socket. SocketId bits must be contiguous.
15:0	DieIdMask. Read-write. Reset: 0000h. DieId Mask for the system. If a bit in this mask is set, then the corresponding bit in the NodeId is used to identify a die. DieId bits must be contiguous.

8.4.2 Function 7 Registers**D18F7x104 [DRAM Hole Control] (DF::DramHoleControl)**

Read-write. Reset: 0000_0000h.

This register is setup by software as a per-instance register, as the value may differ for each coherent station instance. See register DramAddressCtl and its LgcyMmioHoleEn field.

`_instBCST_aliasHOST; D18F7x104``_inst[SPF[15:0],PIE0,IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasHOST; D18F7x104_x[0[56:47],034,0[2F:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x041]``_instBCST_aliasSMN; D18F7x00000104; D18F7=4900_7000h``_inst[SPF[15:0],PIE0,IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasSMN; D18F7x00000104_x[0[56:47],034,0[2F:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x041]`

Bits	Description
31:24	DramHoleBase. Read-write. Reset: 00h. Dram Hole Base[31:24]. Specifies the base address of the IO hole below the 4GB address level for legacy 32-bit devices. MMIO hole cannot spam multiple DRAM ranges; it must lie within a single DRAM address range. DRAM range which spans the IO hole needs to be hoisted (added) to compensate for the hole.
23:1	Reserved.
0	DramHoleValid. Read-write. Reset: 0. 1=Dram Hole Register is valid. Dram Hole Valid.

D18F7x140...D18F7x148 [DRAM Offset] (DF::DramOffset)

Read-write. Reset: 0000_0000h.

In the CS, these registers are associated with DRAM Address Maps 1 through N; see DF::DramBaseAddress and DF::DramLimitAddress. There is no 'DRAM Offset Register 0' because the first DRAM address in range 0 of every CS's DRAM space is zero; hence, the 'offset' for range 0 is always zero.

_instBCST_n[3:1]_aliasHOST; D18F7x14[8,4,0]

_inst[CMP[3:0],CS[11:0]]_n1_aliasHOST; D18F7x140_x00[F:0]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x00[F:0], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x050]

_inst[CMP[3:0],CS[11:0]]_n2_aliasHOST; D18F7x144_x00[F:0]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x00[F:0], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x051]

_inst[CMP[3:0],CS[11:0]]_n3_aliasHOST; D18F7x148_x00[F:0]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x00[F:0], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x052]

_instBCST_n[3:1]_aliasSMN; D18F7x0000_014[8,4,0]; D18F7=4900_7000h

_inst[CMP[3:0],CS[11:0]]_n1_aliasSMN; D18F7x00000140_x00[F:0]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x00[F:0], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x050]

_inst[CMP[3:0],CS[11:0]]_n2_aliasSMN; D18F7x00000144_x00[F:0]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x00[F:0], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x051]

_inst[CMP[3:0],CS[11:0]]_n3_aliasSMN; D18F7x00000148_x00[F:0]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x00[F:0], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x052]

Bits Description

31:25 Reserved.

24:1 **HiAddrOffset**. Read-write. Reset: 00_0000h. Offset address [51:28] for CS DRAM range number corresponds to its component access instance number. Specifies the normalized (DRAM) address at the base of the associated range.

0 **HiAddrOffsetEn**. Read-write. Reset: 0. 1=The offset specified by HiAddrOffset is added when forming the normalized address. Control addition of HiAddrOffset when forming normalized address. This field must be set to one when HiAddrOffset is non-zero.

D18F7x180 [CS Target Remap 0 Register A] (DF::CsTargetRemap0A)

Read-write. Reset: 0A41_8820h.

Logical to Physical CS Target Remap 0 Register A

_instBCST_aliasHOST; D18F7x180

_inst[SPF[15:0],PIE0,IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasHOST; D18F7x180_x[0[56:47],034,0[2F:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x060]

_instBCST_aliasSMN; D18F7x00000180; D18F7=4900_7000h

_inst[SPF[15:0],PIE0,IOS[7:0],IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_aliasSMN; D18F7x00000180_x[0[56:47],034,0[2F:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x060]

Bits Description

31:30 Reserved.

29:25 **RemapCS5**. Read-write. Reset: 05h. Physical CS mapped to logical CS524:20 **RemapCS4**. Read-write. Reset: 04h. Physical CS mapped to logical CS419:15 **RemapCS3**. Read-write. Reset: 03h. Physical CS mapped to logical CS314:10 **RemapCS2**. Read-write. Reset: 02h. Physical CS mapped to logical CS29:5 **RemapCS1**. Read-write. Reset: 01h. Physical CS mapped to logical CS14:0 **RemapCS0**. Read-write. Reset: 00h. Physical CS mapped to logical CS0

D18F7x200...D18F7x330 [DRAM Base Address] (DF::DramBaseAddress)

Read-write. Reset: 0000_0000h.

limit, ctl, and interleave registers provide mapping to identify the target coherent station of the address. See 1.3 [DRAM Address Maps].

_instBCST_n[19:0]_aliasHOST; D18F7x[33:20]0

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n0_aliasHOST; D18F7x200_x[0[56:47],034,0[2F:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x080]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n1_aliasHOST; D18F7x210_x[0[56:47],034,0[27:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x084]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n2_aliasHOST; D18F7x220_x[0[56:47],034,0[27:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x088]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n3_aliasHOST; D18F7x230_x[0[56:47],034,0[27:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x08C]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n4_aliasHOST; D18F7x240_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x090]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n5_aliasHOST; D18F7x250_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x094]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n6_aliasHOST; D18F7x260_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x098]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n7_aliasHOST; D18F7x270_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x09C]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n8_aliasHOST; D18F7x280_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0A0]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n9_aliasHOST; D18F7x290_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0A4]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n10_aliasHOST; D18F7x2A0_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0A8]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n11_aliasHOST; D18F7x2B0_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0AC]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n12_aliasHOST; D18F7x2C0_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0B0]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n13_aliasHOST; D18F7x2D0_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0B4]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n14_aliasHOST; D18F7x2E0_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0B8]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n15_aliasHOST; D18F7x2F0_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0BC]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n16_aliasHOST; D18F7x300_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0C0]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n17_aliasHOST; D18F7x310_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0C4]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n18_aliasHOST; D18F7x320_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0C8]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n19_aliasHOST; D18F7x330_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0CC]

_instBCST_n[19:0]_aliasSMN; D18F7x0000_0[33:20]0; D18F7=4900_7000h

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n0_aliasSMN; D18F7x00000200_x[0[56:47],034,0[2F:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x080]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n1_aliasSMN; D18F7x00000210_x[0[56:47],034,0[27:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x084]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n2_aliasSMN; D18F7x00000220_x[0[56:47],034,0[27:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x088]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n3_aliasSMN; D18F7x00000230_x[0[56:47],034,0[27:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x08C]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n4_aliasSMN; D18F7x00000240_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x090]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n5_aliasSMN; D18F7x00000250_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x094]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n6_aliasSMN; D18F7x00000260_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x098]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n7_aliasSMN; D18F7x00000270_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x09C]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n8_aliasSMN; D18F7x00000280_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0A0]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n9_aliasSMN; D18F7x00000290_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0A4]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n10_aliasSMN; D18F7x000002A0_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0A8]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n11_aliasSMN; D18F7x000002B0_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0AC]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n12_aliasSMN; D18F7x000002C0_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B0]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n13_aliasSMN; D18F7x000002D0_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B4]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n14_aliasSMN; D18F7x000002E0_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B8]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n15_aliasSMN; D18F7x000002F0_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0BC]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n16_aliasSMN; D18F7x00000300_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C0]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n17_aliasSMN; D18F7x00000310_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C4]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n18_aliasSMN; D18F7x00000320_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C8]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n19_aliasSMN; D18F7x00000330_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0CC]	
Bits	Description
31:28	Reserved.
27:0	DramBaseAddr. Read-write. Reset: 000_0000h. Dram Base address[55:28]

D18F7x204...D18F7x334 [DRAM Limit Address] (DF::DramLimitAddress)

Read-write. Reset: 0000_0000h.

The 8 DRAM base, limit, ctl, and interleave registers provides mapping to identify the target coherent station of the address. See DF::DramBaseAddress.

_instBCST_n[19:0]_aliasHOST; D18F7x[33:20]4

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n0_aliasHOST; D18F7x204_x[0[56:47],034,0[2F:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x081]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n1_aliasHOST; D18F7x214_x[0[56:47],034,0[27:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x085]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n2_aliasHOST; D18F7x224_x[0[56:47],034,0[27:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x089]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n3_aliasHOST; D18F7x234_x[0[56:47],034,0[27:00]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x08D]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n4_aliasHOST; D18F7x244_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x091]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n5_aliasHOST; D18F7x254_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x095]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n6_aliasHOST; D18F7x264_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x099]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n7_aliasHOST; D18F7x274_x[0[56:47],034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x09D]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n8_aliasHOST; D18F7x284_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0A1]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n9_aliasHOST; D18F7x294_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0A5]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n10_aliasHOST; D18F7x2A4_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0A9]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n11_aliasHOST; D18F7x2B4_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0AD]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n12_aliasHOST; D18F7x2C4_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0B1]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n13_aliasHOST; D18F7x2D4_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0B5]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n14_aliasHOST; D18F7x2E4_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0B9]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n15_aliasHOST; D18F7x2F4_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0BD]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n16_aliasHOST; D18F7x304_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0C1]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n17_aliasHOST; D18F7x314_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0C5]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n18_aliasHOST; D18F7x324_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0C9]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n19_aliasHOST; D18F7x334_x[034,0[27:10]]; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=7, IndCfgAccRegNum=0x0CD]

_instBCST_n[19:0]_aliasSMN; D18F7x0000_0[33:20]4; D18F7=4900_7000h

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n0_aliasSMN; D18F7x00000204_x[0[56:47],034,0[2F:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[2F:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x081]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n1_aliasSMN; D18F7x00000214_x[0[56:47],034,0[27:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x085]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n2_aliasSMN; D18F7x00000224_x[0[56:47],034,0[27:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x089]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0],CMP[3:0],CS[11:0]]_n3_aliasSMN; D18F7x00000234_x[0[56:47],034,0[27:00]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:00]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x08D]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n4_aliasSMN; D18F7x00000244_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x091]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n5_aliasSMN; D18F7x00000254_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x095]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n6_aliasSMN; D18F7x00000264_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x099]

_inst[SPF[15:0],PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n7_aliasSMN; D18F7x00000274_x[0[56:47],034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[0[56:47],034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x09D]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n8_aliasSMN; D18F7x00000284_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0A1]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n9_aliasSMN; D18F7x00000294_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0A5]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n10_aliasSMN; D18F7x000002A4_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0A9]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n11_aliasSMN; D18F7x000002B4_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0AD]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n12_aliasSMN; D18F7x000002C4_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B1]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n13_aliasSMN; D18F7x000002D4_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B5]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n14_aliasSMN; D18F7x000002E4_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B9]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n15_aliasSMN; D18F7x000002F4_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0BD]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n16_aliasSMN; D18F7x00000304_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C1]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n17_aliasSMN; D18F7x00000314_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C5]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n18_aliasSMN; D18F7x00000324_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C9]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n19_aliasSMN; D18F7x00000334_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0CD]	
Bits	Description
31:28	Reserved.
27:0	DramLimitAddr. Read-write. Reset: 000_0000h. DRAM limit address[55:28]

_instP1E0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0])_n10_aliasSMN; D18F7x000002A8_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0AA]

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n11_aliasSMN; D18F7x000002B8_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0AE]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n12_aliasSMN; D18F7x000002C8_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B2]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n13_aliasSMN; D18F7x000002D8_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B6]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n14_aliasSMN; D18F7x000002E8_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0BA]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n15_aliasSMN; D18F7x000002F8_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0BE]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n16_aliasSMN; D18F7x00000308_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C2]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n17_aliasSMN; D18F7x00000318_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C6]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n18_aliasSMN; D18F7x00000328_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0CA]	
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n19_aliasSMN; D18F7x00000338_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0CE]	
Bits	Description
31:24	Reserved.
23:16	DstFabricID. Read-write. Reset: 00h. Description: When CS interleaving is disabled (see the DramAddressIntlv register and its IntLvNumChan field) which provides the target CS FabricId for this address map. When CS interleaving is enabled, this field provides the CS Fabric ID where interleave starts.
15	HashIntlvCtl1T. Read-write. Reset: 1. Enables inclusion of address bits on the 1TB boundary (bits 40, 41, 42) in a hashed interleave computation.
14:12	Reserved.
11	DramColSwizzle. Read-write. Reset: 0. Enables swizzle mode which groups 1KB memory blocks to column select bits
10	HashIntlvCtl1G. Read-write. Reset: 1. Enables inclusion of address bits on the 1GB boundary (bits 30, 31, 32) in a hashed interleave computation.
9	HashIntlvCtl2M. Read-write. Reset: 1. Enables inclusion of address bits on the 2MB boundary (bits 21, 22, 23) in a hashed interleave computation.
8	HashIntlvCtl64K. Read-write. Reset: 1. Enables inclusion of address bits on the 64KB boundary (bits 16, 17, 18) in a hashed interleave computation.
7	HashIntlvCtl4K. Read-write. Reset: 1. Enables inclusion of address bits on the 4KB boundary (bits 12, 13, 14) in a hashed interleave computation.
6:5	RemapSel. Read-write. Reset: 0h. Remap Select
4	RemapEn. Read-write. Reset: 0. Remap Enable
3	Reserved.
2	SCM. Read-write. Reset: 0. 0=Memory type for this address map is volatile. 1=Memory type for this address map is Storage Class Memory. If set, this field indicates that the addresses described by this address map are stored in Storage Class Memory.
1	LgcyMmioHoleEn. Read-write. Reset: 0. 0=Memory hoisting is not enabled. 1=Enable memory hoisting for this address range. Description: This field is only permitted to be set in at most one DRAM range. BIOS sets this bit for an address range that spans the 4GB boundary and contains a hole for addresses used by legacy MMIO . Additional information needs to be programmed; see DF::DramHoleControl.
0	AddrRngVal. Read-write. Reset: 0. Address range is valid (enabled/disabled). 0=Address range number is disabled and 1=Address range number is valid; corresponds to its component access instance number.


```
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n10_aliasSMN; D18F7x000002AC_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0AB]
```

_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n11_aliasSMN; D18F7x000002BC_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0AF]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n12_aliasSMN; D18F7x000002CC_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B3]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n13_aliasSMN; D18F7x000002DC_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0B7]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n14_aliasSMN; D18F7x000002EC_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0BB]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n15_aliasSMN; D18F7x000002FC_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0BF]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n16_aliasSMN; D18F7x0000030C_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C3]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n17_aliasSMN; D18F7x0000031C_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0C7]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n18_aliasSMN; D18F7x0000032C_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0CB]							
_inst[PIE0,IOM[7:0],IOMMU[3:0],ACM[3:0],CCM[7:0]]_n19_aliasSMN; D18F7x0000033C_x[034,0[27:10]]; D18F7=4900_7000h; FabricIndirectConfigAccessAddress[CfgRegInstID=0x[034,0[27:10]], IndCfgAccFuncNum=0, IndCfgAccRegNum=0x0CF]							
Bits	Description						
31:19	Reserved.						
18	IntLvNumSockets. Read-write. Reset: 0. 0=No socket interleave. 1=2 sockets. Description: Specifies the number of sockets across which addresses are interleaved. Note: A non-zero value is only supported with the following IntLvNumChan encodings: 0x0C, 0x0E, 0x12, 0x22, 0x23, and 0x27.						
17:14	Reserved.						
13:12	IntLvNumDies. Read-write. Reset: 0h. Encoded value which specifies the number of dies across which addresses are interleaved. ValidValues:						
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>1 die (no interleave)</td></tr> <tr> <td>3h-1h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	1 die (no interleave)	3h-1h	Reserved.
Value	Description						
0h	1 die (no interleave)						
3h-1h	Reserved.						
11:10	Reserved.						
9:4	IntLvNumChan. Read-write. Reset: 00h. Encoded value for this address range which specifies the number of coherent stations across which addresses are interleaved. Hash interleaves must program IntLvAddrSel to select address bit [8].						

	ValidValues:	
	Value	Description
	00h	1 channel (no interleave)
	0Bh-01h	Reserved.
	0Ch	IntLvNumSockets == 0 ? 16 channel 1K interleave hash : 8 channel 1K interleave hash
	0Dh	Reserved.
	0Eh	24 channel 1K interleave hash
	0Fh	Reserved.
	10h	2 channel 1K interleave hash
	11h	4 channel 1K interleave hash
	12h	IntLvNumSockets == 0 ? 8 channel 1K interleave hash : 4 channel 1K interleave hash
	13h	3 channel 1K interleave hash
	14h	6 channel 1K interleave hash
	15h	12 channel 1K interleave hash
	16h	5 channel 1K interleave hash
	17h	10 channel 1K interleave hash
	1Fh-18h	Reserved.
	20h	2 channel 2K interleave hash
	21h	4 channel 2K interleave hash
	22h	IntLvNumSockets == 0 ? 8 channel 2K interleave hash : 4 channel 2K interleave hash
	23h	IntLvNumSockets == 0 ? 16 channel 2K interleave hash : 8 channel 2K interleave hash
	24h	3 channel 2K interleave hash
	25h	6 channel 2K interleave hash
	26h	12 channel 2K interleave hash
	27h	24 channel 2K interleave hash
	28h	5 channel 2K interleave hash
	29h	10 channel 2K interleave hash
	3Fh-2Ah	Reserved.
3	Reserved.	
2:0	IntLvAddrSel. Read-write. Reset: 0h. Encoded value for this address range which specifies the starting address bit used for interleaving. The number of address bits used for interleaving depends on the number of channels across which they are interleaved. Values not listed are RESERVED	
	ValidValues:	
	Value	Description
	0h	Addr[8]
	1h	Addr[9]
	2h	Addr[10]
	3h	Addr[11]
	4h	Addr[12]
	7h-5h	Reserved.

8.5 Data Fabric Instance and Fabric IDs

Table 174: InstanceID and ComponentID assignment

Component	InstanceID	ComponentID
-----------	------------	-------------

Coherent Station 0 (UMC0)	00h	00h
Coherent Station 1 (UMC1)	01h	01h
Coherent Station 2 (UMC2)	02h	02h
Coherent Station 3 (UMC3)	03h	03h
Coherent Station 4 (UMC4)	04h	04h
Coherent Station 5 (UMC5)	05h	05h
Coherent Station 6 (UMC6)	06h	06h
Coherent Station 7 (UMC7)	07h	07h
Coherent Station 8 (UMC8)	08h	08h
Coherent Station 9 (UMC9)	09h	09h
Coherent Station 10 (UMC10)	0Ah	0Ah
Coherent Station 11 (UMC11)	0Bh	0Bh
Coherent Station 12 (CMP0)	0Ch	0Ch
Coherent Station 13 (CMP1)	0Dh	0Dh
Coherent Station 14 (CMP2)	0Eh	0Eh
Coherent Station 15 (CMP3)	0Fh	0Fh
Coherent Core Moderator 0 (CCM0)	10h	40h
Coherent Core Moderator 1 (CCM1)	11h	41h
Coherent Core Moderator 2 (CCM2)	12h	42h
Coherent Core Moderator 3 (CCM3)	13h	43h
Coherent Core Moderator 4 (CCM4)	14h	44h
Coherent Core Moderator 5 (CCM5)	15h	45h
Coherent Core Moderator 6 (CCM6)	16h	46h
Coherent Core Moderator 7 (CCM7)	17h	47h
Accelerator Coherent Moderator 0 (ACM0)	18h	50h
Accelerator Coherent Moderator 1 (ACM1)	19h	51h
Accelerator Coherent Moderator 2 (ACM2)	1Ah	52h
Accelerator Coherent Moderator 3 (ACM3)	1Bh	53h
Non-coherent Moderator - IOMMU0 (NCM_IOMMU0)	1Ch	73h
Non-coherent Moderator - IOMMU1 (NCM_IOMMU1)	1Dh	71h
Non-coherent Moderator - IOMMU2 (NCM_IOMMU2)	1Eh	72h
Non-coherent Moderator - IOMMU3 (NCM_IOMMU3)	1Fh	70h
I/O Moderator 0 (IOM0_IOHUBM0)	20h	7Bh
I/O Moderator 1 (IOM1_IOHUBM1)	21h	7Fh
I/O Moderator 2 (IOM2_IOHUBM2)	22h	79h
I/O Moderator 3 (IOM3_IOHUBM3)	23h	7Dh
I/O Moderator 4 (IOM4_IOHUBM4)	24h	7Ah
I/O Moderator 5 (IOM5_IOHUBM5)	25h	7Eh
I/O Moderator 6 (IOM6_IOHUBM6)	26h	78h
I/O Moderator 7 (IOM7_IOHUBM7)	27h	7Ch
IOHUBS0 (IOHUBS0)	28h	23h
IOHUBS1 (IOHUBS1)	29h	27h
IOHUBS2 (IOHUBS2)	2Ah	21h
IOHUBS3 (IOHUBS3)	2Bh	25h
IOHUBS4 (IOHUBS4)	2Ch	22h
IOHUBS5 (IOHUBS5)	2Dh	26h
IOHUBS6 (IOHUBS6)	2Eh	20h
IOHUBS7 (IOHUBS7)	2Fh	24h

ICNG0 (ICNG0)	30h	2Ch
ICNG1 (ICNG1)	31h	2Dh
ICNG2 (ICNG2)	32h	2Eh
ICNG3 (ICNG3)	33h	2Fh
PIE (PIE0)	34h	77h
CAKE_XGMI__0	35h	N/A
CAKE_XGMI__1	36h	N/A
CAKE_XGMI__2	37h	N/A
CAKE_XGMI__3	38h	N/A
CAKE_XGMI__4	39h	N/A
CAKE_XGMI__5	3Ah	N/A
CNLI0 (CNLI0)	3Bh	N/A
CNLI1 (CNLI1)	3Ch	N/A
CNLI2 (CNLI2)	3Dh	N/A
CNLI3 (CNLI3)	3Eh	N/A
PFX0 (PFX0)	3Fh	N/A
PFX1 (PFX1)	40h	N/A
PFX2 (PFX2)	41h	N/A
PFX3 (PFX3)	42h	N/A
PFX4 (PFX4)	43h	N/A
PFX5 (PFX5)	44h	N/A
PFX6 (PFX6)	45h	N/A
PFX7 (PFX7)	46h	N/A
SPF0 (SPF0)	47h	N/A
SPF1 (SPF1)	48h	N/A
SPF2 (SPF2)	49h	N/A
SPF3 (SPF3)	4Ah	N/A
SPF4 (SPF4)	4Bh	N/A
SPF5 (SPF5)	4Ch	N/A
SPF6 (SPF6)	4Dh	N/A
SPF7 (SPF7)	4Eh	N/A
SPF8 (SPF8)	4Fh	N/A
SPF9 (SPF9)	50h	N/A
SPF10 (SPF10)	51h	N/A
SPF11 (SPF11)	52h	N/A
SPF12 (SPF12)	53h	N/A
SPF13 (SPF13)	54h	N/A
SPF14 (SPF14)	55h	N/A
SPF15 (SPF15)	56h	N/A
TCDX0	57h	N/A
TCDX1	58h	N/A
TCDX2	59h	N/A
TCDX3	5Ah	N/A
TCDX4	5Bh	N/A
TCDX5	5Ch	N/A
TCDX6	5Dh	N/A
TCDX7	5Eh	N/A
TCDX8	5Fh	N/A

TCDX9	60h	N/A
TCDX10	61h	N/A
TCDX11	62h	N/A
TCDX12	63h	N/A
TCDX13	64h	N/A
TCDX14	65h	N/A
TCDX15	66h	N/A
TCDX16	67h	N/A
TCDX17	68h	N/A
TCDX18	69h	N/A
TCDX19	6Ah	N/A

9 Unified Memory Controller (UMC)

9.1 DDR5 Overview

Each processor [IOD](#) includes twelve Unified Memory Controllers (UMC). Each UMC controls two DDR5 32-bit channels routed to the same DDR5 DIMM.

The UMC interfaces to the processor Data Fabric ([DF](#)) Coherent Station and the System Management Network (SMN).

9.1.1 UMC and DDR Phy Mapping

9.1.1.1 UMC and DDR Phy Logical Mapping

The table below shows the relationship of logical UMC instance to the processor package DDR channel.

Table 175: Instance to Package Channel Mapping for SP5, 12 DDR5 Channels

Instance	Subchannels
UMC0	CA, CB
UMC1	EA, EB
UMC2	FA, FB
UMC3	AA, AB
UMC4	BA, BB
UMC5	DA, DB
UMC6	IA, IB
UMC7	KA, KB
UMC8	LA, LB
UMC9	GA, GB
UMC10	HA, HB
UMC11	JA, JB

Note: Instance channel/subchannel names map to pin signal names. Example: Channels AA,AB mapped to UMC3 correspond to pin signal names MAA_, MAB_. Channels BA,BB mapped to UMC4 correspond to pin signal names MBA_, MBB_. And so on for other UMC instances/subchannels.

9.2 UMC Performance Monitors

The UMC provides Performance Monitor counters that software can use to count specific events that occur within the DDR channel. Each UMC provides multiple 48-bit performance counters. The UMC provides one dedicated clock counter. All UMC performance monitor events can be counted on the remaining counters.

UMC Performance Monitors support the following features:

- 48-bit MEMCLK counter
- 48-bit programmable performance counters

The UMC includes both subchannels in all events counted. The programmable counters can be configured to monitor a variety of performance parameters. The counters support masking functions.

9.2.1 UMC Performance Monitor Events

UMCPMCx00000000 [Counts MemClk cycles] (UMC::PMC::MEMCLK)

Read-write. Reset: 0000_0000h.	
_umc0_instUMCWPHY0; UMCPMCx00000000	
Bits	Description
31	Enable. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:8	Reserved.
7:0	Eventselect. Read-write. Reset: 00h. Event select: programmed to 0x0

UMCPMCx00000005 [ACTIVATE command sent. Cannot be used with SPM.] (UMC::PMC::ACTCMD)

Read-write. Reset: 0000_0000h.	
_umc0_instUMCWPHY0; UMCPMCx00000005	
Bits	Description
31	Enable. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	RdWrMask. Read-write. Reset: 0h. Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	Eventselect. Read-write. Reset: 00h. Event select: programmed to 0x5

UMCPMCx00000006 [PRECHARGE command sent (For a page conflict in DCQ) . Cannot be used with SPM.] (UMC::PMC::PCHGCMD)

Read-write. Reset: 0000_0000h.	
_umc0_instUMCWPHY0; UMCPMCx00000006	
Bits	Description
31	Enable. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	RdWrMask. Read-write. Reset: 0h. Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	Eventselect. Read-write. Reset: 00h. Event select: programmed to 0x6

UMCPMCx0000000A [DDR5: CAS command sent.] (UMC::PMC::CASCMD)

Read-write. Reset: 0000_0000h.

DDR5: Number of CAS commands sent.

_umc0_instUMCWPHY0; UMCPMCx0000000A

Bits	Description
31	Enable. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	RdWrMask. Read-write. Reset: 0h. Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	Eventselect. Read-write. Reset: 00h. Event select: programmed to 0xA

UMCPMCx00000014 [Number of clocks data bus is utilized. Cannot be used with SPM.] (UMC::PMC::DATASLOTCLKS)

Read-write. Reset: 0000_0000h.

_umc0_instUMCWPHY0; UMCPMCx00000014

Bits	Description
31	Enable. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	RdWrMask. Read-write. Reset: 0h. Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	Eventselect. Read-write. Reset: 00h. Event select: programmed to 0x14

9.3 UMC Registers**9.3.1 Controller Registers****UMC00CHx00000000...UMC11CHx0000000C [DRAM CS Base Address] (UMC::BaseAddr)**

Read-write. Reset: 0000_0000h.

The BaseAddr and AddrMask registers translate request addresses to DRAM chipselects. For each chipselect there is a BaseAddr register and an AddrMask register. For each request normalized address, a chipselect[i] is asserted if: CSEnable[i] & ((RequestAddr & ~AddrMask[i]) == (BaseAddr[i] & ~AddrMask[i])). For each even/odd chipselect-pair, an even CS must be populated if the odd CS is populated.

_umc0_cs0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000000; UMC[11:00]CH=00[B:0]5_0000h

_umc0_cs1_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000004; UMC[11:00]CH=00[B:0]5_0000h

_umc0_cs2_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000008; UMC[11:00]CH=00[B:0]5_0000h

_umc0_cs3_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx0000000C; UMC[11:00]CH=00[B:0]5_0000h

Bits	Description
31:1	BaseAddr: Base Address [39:9]. Read-write. Reset: 0000_0000h. Base address for chipselect decoding.
0	CSEnable. Read-write. Reset: 0. 0=Disable. 1=Enable. Enables chipselect decoding.

UMC00CHx00000010...UMC11CHx0000001C [DRAM CS Base Secondary Address] (UMC::BaseAddrSec)

Read-write. Reset: 0000_0000h.

The BaseAddrSec and AddrMaskSec registers translate request addresses to DRAM chipselects.

`_umc0_cs0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000010; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs1_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000014; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs2_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000018; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs3_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx0000001C; UMC[11:00]CH=00[B:0]5_0000h`**Bits Description****31:1 BaseAddr: Base Address [39:9].** Read-write. Reset: 0000_0000h. Base address for chipselect decoding.**0 CSEnable.** Read-write. Reset: 0. 0=Disable. 1=Enable. Enables chipselect decoding.**UMC00CHx00000020...UMC11CHx0000002C [DRAM CS Mask Address] (UMC::AddrMask)**

Read-write. Reset: 0000_0000h.

The BaseAddr and AddrMask registers translate request addresses to DRAM chipselects.

`_umc0_cs0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000020; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs1_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000024; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs2_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000028; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs3_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx0000002C; UMC[11:00]CH=00[B:0]5_0000h`**Bits Description****31:1 AddrMask: Address Mask [39:9].** Read-write. Reset: 0000_0000h. Address mask for chipselect decoding.**0** Reserved.**UMC00CHx00000030...UMC11CHx0000003C [DRAM CS Mask Secondary Address] (UMC::AddrMaskSec)**

Read-write. Reset: 0000_0000h.

The BaseAddrSec and AddrMaskSec registers translate request addresses to DRAM chipselects.

`_umc0_cs0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000030; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs1_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000034; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs2_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000038; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs3_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx0000003C; UMC[11:00]CH=00[B:0]5_0000h`**Bits Description****31:1 AddrMask: Address Mask [39:9].** Read-write. Reset: 0000_0000h. Address mask for chipselect decoding.**0** Reserved.

UMC00CHx00000040...UMC11CHx0000004C [DRAM Address Configuration] (UMC::AddrCfg)

Read-write. Reset: 0015_0508h.

Specifies the number of bits used for DRAM address generation.

`_umc0_cs0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000040; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs1_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000044; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs2_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000048; UMC[11:00]CH=00[B:0]5_0000h``_umc0_cs3_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx0000004C; UMC[11:00]CH=00[B:0]5_0000h`

Bits	Description												
31:30	CSXor : CSXor[1:0] . Read-write. Reset: 0h. Specifies XOR function bits to remap the CS decoder.												
29:22	Reserved.												
21:20	NumBanks . Read-write. Reset: 1h. Specifies the number of total bank address bits, including NumBankGroups. ValidValues:												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>8 Banks (3 bit)</td></tr> <tr> <td>1h</td><td>16 Banks (4 bit)</td></tr> <tr> <td>2h</td><td>32 Banks (5 bit)</td></tr> <tr> <td>3h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	8 Banks (3 bit)	1h	16 Banks (4 bit)	2h	32 Banks (5 bit)	3h	Reserved.		
Value	Description												
0h	8 Banks (3 bit)												
1h	16 Banks (4 bit)												
2h	32 Banks (5 bit)												
3h	Reserved.												
19:16	NumCol . Read-write. Reset: 5h. Specifies the number of column address bits. ValidValues:												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>Bh-0h</td><td>5 + NumCol Bits</td></tr> <tr> <td>Fh-Ch</td><td>Reserved.</td></tr> </table>	Value	Description	Bh-0h	5 + NumCol Bits	Fh-Ch	Reserved.						
Value	Description												
Bh-0h	5 + NumCol Bits												
Fh-Ch	Reserved.												
15:12	Reserved.												
11:8	NumRow . Read-write. Reset: 5h. Specifies the number of row address bits. ValidValues:												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>8h-0h</td><td>10 + NumRow Bits</td></tr> <tr> <td>Fh-9h</td><td>Reserved.</td></tr> </table>	Value	Description	8h-0h	10 + NumRow Bits	Fh-9h	Reserved.						
Value	Description												
8h-0h	10 + NumRow Bits												
Fh-9h	Reserved.												
7	Reserved.												
6:4	NumRM . Read-write. Reset: 0h. Specifies the number of RM bits. ValidValues:												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>No RM (0 bit)</td></tr> <tr> <td>1h</td><td>2x RM (1 bit)</td></tr> <tr> <td>2h</td><td>4x RM (2 bit)</td></tr> <tr> <td>3h</td><td>8x RM (3 bit)</td></tr> <tr> <td>7h-4h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	No RM (0 bit)	1h	2x RM (1 bit)	2h	4x RM (2 bit)	3h	8x RM (3 bit)	7h-4h	Reserved.
Value	Description												
0h	No RM (0 bit)												
1h	2x RM (1 bit)												
2h	4x RM (2 bit)												
3h	8x RM (3 bit)												
7h-4h	Reserved.												
3:2	NumBankGroups . Read-write. Reset: 2h. Specifies the number of BG bits. ValidValues:												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>No Bank Groups (0 bit)</td></tr> <tr> <td>1h</td><td>2 Bank Groups (1 bit)</td></tr> <tr> <td>2h</td><td>4 Bank Groups (2 bit)</td></tr> <tr> <td>3h</td><td>8 Bank Groups (3 bit)</td></tr> </table>	Value	Description	0h	No Bank Groups (0 bit)	1h	2 Bank Groups (1 bit)	2h	4 Bank Groups (2 bit)	3h	8 Bank Groups (3 bit)		
Value	Description												
0h	No Bank Groups (0 bit)												
1h	2 Bank Groups (1 bit)												
2h	4 Bank Groups (2 bit)												
3h	8 Bank Groups (3 bit)												
1:0	Reserved.												

UMC00CHx00000090...UMC11CHx00000094 [DIMM Configuration] (UMC::DimmCfg)

Read-write. Reset: 0000_0000h.

_umc0_dimm0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000090; UMC[11:00]CH=00[B:0]5_0000h

_umc0_dimm1_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000094; UMC[11:00]CH=00[B:0]5_0000h

Bits	Description
31:11	Reserved.
10	PkgRnkTimingAlign. Read-write. Reset: 0. Check: 0. 0=Disable. 1=Enable. Enable Package Rank Timing Alignment.
9	DimmRefDis. Read-write. Reset: 0. Check: 0. 0=Enable. 1=Disable. Per-DIMM Refresh Disable. Ranks are refreshed if (CSEnable & ~(DimmRefDis DisAutoRefresh)).
8	DqMapSwapDis. Read-write. Reset: 0. 0=Enable. 1=Disable. Disables odd rank DQMap bit swapping behavior at the DDR device. The DQMap swap from even -> odd rank is as follows: DQ0 -> DQ1, DQ1 -> DQ0, DQ2 -> DQ3, DQ3 -> DQ2.
7	X16Dram. Read-write. Reset: 0. 0=Disable. 1=Enable. Specifies x16 DRAMs are populated.
6	X4Dram. Read-write. Reset: 0. 0=Disable. 1=Enable. Specifies x4 DRAMs are populated.
5	LRDIMM. Read-write. Reset: 0. 0=Disable. 1=Enable. Specifies an LRDIMM is populated.
4	RDIMM. Read-write. Reset: 0. 0=Disable. 1=Enable. Specifies an RDIMM is populated.
3	CIsCS. Read-write. Reset: 0. Check: 0. 0=Disable. 1=Enable. Specifies the C pins decode as CS for addressing. Reserved if NumRM != 1. Reserved if DDR5 mode.
2	Dram3DS. Read-write. Reset: 0. 0=Disable. 1=Enable. Specifies 3DS DRAMs are populated.
1	OutputInvert. Read-write. Reset: 0. Check: 0. 0=Disable. 1=Enable. Specifies the DIMM has A-side/B-side inverted addressing. Reserved if DDR5 mode.
0	OnDimmMirror. Read-write. Reset: 0. Check: 0. 0=Disable. 1=Enable. Specifies the DIMM has mirrored addressing for MRS commands. Reserved if DDR5 mode.

UMC[00...11]CHx00000100 [UMC Configuration] (UMC::UmcConfig)

Read-write. Reset: 0000_0200h.

_umc0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000100; UMC[11:00]CH=00[B:0]5_0000h

Bits	Description										
31	DramReady. Read-write. Reset: 0. 0=Disable. 1=Enable. Specifies the memory sub-system is initialized and ready for system traffic.										
30:13	Reserved.										
12	DimmEccEn. Read-write. Reset: 0. 0=Disable. 1=Enable. Indicates all populated DIMMs on the channel support ECC check bits.										
11:10	BurstCtrl. Read-write. Reset: 0h. Specifies the burst control for BurstLength. Fixed BL mode must be set in conjunction with DF configuration bit for 64B accesses. Dynamic BL16/BC8 in DDR5 mode.										
ValidValues:											
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Dynamic BL/BC</td></tr> <tr> <td>1h</td><td>Fixed BL</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	Dynamic BL/BC	1h	Fixed BL	3h-2h	Reserved.		
Value	Description										
0h	Dynamic BL/BC										
1h	Fixed BL										
3h-2h	Reserved.										
9:8	BurstLength. Read-write. Reset: 2h. Init: 3h. Specifies the DDR burst length. Constraint: BL2, BL4, and BL8 modes are reserved in DDR5 mode. See: BurstCtrl.										
ValidValues:											
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>BL2</td></tr> <tr> <td>1h</td><td>BL4</td></tr> <tr> <td>2h</td><td>BL8</td></tr> <tr> <td>3h</td><td>BL16</td></tr> </table>	Value	Description	0h	BL2	1h	BL4	2h	BL8	3h	BL16
Value	Description										
0h	BL2										
1h	BL4										
2h	BL8										
3h	BL16										
7:3	Reserved.										
2:0	DdrType. Read-write. Reset: 0h. Init: 1h. Specifies the DRAM type. See: DramTypeDis.										
ValidValues:											
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Reserved.</td></tr> <tr> <td>1h</td><td>DDR5</td></tr> <tr> <td>7h-2h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	Reserved.	1h	DDR5	7h-2h	Reserved.		
Value	Description										
0h	Reserved.										
1h	DDR5										
7h-2h	Reserved.										

UMC[00...11]CHx00000104 [SDP Control] (UMC::SdpCtrl)

Reset: 6040_0008h.

_umc0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx00000104; UMC[11:00]CH=00[B:0]5_0000h

Bits	Description								
31	SdpInit. Reset: 0. 0=Disable. 1=Enable. Initializes the SDP port. BIOS should disable the SDP port for unused UMC channels. AccessType: (UMC::UmcCap[MemChanDis] == 0) ? Read-write : Read-only.								
30:24	DatBufferCount. Read-write. Reset: 60h. Init: 60h. Specifies the number of SDP data buffer credits. ValidValues:								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>60h-01h</td><td>DatBufferCount Credits</td></tr> <tr> <td>7Fh-61h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Reserved.	60h-01h	DatBufferCount Credits	7Fh-61h	Reserved.
Value	Description								
00h	Reserved.								
60h-01h	DatBufferCount Credits								
7Fh-61h	Reserved.								
23:16	CmdBufferCount. Read-write. Reset: 40h. Specifies the number of SDP command buffer credits. ValidValues:								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>80h-01h</td><td>CmdBufferCount Credits</td></tr> <tr> <td>FFh-81h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Reserved.	80h-01h	CmdBufferCount Credits	FFh-81h	Reserved.
Value	Description								
00h	Reserved.								
80h-01h	CmdBufferCount Credits								
FFh-81h	Reserved.								
15:12	SdpAckDly. Read-write. Reset: 0h. Specifies the minimum number of cycles to delay CompClkAck after a OrigClkReq from CS. ValidValues:								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>Fh-0h</td><td><SdpAckDly> * 32 UCLKs</td></tr> </table>	Value	Description	Fh-0h	<SdpAckDly> * 32 UCLKs				
Value	Description								
Fh-0h	<SdpAckDly> * 32 UCLKs								
11	Reserved.								
10	SdpSubChnTokenEn. Read-write. Reset: 0. Init: 1. 0=Disable. 1=Enable. Enable dual DCQ SDP token management. In this mode, UMC will send separate request credits for the two UMC subchannels and DF CS is required to send subchannel information with SDP requests. This mode must be set in conjunction with DF configuration bits. Constraint:(!SdpSubChnTokenEn (VcmEn && (ChanBit == 1))).								
9	DramScrubCrdt. Read-write. Reset: 0. 0=Disable. 1=Enable. Enable reserving credit for scrubs in the credit control logic in FEI. Needs to be set prior to MemClrDone assertion if MemClrEn=1 or prior to DramReady assertion if MemClrEn=0. See: DramScrubCtrl for scrub modes.								
8:4	Reserved.								
3	SdpCancelEn. Read-write. Reset: 1. Check: 0. 0=Disable. 1=Enable. Specifies SDP CANCEL command processing is enabled.								
2	Reserved.								
1	SdpParityEn. Read-write. Reset: 0. Check: 1. 0=Disable. 1=Enable. Enables SDP originator data parity checking.								
0	SdpFatalDatErr. Read-write. Reset: 0. Check: 0. 0=Disable. 1=Enable. Specifies the SDP DATERR indication mode is enabled for all read responses after any FATAL error.								

UMC[00...11]CHx0000014C [DRAM ECC Control] (UMC::EccCtrl)

Reset: 0000_4000h.

Configuration bits for DRAM ECC.

_umc0_inst[UMCWPHY[11:0]]_aliasSMN; UMC[11:00]CHx0000014C; UMC[11:00]CH=00[B:0]5_0000h

Bits	Description
31:20	Reserved.
19	DpprUCErrPoisonEn. Read-write. Reset: 0. 0=Disable. 1=Enable. When this bit is enabled, an uncorrectable error detected on a DPPR read transaction will trigger a poison scrub
18	EccClkGaterDis. Read-write. Reset: 0. 0=Enable. 1=Disable. Disables medium grain clock gating in ECC logic.
17	EccOvrRdCrcErr. Read-write. Reset: 0. 0=Disable. 1=Enable. Enable an override of RD CRC error detection with ECC code detection in replay logic.
16	D5BfEccEn. Read-write. Reset: 0. 0=Disable. 1=Enable. Enable DDR5 Bounded Fault ECC Code.
15	AddrXorEn. Read-write. Reset: 0. Init: 1. 0=Disable. 1=Enable. Enable XOR of write and read data with the normalized address.
14	PinReducedEcc. Read-write. Reset: 1. 0=Disable. 1=Enable. Reduce ECC to 8-bits per 64-bits of data for DDR5.
13:11	Reserved.
10	RdEccEn. Reset: 0. 0=Disable. 1=Enable. Enable DRAM data ECC checking and correction. AccessType: (UMC::UmcCap[EccDis] == 0) ? Read-write : Read-only.
9	EccSymbolSize16. Read-write. Reset: 0. 0=Use EccSymbolSize (x4/x8 symbol). 1=x16 symbol.
8	UCFatalEn. Read-write. Reset: 0. Check: 0. 0=Disable. 1=Enable. Promote uncorrectable errors from deferred to fatal.
7	EccSymbolSize. Read-write. Reset: 0. Check: 0. 0=x4 symbol. 1=x8 symbol. Specifies ECC symbol size. Reserved if EccSymbolSize16 = 1.
6	EccBitInterleaving. Read-write. Reset: 0. 0=Disable. 1=Enable. Enables data burst bit interleaving for ECC.
5	EccHardwareHistoryEn. Read-write. Reset: 0. 0=Disable. 1=Enable. Enables the hardware managed ECC history mechanism for x8 and x16 symbol size.
4	EccBadDramSymEn. Read-write. Reset: 0. 0=Disable. 1=Enable. Enables the software managed ECC history mechanism for x8 and x16 symbol size.
3:1	Reserved.
0	WrEccEn. Reset: 0. 0=Disable. 1=Enable. Enables ECC generation for DRAM data. Also enables BEQ transmit of ECC data. AccessType: (UMC::UmcCap[EccDis] == 0) ? Read-write : Read-only.